Max-Plus Algebra Toolbox for Matlab®

Jarosław Stańczyk
<jaroslaw.stanczyk@up.wroc.pl>

Version 1.7, Friday 17th June, 2016

Contents

Preface

It is a short guide, to the Max–Plus Algebra Toolbox for Matlab®. This toolbox can be useful tool for calculation in the (max, +) algebra and for design and analysis of certain classes of Discrete Event Systems (DESs). The systems for which the (max, +) algebra is a convenient formalism are characterised by the aspect of synchronisation.

The Max–Plus Algebra Toolbox is a collection of functions that extend the capabilities of the Matlab® computing environment. It also correctly works under GNU Octave.

All functions in this toolbox are available in source code as M-files. The code for these functions can be viewed by using the statement:

```
>> type function_name
```

Users can extend the capabilities of this toolbox by writing their own M-files, or by using it in combination with other toolboxes. A short description of all the functions in this toolbox can be obtain by:

```
>> help function_name
```

Technical Conventions

In this contribution, we follow the (max, +) algebra notation used in [Baccelli et al. 1992].

Copyright Information

Feedback

I hope you will find this Max–Plus Algebra Toolbox helpful. If you have any suggestions or comments related to this toolbox or manual please contact. If you are using this toolbox, I would like to be informed, so, please

send me an email, I will let you know about new versions, bug fixed, updates, etc.

Jarosław Stańczyk
<jaroslaw.stanczyk@up.wroc.pl>

The current version of this document is available on
http://gen.up.wroc.pl/stanczyk/mpa/

Installation

Installation in GNU Octave

Max-Plus Algebra Toolbox package is available as a `.tar.gz` file. Using `File Browswer` window go to the directory where the `max-plus-1.7.tar.gz` is located. Then package can be installed from the Octave prompt with the command

```
>> pkg install max-plus-1.7.tar.gz
```

If the package is installed successfully nothing will be printed on the prompt, but if an error occurred during installation it will be reported.
It is possible to install several package versions. If a different version of the package is already installed it will be removed prior to installing the new package. This makes it easy to upgrade and downgrade the version of a package, but makes it impossible to have several versions of the same package installed at once.

To see which packages are installed type

```
>> pkg list

-| Package Name  | Version | Installation directory
-| --------------+---------+-----------------------
-|     max-plus *|   1.7.0 | /home/js/octave/max-plus-1.7.0
```

In this case only one packake is installed, it means `max-plus` version `1.7`. The '*' character next to the package name shows that the image package is loaded and ready for use.

To load the package into octave workspace type

```
>> pkg load max-plus
```

To use the toolbox it is necessary to invoke GNU Octave with option `--traditional` (compatibility with Matlab®), i.e.

```
$ octave --traditional
```

1 Introduction

Many phenomena from manufacturing systems, telecommunication networks and transportation systems can be described as so-called discrete event systems (DES), or discrete event dynamic systems. A DES is a dynamic asynchronous system where the state transitions are initiated by events that occur at discrete instants of time. An event corresponds to the start or the end of an activity. A common property of such examples is that the start of an activity depends on termination of several other activities. Such systems cannot conveniently be described by differential or difference equations, and naturally exhibit a periodic behaviour.

An introduction to DES is given in e.g. [Cassandras and Lafortune 2007]. Many frameworks exist for studying DES. Examples are queueing theory, e.g. [Gross et al. 2008], Petri nets, e.g.[Murata 1989], the (max, +) algebra [Baccelli et al. 1992] and many others. The most widely used technique to analyse DES is computer simulation. Major drawback of simulation is that it often does not give a real understanding of how parameter changes affect important system properties such as stability, robustness and optimality of system performance. Analytical techniques can provide a much better insight in this respect. Therefore, formal methods are to be preferred as tools for modelling, analysis and control of DES. The theory of DES can be divided presently into three main approaches:

- *the logical approach* which considers the occurrence of events or the impossibility of this occurrence and the series of these events, but does not consider the precise time of those occurrences, e.g. an automata and formal language theory [Ramadge and Wonham 1989];
- *the quantitative approach* which addresses the issue of *performance evaluation* (evaluated by the number of events occurring in a given lapse of time), and that of *performance optimisation*, e.g. timed Petri nets or the (max, +) algebra;
- *the stochasic approach* which considers the occurrence of events under some given statistical conditions e.g. semi–Markov precesses [Limnios and Oproşan 2013].

The (max, +) algebra was first introduced in [Cuninghame–Green 1979]. A standard reference is [Baccelli et al. 1992], a brief survey of methods and applications of this algebra is given in [Cohen, Gaubert, and Quadrat 1999], [Bernd Heidergott, G. J. Olsder, and van der Woude 2005] and [De Schutter and Ton van den Boom 2008]. In certain aspects, the (max, +) algebra is comparable to the conventional algebra. In the (max, +) algebra the addition (+) and multiplication (×) operators from the conventional algebra are replaced by the maximization (max) and addition (+) operators, respectively.

Using these operators, a linear description (in the (max, +) algebra sense) of certain non–linear systems (in the conventional algebra) is achieved. Systems for which the (max, +) algebra is a proper formalism are characterized by aspect of synchronization.

In the last decade, a number of new directions appear in (max, +) systems theory, e.g. optimal control [Komenda, El Moudni, and Zerhouni 2001], [Maia et al. 2003], adaptive control [Menguy et al. 2000], stochastic control [B. Heidergott and de Vries 2001], model predictive control [T. van den Boom and De Schutter 2002], [T. J. van den Boom et al. 2003], etc.

This paper presents a software tool for rapid prototyping, modelling, control and analysis of certain classes of DESs. A Max-Plus Algebra Toolbox for Matlab [Stańczyk 2016], [Stańczyk, Mayer, and Raisch 2004] is a set of several dozen of functions implementing major aspects of the (max, +) algebra in the Matlab environment.

There are other tools available in the Internet for computation in (max, +) algebra, e.g. open-source distributed:

- *the MaxPlus Toolbox for Scilab* [MaxPlus Working Group 2003];
- *MAX*: a Maple package [Gaubert 1992];
- *C++ MinMax library*: [Gruet et al. 2015].

Main drawback of first two toolboxes presented above is that are not developed and expanded. Only C++ library is a new one and fast, but is devoted to use in C++ language.

2 (max, +) algebra

2.1 Basic operations

Definition 2.1 Max–plus algebra
The (max, +) algebra is defined as follows:

- $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{-\infty\}$, where $\mathbb{R}$ is the field of real numbers;
- $\forall a, b \in \mathbb{R}_\varepsilon : \quad a \oplus b \equiv \max(a, b)$;
- $\forall a, b \in \mathbb{R}_\varepsilon : \quad a \otimes b \equiv a + b$.

The algebraic structure $\mathbb{R}_{\max} = (\mathbb{R}_\varepsilon, \oplus, \otimes)$ is called *the max-plus algebra*. We introduce the notation of $\varepsilon = -\infty$ and $e = 0$ folowing [Baccelli et al. 1992]. Notation ε and e instead of $-\infty$ and 0 respectively, is used for emphasis their special meanings and to avoid confusion with their roles in the conventional algebra. More specifically, the algebraic structure $\mathbb{R}_{\max}$ is an idempotent, commutative semiring (or dioid). This structure satisfies axioms:

- operation $\oplus$:
  - associativity:
    $$\forall a, b, c \in \mathbb{R}_\varepsilon : (a \oplus b) \oplus c = a \oplus (b \oplus c);$$
  - commutativity:
    $$\forall a, b \in \mathbb{R}_\varepsilon : a \oplus b = b \oplus a;$$
  - a neutral element (ε):
    $$\forall a \in \mathbb{R}_\varepsilon : a \oplus \varepsilon = \varepsilon \oplus a = a;$$
  - idempotent:
    $$\forall a \in \mathbb{R}_\varepsilon : a \oplus a = a;$$
- operation $\otimes$:
  - associativity:
    $$\forall a, b, c \in \mathbb{R}_\varepsilon : (a \otimes b) \otimes c = a \otimes (b \otimes c);$$

– commutative:

$$\forall a, b \in \mathbb{R}_\varepsilon : a \otimes b = b \otimes a;$$

– a neutral element (e):

$$\forall a \in \mathbb{R}_\varepsilon : e \otimes a = a \otimes e = a;$$

– an absorbing element (ε)[1]:

$$\forall a \in \mathbb{R}_\varepsilon : a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon;$$

– distributivity with respect to $\oplus$:

$$\begin{aligned} \forall a, b, c \in \mathbb{R}_\varepsilon : a \otimes (b \oplus c) &= (a \otimes b) \oplus (a \otimes c), \\ (b \oplus c) \otimes a &= (b \otimes a) \oplus (c \otimes a). \end{aligned}$$

We do not use e to denote a neutral element of operation $\otimes$, instead we use its numerical value (0) to avoid confusion with the number $e = \exp(1)$.
Note that non–zero elements do not have an inverse for $\oplus$, because $a \oplus x = b$ does not have a solution if $a > b$.
We will write ab for $a \otimes b$ whenever there is no possible confusion.

Definition 2.2 (max, +) power
Let $a \in \mathbb{R}$ and $b \in \mathbb{R}_\varepsilon$, the a–th (max, +) power of b is denoted by b^a, and corresponds to ab in conventional algebra.
If $a = 0$ then $b^a = b^0 = 0$.
If $b = \varepsilon$ and $a > 0$ then $b^a = \varepsilon^a = \varepsilon$.
If $b = \varepsilon$ and $a < 0$ then $b^a = \varepsilon^a$ is not defined.
$\varepsilon^0 = 0$ by definition.

Definition 2.3 (max, +) inversion

$$\forall a \in \mathbb{R}_\varepsilon : a^{-1} \otimes a = a \otimes a^{-1} = 0 \tag{1}$$

Definition 2.4 (max, +) division
Let $a \in \mathbb{R}_\varepsilon$ and $b \in \mathbb{R}$, the (max, +) division is defined as follows:

$$\frac{a}{b} = a \oslash b = ab^{-1}. \tag{2}$$

If $b = \varepsilon$ then $a \oslash b$ is not defined.

It is possible to derive the min operation from the (max, +) operations as follows:

$$\min(a, b) = ab \oslash (a \oplus b).$$

[1] In this convention $-\infty + \infty = -\infty$.

Table 1: Notations in the (max, +) and the conventional algebra.

(max, +) notation	conventional notation	example
$a \oplus b$	$\max(a, b)$	$3 \oplus 2 = 3$
$a \otimes b$	$a + b$	$3 \otimes 2 = 5$
a^b	$b \times a$	$3^2 = 3 \otimes 3 = 6$
$a^{1/b}$	$a \times \frac{1}{b}$	$3^{1/2} = \sqrt{3} = 1.5$
a^{-b}	$-b \times a$	$3^{-2} = -6$
$\frac{a}{b} = a \oslash b$	$a - b$	$0 \oslash 3 = -3$

Table 2: Scalar's basic functions.

operation	toolbox function				
ε	`mp_zero`	or	`mp_zeros`	or	`-Inf`
0	`mp_one`	or	`mp_ones`	or	`0`
$a \oplus b$	`mp_add(a,b)`				
$a \otimes b$	`mp_multi(a,b)`				
$a^{(-1)}$	`mp_inv(a)`	or	`mp_power(a,-1)`		
a^b	`mp_power(a,b)`				
$a^{1/b}$	`mp_power(a,1/b)`				
a^{-b}	`mp_power(a,-b)`				
$a \oslash b$	`mp_div(a,b)`	or	`mp_multi(a, mp_inv(b))`		

2.2 Matrices

Now, we extend the (max, +) algebra operations to vectors and matrices.

Definition 2.5 Scalar–vector addition
The (max, +) sum of a scalar $a \in \mathbb{R}_\varepsilon$ and a vector $\mathbf{b} \in \mathbb{R}_\varepsilon^n$ is a vector $(a \oplus \mathbf{b}) \in \mathbb{R}_\varepsilon^n$, defined by:

$$(a \oplus \mathbf{b})_i = a \oplus (\mathbf{b})_i, \qquad i = 1, \ldots, n. \tag{3}$$

A scalar–matrix addition is defined in a similar way.

Example 2.6 A (max, +) scalar-vector addition

$$4 \oplus \begin{bmatrix} 2 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \oplus 2 \\ 4 \oplus 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \end{bmatrix}.$$

Definition 2.7 Matrix addition
The sum $\oplus$ of matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}_\varepsilon^{m \times n}$ is a matrix $(\mathbf{A} \oplus \mathbf{B}) \in \mathbb{R}_\varepsilon^{m \times n}$

obtained by adding corresponding entries. That is,

$$(\mathbf{A} \oplus \mathbf{B})_{ij} = (\mathbf{A})_{ij} \oplus (\mathbf{B})_{ij}, \qquad i = 1, \ldots, m;\ j = 1, \ldots, n. \tag{4}$$

Example 2.8 A (max, +) matrix addition

$$\begin{bmatrix} 1 & 6 \\ 8 & 3 \end{bmatrix} \oplus \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix} = \begin{bmatrix} 1 \oplus 2 & 6 \oplus 5 \\ 8 \oplus 3 & 3 \oplus 3 \end{bmatrix} = \begin{bmatrix} 2 & 6 \\ 8 & 3 \end{bmatrix}.$$

Definition 2.9 Scalar–vector multiplication
We define the product of a scalar $a \in \mathbb{R}_\varepsilon$ and a vector $\mathbf{b} \in \mathbb{R}_\varepsilon^n$ as a vector $(a \otimes \mathbf{b}) \in \mathbb{R}_\varepsilon^n$:

$$(a \otimes \mathbf{b})_i = a \otimes (\mathbf{b})_i, \qquad i = 1, \ldots, n. \tag{5}$$

Scalar–matrix multiplication is defined in a similar way.

Example 2.10 A (max, +) scalar–vector multiplication

$$4 \otimes \begin{bmatrix} 2 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \otimes 2 \\ 4 \otimes 8 \end{bmatrix} = \begin{bmatrix} 6 \\ 12 \end{bmatrix}.$$

Definition 2.11 Matrix multiplication
The product $\otimes$ of matrices $\mathbf{A} \in \mathbb{R}_\varepsilon^{m \times p}$ and $\mathbf{B} \in \mathbb{R}_\varepsilon^{p \times n}$ is a matrix $(\mathbf{A} \otimes \mathbf{B}) \in \mathbb{R}_\varepsilon^{m \times n}$, whose (i, j)–entry is the inner product of the i^{th} row of $\mathbf{A}$ with the j^{th} column in $\mathbf{B}$. That is,

$$\begin{aligned} (\mathbf{A} \otimes \mathbf{B})_{ij} &= \bigoplus_{k=1}^{p} (\mathbf{A})_{ik} \otimes (\mathbf{B})_{kj} \qquad (6) \\ &\equiv \max_k ((\mathbf{A})_{ik} + (\mathbf{B})_{kj}), \quad i = 1, \ldots, m;\ j = 1, \ldots, n, \end{aligned}$$

where: $\bigoplus_{j=1}^{m} a_j$ is short–hand for $a_1 \oplus \cdots \oplus a_m$.

A vector-matrix product is defined in a similar way.

Example 2.12 A (max, +) vector-matrix and matrix-matrix products

$$\begin{aligned} \begin{bmatrix} 2 & 8 \end{bmatrix} \otimes \begin{bmatrix} 2 & 0 \\ \varepsilon & 5 \end{bmatrix} &= \\ \begin{bmatrix} (2 \otimes 2) \oplus (8 \otimes \varepsilon) & (2 \otimes 0) \oplus (8 \otimes 5) \end{bmatrix} &= \\ \begin{bmatrix} 4 \oplus \varepsilon & 2 \oplus 13 \end{bmatrix} &= \begin{bmatrix} 4 & 13 \end{bmatrix}; \end{aligned}$$

c)

$$\begin{bmatrix} 1 & 6 & 2 \\ 8 & 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 2 & 5 \\ 3 & 3 \\ 1 & 6 \end{bmatrix} =$$

$$\begin{bmatrix} (1 \otimes 2) \oplus (6 \otimes 3) \oplus (2 \otimes 1) & (1 \otimes 5) \oplus (6 \otimes 3) \oplus (2 \otimes 6) \\ (8 \otimes 2) \oplus (3 \otimes 3) \oplus (4 \otimes 1) & (8 \otimes 5) \oplus (3 \otimes 3) \oplus (4 \otimes 6) \end{bmatrix} =$$

$$\begin{bmatrix} 3 \oplus 9 \oplus 3 & 6 \oplus 9 \oplus 8 \\ 10 \oplus 6 \oplus 5 & 13 \oplus 6 \oplus 10 \end{bmatrix} = \begin{bmatrix} 9 & 9 \\ 10 & 13 \end{bmatrix}.$$

Definition 2.13 Identity matrix
The matrix $\mathbf{I}_n \in \mathbb{R}_\varepsilon^{n \times n}$ with 0's on the main diagonal and ε's elsewhere is called the identity matrix of order n.

Identity matrix is a neutral element for matrices $\otimes$ operation:

$$\mathbf{A} \otimes \mathbf{I} = \mathbf{A} \tag{7}$$

Definition 2.14 Zero matrix
The matrix $\boldsymbol{\varepsilon}^{m \times n}$ with $\left(\boldsymbol{\varepsilon}\right)_{ij} = \varepsilon$ for all i, j, is the zero–matrix.

Zero matrix is a neutral element for matrices $\oplus$ operation:

$$\mathbf{A} \oplus \boldsymbol{\varepsilon} = \mathbf{A} \tag{8}$$

and

$$\mathbf{A} \otimes \boldsymbol{\varepsilon} = \boldsymbol{\varepsilon} \tag{9}$$

Definition 2.15 (max, +) matrix power
Let $\mathbf{A} \in \mathbb{R}_\varepsilon^{n \times n}$ and $b \in \mathbb{N}_0$, the b–th (max, +) power of $\mathbf{A}$ is defined as follows:

$$\mathbf{A}^b = \underbrace{\mathbf{A} \otimes \cdots \otimes \mathbf{A}}_{b} = \bigotimes_{i=1}^{b} \mathbf{A}. \tag{10}$$

If $b = 0$ then $\mathbf{A}^0 = \mathbf{I}$.
If $b = -1$ see theorem 2.16.
If $b \notin \mathbb{N}_0 \cup \{-1\}$ then $\mathbf{A}^b$ is not defined.
$\mathbb{N}_0$ is the set of nonnegative integers.

Theorem 2.16 *[Cuninghame–Green 1979]*
A matrix $\mathbf{A} \in \mathbb{R}_\varepsilon^{n\times n}$ is invertible in the (max, +) algebra if and only if it can be factorized as

$$\mathbf{A} = \mathbf{DP}, \tag{11}$$

where:

$\mathbf{D} \in \mathbb{R}_\varepsilon^{n\times n}$ is a matrix with non-εdiagonal entries
$\mathbf{P} \in \mathbb{R}_\varepsilon^{n\times n}$ is a permutation matrix.

Then

$$\mathbf{A}^{-1} = \mathbf{P}^{-1}\mathbf{D}^{-1}$$

where:

$$(\mathbf{D}^{-1})_{ii} = -(\mathbf{D})_{ii},$$
$$\mathbf{P}^{-1} = \mathbf{P}^T.$$

Example 2.17 *Matrix inversion*
Consider $\mathbf{A} = \begin{bmatrix} \varepsilon & \varepsilon & 3 \\ -2 & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon \end{bmatrix}$.
This matrix is (max, +)–invertible:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \varepsilon & \varepsilon & 3 \\ -2 & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon \end{bmatrix} \\ &= \mathbf{DP} \\ &= \begin{bmatrix} 3 & \varepsilon & \varepsilon \\ \varepsilon & -2 & \varepsilon \\ \varepsilon & \varepsilon & 0 \end{bmatrix} \begin{bmatrix} \varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 0 \end{bmatrix} \end{aligned}$$

We have

$$\begin{aligned} \mathbf{A}^{-1} &= \mathbf{P}^{-1}\mathbf{D}^{-1} \\ &= \begin{bmatrix} \varepsilon & 0 & \varepsilon \\ \varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} \varepsilon & 2 & \varepsilon \\ \varepsilon & \varepsilon & 0 \\ -3 & \varepsilon & \varepsilon \end{bmatrix} \\ &= \begin{bmatrix} \varepsilon & 2 & \varepsilon \\ \varepsilon & \varepsilon & 0 \\ -3 & \varepsilon & \varepsilon \end{bmatrix}. \end{aligned}$$

Lets check:

$$\begin{aligned}\mathbf{A}^{-1}\mathbf{A} &= \begin{bmatrix} \varepsilon & 2 & \varepsilon \\ \varepsilon & \varepsilon & 0 \\ -3 & \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} \varepsilon & \varepsilon & 3 \\ -2 & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon \end{bmatrix} \\ &= \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon \\ \varepsilon & \varepsilon & 0 \end{bmatrix} \\ &= \mathbf{I}\end{aligned}$$

Example 2.18 Not inversible matrix
Consider $\mathbf{A} = \begin{bmatrix} 2 & -1 \\ 3 & 2 \end{bmatrix}$.

$$\begin{aligned}\mathbf{A}^{-1}\mathbf{A} &= \begin{bmatrix} 2 & -1 \\ 3 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 2 & -1 \\ 3 & 2 \end{bmatrix} \\ &= \begin{bmatrix} -2 & -3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 3 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 \\ 3 & 0 \end{bmatrix} \\ &\neq \mathbf{I}\end{aligned}$$

Definition 2.19 (max, +) matrix division
Let $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n\times n}$ and $\mathbf{B} \in \mathbb{R}_{\varepsilon}^{n\times n}$.
If $\mathbf{B}$ is (max, +)–invertible then the (max, +) matrix division $\mathbf{A}$ by $\mathbf{B}$ is defined as follows:

$$\mathbf{A} \oslash \mathbf{B} = \mathbf{B}^{-1}\mathbf{A}. \tag{12}$$

Definition 2.20 Trace of a matrix
The *trace* of a matrix $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n\times n}$ is the sum of the entries on its main diagonal and is denoted by $tr(\mathbf{A})$. That is,

$$tr(\mathbf{A}) = \bigoplus_{i=1}^{n} (\mathbf{A})_{ii}. \tag{13}$$

Equivalently, $tr(\mathbf{A})$ is a maximal element from main diagonal.

Table 3: Functions to operate on matrices.

function	short description	
`mp_zeros(n,m)`	n–by–m zeros matrix	
`mp_ones(n,m)`	n–by–m ones matrix	
`mp_eye(n,m)`	n–by–m identity matrix	$\mathbf{I}_n$
`mp_add(A,B)`	addition of $\mathbf{A}$ and $\mathbf{B}$	$\mathbf{A} \oplus \mathbf{B}$
`mp_multi(A,B)`	multiplication of $\mathbf{A}$ by $\mathbf{B}$	$\mathbf{A} \otimes \mathbf{B}$
`mp_power(A,n)`	n–th power of $\mathbf{A}$	$\mathbf{A}^n$
`mp_inv(A)`	invertion of $\mathbf{A}$	$\mathbf{A}^{-1}$
`mp_div(A,B)`	division of $\mathbf{A}$ by $\mathbf{B}$	$\mathbf{B}^{-1}\mathbf{A}$
`mp_trace(A)`	trace of $\mathbf{A}$	$tr(\mathbf{A})$

2.3 Connection with graph theory

There exists a close relation between the (max, +) algebra and graphs. In this subsection we first give a short introduction to graph theory, and next we give a graph–theoretic interpretation of some basic (max, +) operations and concepts, which will be used later on. For the introduction to the graphs we refer the reader to e.g. [Aldous and Wilson 2000].

Definition 2.21 graph
A *graph* $\mathcal{G}$ is defined as a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of elements called *nodes* (or *vertices*) and $\mathcal{E}$ is a set the elements called *edges*. Each edge joints two nodes.

Definition 2.22 digraph
A *directed graph* (or shorter *digraph*) $\mathcal{G}$ is defined as a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of elements called *nodes* (or *vertices*) and $\mathcal{E}$ is a set the elements which are ordered pairs of nodes called *arcs*.

Definition 2.23 weighted digraph
A digraph is called *weighted digraph* if there exists associated element a_{ij} with each arcs $(j, i) \in \mathcal{E}$. The quantity a_{ij} is called the *weight* of arc (j, i).

Definition 2.24 initial and final node, predecessor and successor
Denote the number of nodes by n and number of the individual nodes $1, 2, \ldots, n$.
If a pair $(i, j) \in \mathcal{E}$, then i is called *initial node* (or *origin*) of the arc (i, j), and j the *final node* (or *destination*) of arc (i, j).
If a pair $(i, j) \in \mathcal{E}$, then i is called *predecessor* of j and j is called

successor of i. The set of all predecessors of j is indicated by $\pi(j)$.

Definition 2.25 path, circuit and acyclic digraph
A *path* ρ is a sequence of nodes $(i_1, i_2, \ldots, i_p)$, $p > 1$, such that $i_j \in \pi(i_{j+1})$, $j = 1, \ldots, p-1$.
Equivalently, a path is a sequence of arcs which connects a sequence of nodes.
A *circuit* is a path, in which the initial and final nodes coincide.
A digraph is *acyclic* if it does not contain circuits.

Definition 2.26 lenght and weight of a path
A *length* of a path (or circuit) $|\rho|_l$ is equal to the sum of the lengths of the arcs of which it is composed.
A *weight* of a path (or circuits) $|\rho|_w$ is the sum of the weights of the individual arcs.

Definition 2.27 connected and strongly connected digraph
A digraph is *connected* if its underlying graph is a connected graph (i.e. there exists a path between each pair of vertices), and is *disconnected* otherwise.
A digraph is *strongly connected* if there is a path between each pair of vertices.

Fig. 1 shows a connected but not strongly connected digraph.

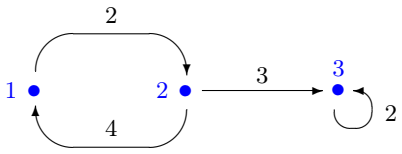

Figure 1: A (not strongly) connected digraph.

Definition 2.28 Precedence graph
The *precedence graph* $\mathcal{G}(\mathbf{A})$ of matrix $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$ is a weighted digraph with n nodes and arc (j, i) if and only if $(\mathbf{A})_{ij} \neq \varepsilon$.

Theorem 2.29
Let $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$ and let $\mathbf{B}$ be a matrix

$$\mathbf{B} = \mathbf{A} \oplus \mathbf{A}^2 \oplus \cdots \oplus \mathbf{A}^n, \tag{14}$$

then $\mathcal{G}(\mathbf{A})$ is strongly connected if and only if each entry in $\mathbf{B}$ is greater than ε.

The proof foolows straight from the standard version, we refer the reader to e.g. [Aldous and Wilson 2000].

Definition 2.30 Maximum cycle mean
The *maximum cycle mean* of precedence graph $\mathcal{G}(\mathbf{A})$ is the maximum of the average weight of circuits

$$\lambda = \max_{\rho} \frac{|\rho|_w}{|\rho|_l}, \tag{15}$$

where:
$|\rho|_l$ is the length and $|\rho|_w$ is the weight of the circuit.

It can be rewritten in (max, +) notation:

$$\lambda = \bigoplus_{j=1}^{n} tr(\mathbf{A}^j)^{\frac{1}{j}}, \tag{16}$$

where:
$(\mathbf{A}^j)_{ii}$ — the maximum weight of all circuits of length j which pass through node i,
$tr(\mathbf{A}^j)$ — maximum over all nodes.

a)

$$\mathbf{A} = \begin{bmatrix} 5 & \varepsilon & 5 \\ \varepsilon & 6 & 3 \\ 11 & 12 & 11 \end{bmatrix}$$

b)

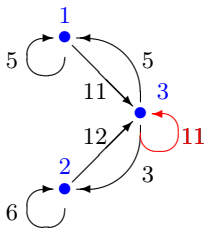

Figure 2: An exemplary matrix (a) and its precedence graph (b) with maximum cycle mean equals 11.

Theorem 2.31
[Karp 1978] a bit modified by [Gaubert and Scilab 1998].
If precedence graph of $\mathbf{A}$ is strongly connected (or if $\mathbf{A}$ irreducible), then

$$\lambda = \max_{\substack{1\leqslant j\leqslant n \\ (\mathbf{A}^n)_{ij}\neq -\infty}} \min_{1\leqslant k\leqslant n} \frac{(\mathbf{A}^n)_{ij} - (\mathbf{A}^{n-k})_{ij}}{k}. \tag{17}$$

So, in the (max, +) notation

$$\lambda = \bigoplus_{\substack{1\leqslant j\leqslant n \\ (\mathbf{A}^n)_{ij}\neq \varepsilon}} \bigvee_{1\leqslant k\leqslant n} \left(\frac{(\mathbf{A}^n)_{ij}}{(\mathbf{A}^{n-k})_{ij}} \right)^{\frac{1}{k}}. \tag{18}$$

where:

$a \vee b \equiv \min(a, b)$.

A good bibliography on the maximal cycle mean problem, and a comparison of Karp's algorithm with other classical algorithms, can be found in [Dasdan and Gupta 1998].

Table 4: Graph functions.

function	short description
`mp_is_pga(A)`	checks, if a precedence graph $\mathcal{G}(\mathbf{A})$ is acyclic
`mp_is_pgc(A)`	checks, if a precedence graph $\mathcal{G}(\mathbf{A})$ is connected
`mp_is_pgsc1(A)`	checks, if a precedence graph $\mathcal{G}(\mathbf{A})$ is strongly connected (from definition 2.27)
`mp_is_pgsc2(A)`	checks, if a precedence graph $\mathcal{G}(\mathbf{A})$ is strongly connected (from theorem 2.29)
`mp_mcm(A)`	maximum cycle mean of $\mathcal{G}(\mathbf{A})$ (from (16))
`mp_mcm_fw(A)`	maximum cycle mean of $\mathcal{G}(\mathbf{A})$ Floyd–Warshall algorithm [Floyd 1962]
`mp_mcp_karp(A)`	maximum cycle mean of $\mathcal{G}(\mathbf{A})$ Karp algorithm: 2.31

2.4 Linear equations

Some of the examples in this chapter are taken from [Gaubert and Scilab 1998].

2.4.1 Problem Ax = b

(max, +) algebra is equipped with the following natural order relation:

$$\forall a, b \in \mathbb{R}_\varepsilon : a \preccurlyeq b \iff a \oplus b = b. \tag{19}$$

Let us consider equation (20). Let $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}_{\varepsilon}^{n}$:

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{20}$$

Generally (20) has no solution, but (21) always does.

$$\mathbf{A}\mathbf{x} \preccurlyeq \mathbf{b}. \tag{21}$$

We can try solve it via residuation [Blyth and Janowitz 1972]. Precisely, this method solves (21).

Theorem 2.32 [Baccelli et al. 1992]
Given $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}_{\varepsilon}^{n}$, the greatest subsolution of (20) exists, and is given by

$$(\mathbf{i} \oslash \mathbf{x})^T = (\mathbf{i} \oslash \mathbf{b})^T \mathbf{A}, \tag{22}$$

where $\mathbf{i}$ is (max, +)–algebraic identity vector with appropriate size.

Let solution of (22) denote as

$$\mathbf{x} = (\mathbf{A} \backslash \mathbf{b}). \tag{23}$$

Equation (20) has a solution if and only if

$$\mathbf{A}(\mathbf{A} \backslash \mathbf{b}) = \mathbf{b}. \tag{24}$$

Table 5: Toolbox functions assigned to the problem $\mathbf{A}\mathbf{x} = \mathbf{b}$.

function	short description
`mp_solve_Axb(A,b)`	the greatest subsolution of $\mathbf{A}\mathbf{x} = \mathbf{b}$

2.4.2 Problem $\mathbf{x} = \mathbf{A}\mathbf{x} \oplus \mathbf{b}$

Theorem 2.33 [Baccelli et al. 1992]
Let $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$, and $\mathbf{b} \in \mathbb{R}_{\varepsilon}^{n}$. The minimal column vector $\mathbf{x} \in \mathbb{R}_{\varepsilon}^{n}$, such that

$$\mathbf{x} = \mathbf{A}\mathbf{x} \oplus \mathbf{b} \tag{25}$$

is given by

$$\mathbf{x} = \mathbf{A}^{\star}\mathbf{b}. \tag{26}$$

Definition 2.34 Star operator
The operator * for square matrix $\mathbf{A} \in \mathbb{R}_\varepsilon^{n\times n}$ is defined by:

$$\mathbf{A}^\star = \bigoplus_{k\in\mathbb{N}_0} \mathbf{A}^k \tag{27}$$

where:

$$\mathbf{A}^0 = \mathbf{I}_n,\ \mathbf{A}^k = \mathbf{A} \otimes \mathbf{A}^{k-1},$$

$\mathbb{N}_0$ is the set of nonnegative integers.

We can interpret $(\mathbf{A}^\star)_{ij}$ as the maximal weight of a path from i to j of any lenght, in the $\mathcal{G}(\mathbf{A})$. $\mathbf{A}^\star$ is a priori defined in $(\mathbb{R}_\varepsilon \cup \{+\infty\})^{n\times n}$, but the $+\infty$ value is undesired in most application.

Theorem 2.35 [Gaubert 1997]
Let $\mathbf{A} \in \mathbb{R}_\varepsilon^{n\times n}$, iff there are no circuits with positive weight in precedence graph $\mathcal{G}(\mathbf{A})$, then

$$\mathbf{A}^\star = \mathbf{A}^0 \oplus \mathbf{A}^1 \oplus \cdots \oplus \mathbf{A}^{n-1}. \tag{28}$$

Table 6: Toolbox functions assigned to the problem $\mathbf{x} = \mathbf{A}\mathbf{x} \oplus \mathbf{b}$.

function	short description	
`mp_star(A)`	(max, +) star operator	$\mathbf{A}^\star$
`mp_solve_xAxb(A,b)`	the solution of $\mathbf{x} = \mathbf{A}\mathbf{x} \oplus \mathbf{b}$	$\mathbf{x} = \mathbf{A}^\star\mathbf{b}$

2.4.3 Spectral problem $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

The most useful (max, +) practicable results are related to the spectral problem (29).

Definition 2.36 The (max, +) spectral problem
The matrix $\mathbf{A} \in \mathbb{R}_\varepsilon^{n\times n}$ has an eigenvalue in the (max, +) algebra sense, if there exist a real number $\lambda \in \mathbb{R}$ and a vector $\mathbf{v} \in \mathbb{R}^n$ such that

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}. \tag{29}$$

The vector $\mathbf{v}$ is then called an eigenvector for the eigenvalue λ.

The theory is extremely similar to the well known Perron–Frobenius theory (see e.g. [Bapat 1998]). Every square matrix with entries in $\mathbb{R}_\varepsilon$ has at least one eigenvalue. Unlike in conventional Perron–Frobenius theory, an

irreducible matrix can have several (non proportional) eigenvectors.

(max, +) algebraic eigenproblem has been studied in many publications (see e.g. [Baccelli et al. 1992] and in particular [Braker 1993] for a power algorithm). In some publications, authors allows components of vector $\mathbf{v}$ in the eigenvalue problem to assume the value ε, with the exception of (max, +) zero vector — a trivial solution of (29).

Determine an eigenvalue

Theorem 2.37 [Baccelli et al. 1992]
If $\mathbf{A}$ is irreducible, or equivalently if $\mathcal{G}(\mathbf{A})$ is strongly connected, there exists one, and only one eigenvalue (but possibly several eigenvectors). This eigenvalue is equal to the maximum cycle mean of $\mathcal{G}(\mathbf{A})$.

If $\mathcal{G}(\mathbf{A})$ is not strongly connected then we can find the maximum cycle mean by determining the maximum cycle mean for each strong component of $\mathcal{G}(\mathbf{A})$. The strong components can be found using algorithm presented in [Tarjan 1972].

Lemma 2.38 [G.-J. Olsder, Roos, and van Egmond 1999]
A solution of eigenproblem (29) is feasible if and only if $\mathcal{G}(\mathbf{A})$ contains a circuit.

Example 2.39 Eigenvalue of not irreducible matrix

- The following example shows the uniqueness of the eigenvalue when $\mathcal{G}(\mathbf{A})$ is not connected:

$$\begin{bmatrix} 1 & \varepsilon \\ \varepsilon & 2 \end{bmatrix} \begin{bmatrix} 0 \\ \varepsilon \end{bmatrix} = 1 \begin{bmatrix} 0 \\ \varepsilon \end{bmatrix} = \begin{bmatrix} 1 \\ \varepsilon \end{bmatrix},$$

$$\begin{bmatrix} 1 & \varepsilon \\ \varepsilon & 2 \end{bmatrix} \begin{bmatrix} \varepsilon \\ 0 \end{bmatrix} = 2 \begin{bmatrix} \varepsilon \\ 0 \end{bmatrix} = \begin{bmatrix} \varepsilon \\ 2 \end{bmatrix}.$$

- In the following example $\mathcal{G}(\mathbf{A})$ is connected but not strongly connected, and there are two eigenvalues:

$$\begin{bmatrix} 0 & 0 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \varepsilon \end{bmatrix} = 0 \begin{bmatrix} 0 \\ \varepsilon \end{bmatrix} = \begin{bmatrix} 0 \\ \varepsilon \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

We can also use an another way to obtain the eigenvalue. The starting point is the equation of linear system (30), which describes the time evolution of an autonomous DES (for more details see § 4):

$$\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k-1), \tag{30}$$

where $\mathbf{x} \in \mathbb{R}_{\varepsilon}^{n}$ and $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n\times n}$.

Theorem 2.40
Given an initial vector $\mathbf{x}(0)$ (the initial state of the system), we say that the system (30), after a number of steps, ends up in a periodic behaviour, if there exists integers p, q with $p > q \geqslant 0$ and real number c such that $\mathbf{x}(p) = \mathbf{x}(q) \oplus c$. Then the eigenvalue λ is given by

$$\lambda = \frac{c}{p-q} \qquad \text{(in the conventional algebra).} \tag{31}$$

Example 2.41
Let $\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\mathbf{A} = \begin{bmatrix} 3 & 7 \\ 2 & 4 \end{bmatrix}$ then $\mathbf{x}(1) = \begin{bmatrix} 7 \\ 4 \end{bmatrix} \rightarrow \mathbf{x}(2) = \begin{bmatrix} 11 \\ 9 \end{bmatrix} \rightarrow \mathbf{x}(3) = \begin{bmatrix} 16 \\ 13 \end{bmatrix}$. So, $\mathbf{x}(3) - \mathbf{x}(1) = \begin{bmatrix} 9 \\ 9 \end{bmatrix}$. Hence $\lambda = \dfrac{9}{3-1} = 4.5$.

Looking for eigenvectors: implemented algorithms

Algorithm 1 [G.-J. Olsder 1991]
Candidate eigenvector — first so-called power algorithm.
If we have p, q and c (see theorem 2.40), then the eigenvector is given by:

$$\mathbf{v} = \frac{1}{p-q} \sum_{j=1}^{p-q} \mathbf{x}(q+j-1) \qquad \text{(in the conventional algebra).} \tag{32}$$

Note: This algorithm does not always give an eigenvector.

Example 2.42
Let $\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\mathbf{A} = \begin{bmatrix} 3 & 7 \\ 2 & 4 \end{bmatrix}$. So $p = 3, q = 1, c = 9$.
Hence $\mathbf{v} = \dfrac{1}{3-1}\big(\mathbf{x}(1) + \mathbf{x}(2)\big) = \dfrac{1}{2}\Big(\begin{bmatrix} 7 \\ 4 \end{bmatrix} + \begin{bmatrix} 11 \\ 9 \end{bmatrix}\Big) = \begin{bmatrix} 9 \\ 6.5 \end{bmatrix}$.

Algorithm 2 [Braker and G.-J. Olsder 1993]
Extended alg. 1 — power algorithm.

1. Calculate a candidate for an eigenvector from (32);
2. If $\mathbf{v}$ is an eigenvector then stop, if not, go to step 3;
3. Define a new vector:

$$\overline{v_i} = \begin{cases} v_i, & \text{if } (\mathbf{A} \otimes \mathbf{v})_i = \lambda \otimes v_i \\ \varepsilon, & \text{elsewhere} \end{cases} \tag{33}$$

4. Restart algorithm with $\mathbf{x}(0) = \overline{\mathbf{v}}$,
   until for some $r \geqslant 0$, there holds $\mathbf{x}(r+1) = \lambda \otimes \mathbf{x}(r)$,
   then $\mathbf{x}(r)$ is an eigenvector of $\mathbf{A}$.

Example 2.43

Let $\mathbf{A} = \begin{bmatrix} \varepsilon & 3 & \varepsilon & 1 \\ 2 & \varepsilon & 1 & \varepsilon \\ 1 & 2 & 2 & \varepsilon \\ \varepsilon & \varepsilon & 1 & \varepsilon \end{bmatrix}$, and $\mathbf{x}(0) = \begin{bmatrix} 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}$, so $\lambda = 2\frac{1}{2}$.

A result form alg. 2 is $\mathbf{v} = \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ 5 \\ 3\frac{1}{2} \end{bmatrix}$.

It is easy to see that $\mathbf{A} \otimes \mathbf{v} = \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 7 \\ 6 \end{bmatrix} \neq \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 7\frac{1}{2} \\ 6 \end{bmatrix} = \lambda \otimes \mathbf{v}$.

The new vector $\overline{\mathbf{v}} = \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ \varepsilon \\ 3\frac{1}{2} \end{bmatrix} = \mathbf{x}(0)$, and then

$$\mathbf{x}(0) = \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ \varepsilon \\ 3\frac{1}{2} \end{bmatrix} \to \mathbf{x}(1) = \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 6\frac{1}{2} \\ \varepsilon \end{bmatrix} \to \mathbf{x}(2) = \begin{bmatrix} 10 \\ 9\frac{1}{2} \\ 9 \\ 7\frac{1}{2} \end{bmatrix} \to \mathbf{x}(3) = \begin{bmatrix} 12\frac{1}{2} \\ 12 \\ 11\frac{1}{2} \\ 10 \end{bmatrix} = \lambda \otimes \mathbf{x}(2).$$

Hence, a result of alg. 2 is $\begin{bmatrix} 10 \\ 9\frac{1}{2} \\ 9 \\ 7\frac{1}{2} \end{bmatrix}$.

Algorithm 3 [G.-J. Olsder, Roos, and van Egmond 1999]
With adapted Floyd–Warshall procedure [Ahuja, Magnanti, and Orlin 1993].
Let $\mathcal{G}(\mathbf{A}) = (\mathcal{V}, \mathcal{E})$ be a precedence graph of $\mathbf{A}$, $\mathbf{J}$ denotes the all–one matrix.

1. We start by choosing a lower bound μ for λ.
   For this we can take $\mu = \max_{i\in\mathcal{V}}(\mathbf{A})_{ii}$ if the diagonal of $\mathbf{A}$ is not void; otherwise we may use $\mu = \min_{i,j\in\mathcal{E}}(\mathbf{A})_{ij}$, because the average weight of a circuit cannot be smaller.

2. Then we check (see below) whether there exists a circuit C with positive weight with respect to $\mathbf{A} - \mu\mathbf{J}$. If this is the case we increase μ to the average weight of C and restart the procedure. This is repeated until there are no circuits having positive weight with respect to $\mathbf{A}' = \mathbf{A} - \mu\mathbf{J}$. It implies that $\lambda = \mu$.

   The (adapted) Floyd–Warshall procedure to detect the existence of circuits having positive weight goes as follows. Starting with $\mathbf{A}_0 = \mathbf{A}$ we successively construct matrices $\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_n$ by defining

   $$(\mathbf{A}_k)_{ij} = \max\big((\mathbf{A}_{k-1})_{ij}, (\mathbf{A}_{k-1})_{ik} + (\mathbf{A}_{k-1})_{kj}\big), \qquad k = 1, 2, ..., n. \tag{34}$$

   If we apply this procedure to the matrix $\mathbf{A}' = \mathbf{A} - \mu\mathbf{J}$ and

   - if there exists no positive circuits with respect to $\mathbf{A}' = \mathbf{A} - \mu\mathbf{J}$ (i.e. when $\mu = \lambda$), then the nodes lying on a critical circuit are exactly those i for which $(\mathbf{A}')_{ii} = 0$;
   - if there exists a positive circuits with respect to $\mathbf{A}'$ (i.e., when $\mu < \lambda$), the existence of such a circuit is detected by the Floyd–Warshall procedure as soon as a diagonal entry in one of the matrices $\mathbf{A}'_k$ becomes positive. This circuit, C say, can be found by backtracking the Floyd–Warshall procedure, starting from the node corresponding to the positive diagonal entry. We replace μ by the average weight of C with respect to $\mathbf{A}$ and restart the Floyd–Warshall procedure on the updated matrix $\mathbf{A} - \mu\mathbf{J}$.

3. The eigenvectors can be immediately obtained from the final matrix $\mathbf{A}'_n$ generated by the Floyd–Warshall procedure. This matrix reveals all critical circuits and all nodes lying on critical circuits; these are precisely the nodes whose columns in $\mathbf{A}'_n$ have a zero on the diagonal. We denote the critical circuits by C_k where k runs through an index set K whose cardinality is equal to the number of critical circuits. For each $k \in K$ we choose a node r_k on C_k. We call r_k the reference node for the critical circuit C_k. Moreover, for each node i we define

   $$x_i^{(k)} = \text{maximum weight of all paths in } \mathcal{G}(\mathbf{A}') \text{ from } r_k \text{ to } i,\ i \in \mathcal{V}.$$

   If there exists no path from r_k to i then the corresponding entry of $x^{(k)}$ will be taken ε. Note that $x^{(k)}$ is just the column in the matrix $\mathbf{A}'_n$ corresponding to the node r_k.

Now let L be any nonempty subset of K. For any such subset we define

$$x_i^L = \max\{x_i^{(k)} : \ k \in L\}, i \in \mathcal{V}. \tag{35}$$

If L is such that the vector $\mathbf{x}^L$ is finite then $\mathbf{x}^L$ is a solution of the eigenvalue problem (29).

Example 2.44
Consider the following matrix:

$$\mathbf{A} = \begin{bmatrix} 4 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 4 \\ 2 & 5 & 1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 6 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 8 & 3 & \varepsilon & \varepsilon \\ 9 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 8 & 3 \end{bmatrix}.$$

As a first guess for the maximum cycle mean of $\mathcal{G}(\mathbf{A})$ we take the maximum entry on the diagonal of $\mathbf{A}$, thus $\mu = (\mathbf{A})_{22} = 5$. Next we apply the Floyd–Warshall procedure to the matrix $\mathbf{A} - 5\mathbf{J}$. After three iterations we find a positive entry on the diagonal:

$$(\mathbf{A} - 5\mathbf{J})_3 = \begin{bmatrix} -1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & -1 \\ -3 & 0 & -4 & -3 & \varepsilon & -4 \\ \varepsilon & \varepsilon & \varepsilon & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 3 & 4 & \varepsilon & \varepsilon \\ 4 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 3 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 3 & -2 \end{bmatrix}.$$

There is a circuit containing node 4 of weight 4. Indeed we have a circuit $4 \to 3 \to 4$ which has circuit mean 7 with respect to $\mathbf{A}$. We therefore proceed with $\mu = 7$ and repeat the Floyd–Warshall procedure to the matrix $\mathbf{A} - 7\mathbf{J}$. This time all diagonal entries remain nonpositive:

$$(\mathbf{A} - 7\mathbf{J})_6 = \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon & -2 & -3 \\ -5 & -2 & -6 & -7 & -7 & -8 \\ \varepsilon & \varepsilon & 0 & -1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 1 & 0 & \varepsilon & \varepsilon \\ 2 & \varepsilon & \varepsilon & \varepsilon & 0 & -1 \\ 3 & \varepsilon & \varepsilon & \varepsilon & 1 & 0 \end{bmatrix}.$$

The zero diagonal entries of $(\mathbf{A} - 7\mathbf{J})_6$ give us the critical nodes.
It turns out that $\mathcal{G}(\mathbf{A})$ has two critical circuits: $4 \to 3 \to 4$ and $1 \to 5 \to 6 \to$

1 with $\mu = 7$. Application of algorithm yields the following six eigenvectors ($\mathbf{x}^{\{6,4\}}$ is a linear combination of $\mathbf{x}^{\{5,3\}}$):

$$\mathbf{x}^{\{1,3\}} = \begin{bmatrix} 0 \\ -5 \\ 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{x}^{\{5,3\}} = \begin{bmatrix} -2 \\ -6 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{x}^{\{6,3\}} = \begin{bmatrix} -3 \\ -6 \\ 0 \\ 1 \\ -1 \\ 0 \end{bmatrix},$$

$$\mathbf{x}^{\{1,4\}} = \begin{bmatrix} 0 \\ -5 \\ -1 \\ 0 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{x}^{\{5,4\}} = \begin{bmatrix} -2 \\ -7 \\ -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{x}^{\{6,4\}} = \begin{bmatrix} -3 \\ -7 \\ -1 \\ 0 \\ -1 \\ 0 \end{bmatrix}.$$

Algorithm 4 [Subiono and van der Woude 2000]
Extended alg. 1 and 2.

1. Calculate a candidate for an eigenvector from (32);
2. If $\mathbf{v}$ is an eigenvector then stop, if not, go to step 3;
3. Restart algorithm with $\mathbf{x}(0) = \mathbf{v}$,
   until for some $r \geqslant 0$, there holds $\mathbf{x}(r+1) = \lambda \otimes \mathbf{x}(r)$,
   then $\mathbf{x}(r)$ is an eigenvector of $\mathbf{A}$.

Example 2.45

Let $\mathbf{A} = \begin{bmatrix} \varepsilon & 3 & \varepsilon & 1 \\ 2 & \varepsilon & 1 & \varepsilon \\ 1 & 2 & 2 & \varepsilon \\ \varepsilon & \varepsilon & 1 & \varepsilon \end{bmatrix}$, and an initial state $\mathbf{x}(0) = \begin{bmatrix} 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}$, so $\lambda = 2\frac{1}{2}$.

A result form alg. 1 is $\mathbf{v} = \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ 5 \\ 3\frac{1}{2} \end{bmatrix}$.

We can see that $\mathbf{A} \otimes \mathbf{v} = \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 7 \\ 6 \end{bmatrix} \neq \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 7\frac{1}{2} \\ 6 \end{bmatrix} = \lambda \otimes \mathbf{v}$.

So, the new vector $\mathbf{x}(0) = \mathbf{v}$, and:

$$\mathbf{x}(0) = \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ 5 \\ 3\frac{1}{2} \end{bmatrix} \rightarrow \mathbf{x}(1) = \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 7 \\ 6 \end{bmatrix} \rightarrow \mathbf{x}(2) = \begin{bmatrix} 10 \\ 9\frac{1}{2} \\ 9 \\ 8 \end{bmatrix} \rightarrow$$

$$\rightarrow \mathbf{x}(3) = \begin{bmatrix} 12\frac{1}{2} \\ 12 \\ 11\frac{1}{2} \\ 10 \end{bmatrix} \rightarrow \mathbf{x}(4) = \begin{bmatrix} 15 \\ 14\frac{1}{2} \\ 14 \\ 12\frac{1}{2} \end{bmatrix} = \lambda \otimes \mathbf{x}(3).$$

Hence, a result of alg. 4 is $\begin{bmatrix} 12\frac{1}{2} \\ 12 \\ 11\frac{1}{2} \\ 10 \end{bmatrix}$.

Algorithm 5 [Subiono and van der Woude 2000]
If we have p, q and c (see theorem 2.40), then the eigenvector is given by:

$$\mathbf{v} = \bigoplus_{j=1}^{p-q} (\lambda^{p-q-j} \otimes \mathbf{x}(q+j-1)). \tag{36}$$

Example 2.46
Let $\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\mathbf{A} = \begin{bmatrix} 3 & 7 \\ 2 & 4 \end{bmatrix}$. So $p = 3, q = 1, c = 9$ and $\lambda = 4.5$.
Hence $\mathbf{v} = \lambda^1 \mathbf{x}(1) \oplus \lambda^0 \mathbf{x}(2) = 4.5 \begin{bmatrix} 7 \\ 4 \end{bmatrix} \oplus 1 \begin{bmatrix} 11 \\ 9 \end{bmatrix} = \begin{bmatrix} 11.5 \\ 9 \end{bmatrix}$.

Table 7: Toolbox functions assign to the spectral problem $\mathbf{Av} = \lambda\mathbf{v}$.

function	short description
mp_is_pga	checks if A has got any eigenvalue
mp_is_pgsc1	checks if A has got exactly one eigenvalue
mp_is_pgsc2	checks if A has got exactly one eigenvalue
mp_is_egv1	checks if $\mathbf{v}$ is an eigenvector of $\mathbf{A}$
mp_is_egv2	checks if $\mathbf{v}$ is an eigenvector of $\mathbf{A}$
mp_pqc	calculate components of an eigenvalue of $\mathbf{A}$, from (31)
mp_egv_pqc	eigenvalue of $\mathbf{A}$, from (31)
mp_mcm	eigenvalue of $\mathbf{A}$, from (16)
mp_mcm_fw	eigenvalue of $\mathbf{A}$ - Floyd–Warshall algorithm
mp_mx_fw	final Floyd–Warshall matrix of $\mathbf{A}$
mp_ev_fw	eigenvectors of $\mathbf{A}$ - Floyd–Warshall algorithm
mp_mcm_karp	eigenvalue of $\mathbf{A}$ - Karp algorithm
mp_egv_o91	eigenvalue and eigenvector of $\mathbf{A}$, [G.-J. Olsder 1991]
mp_egv_bo93	eigenvalue and eigenvector of $\mathbf{A}$, [Braker and G.-J. Olsder 1993]
mp_egv_sw001	eigenvalue and eigenvector of $\mathbf{A}$, [Subiono and van der Woude 2000]
mp_egv_sw002	eigenvalue and eigenvector of $\mathbf{A}$, [Subiono and van der Woude 2000]

3 A bit of the (min, +) algebra

Definition 3.1 The (min, +) algebra
(min, +) algebra is defined as follows:

- $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{+\infty\}$, where $\mathbb{R}$ is the field of real numbers;
- $\forall a, b \in \mathbb{R}_\varepsilon : \quad a \oplus b \equiv \min(a, b)$;
- $\forall a, b \in \mathbb{R}_\varepsilon : \quad a \otimes b \equiv a + b$.

The algebraic structure $\mathbb{R}_{\min} = (\mathbb{R}_\varepsilon, \oplus, \otimes)$, is called *the min-plus algebra*. The reason for not distinguishing between the maximum in (max, +) and the minimum in (min, +) operations is that $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$ are isomorphic algebraic structures and such notations are frequently used in literature. But in this paper to avoid any ambiguity we will use different symbols:

- $a \vee b \equiv \min(a, b)$;
- $a \wedge b \equiv a + b$.

We decide to use different symbol for + in (max, +) and in (min, +), because

$$-\infty \otimes \infty = -\infty \neq -\infty \wedge \infty = \infty. \tag{37}$$

The (max, +) and (min, +) algebra together comprise the *minimax algebra* [Cuninghame–Green 1979].

The basic scalar, vector and matrix operations in the (min, +) are formed analogously with the (max, +), and they are ommited here.

Definition 3.2 Plus operator
The operator $^+$ for square matrix $\mathbf{A} \in \mathbb{R}_\varepsilon^{n \times n}$ is defined by:

$$\mathbf{A}^+ = \bigvee_{k \in \mathbb{N}} \mathbf{A}^k, \tag{38}$$

where $\mathbb{N}$ is the set of natural numbers.

It does mean, that

$$\mathbf{A}^+ \vee \mathbf{A}^0 = \mathbf{A}^\star. \tag{39}$$

The matrix $\mathbf{A}^+$ sometimes referred as the *shortest path matrix*.

Example 3.3 Shortest path problem
Let us consider a shortest path problem for digraph shown on Fig. 3.

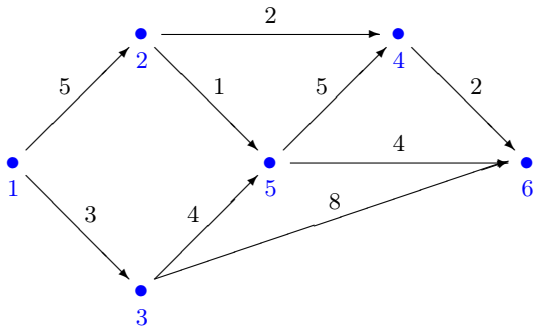

Figure 3: An exemplary digraph.

$$\mathbf{A} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 5 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 2 & \varepsilon & \varepsilon & 5 & \varepsilon \\ \varepsilon & 1 & 4 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 8 & 2 & 4 & \varepsilon \end{bmatrix},$$

so,

$$\mathbf{A}^{+} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 5 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 7 & 2 & 9 & \varepsilon & 5 & \varepsilon \\ 6 & 1 & 4 & \varepsilon & \varepsilon & \varepsilon \\ 9 & 4 & 8 & 2 & 4 & \varepsilon \end{bmatrix}.$$

Hence, the shortest path e.g. from node 1 to node 6 is 9 — see $(\mathbf{A}^{+})_{6,1}$.

Table 8: (min, +) functions.

function	short description	
`mpm_zero`	neutral element for opertion $\vee$	
`mpm_zeros`	zeros matrix, vector or scalar	
`mpm_one`	neutral element for opertion $\wedge$	
`mpm_ones`	ones matrix (vector / scalar)	
`mpm_eye`	identity matrix	$\mathbf{I}_n$
`mpm_add`	addition	$a \vee b$, $\mathbf{A} \vee \mathbf{B}$
`mpm_multi`	multiplication	$a \wedge b$, $\mathbf{A} \wedge \mathbf{B}$
`mpm_plus(A)`	plus operator of $\mathbf{A}$	$\mathbf{A}^+ = \mathbf{A}^1 \vee \mathbf{A}^2 \vee \ldots$
`mpm_star(A)`	star operator of $\mathbf{A}$	$\mathbf{A}^\star = \mathbf{A}^0 \vee \mathbf{A}^+$
`mpm_inv(A)`	invertion of $\mathbf{A}$	$\mathbf{A}^{-1}$
`mpm_div`	division	a/b, $\mathbf{B}^{-1}\mathbf{A}$
`mpm_power`	n–th power	a^n, $\mathbf{A}^n$

4 State space description of DES

4.1 Introduction

Probably the most well–known equation in the theory of difference equations is

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t-1), \qquad t = 1, 2, \ldots \tag{40}$$

The vector $\mathbf{x} \in \mathbb{R}^n$ represents the *state* of the underlying model and this state evolves in time according to this equation. If an initial condition

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{41}$$

is given, then the whole future evolution of (40) is determined. Equation (40) rewritten in (max, +) notation ($\mathbf{x} \in \mathbb{R}_\varepsilon^n$, $\mathbf{A} \in \mathbb{R}_\varepsilon^{n\times n}$, $k \in \mathbb{N}_0$) is given by:

$$\mathbf{x}(k) = \mathbf{A} \otimes \mathbf{x}(k-1). \tag{42}$$

Where k is a cycle index.

Example 4.1 An introductory example to state space description
As an example, we take matrix $\mathbf{A} \in \mathbb{R}_\varepsilon^{2\times 2}$. Let $\mathbf{A} = \begin{bmatrix} 3 & 7 \\ 2 & 4 \end{bmatrix}$ and the initial condition $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. The time evolution of (65) becomes:

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \to \mathbf{x}(1) = \begin{bmatrix} 7 \\ 4 \end{bmatrix} \to \mathbf{x}(2) = \begin{bmatrix} 11 \\ 9 \end{bmatrix} \to \mathbf{x}(3) = \begin{bmatrix} 16 \\ 13 \end{bmatrix} \ldots$$

State space 1–order model with dependence on inputs and outputs is an extension of (65):

$$\begin{aligned} &\forall k \in \mathbb{N}: \\ &\quad \mathbf{x}(k) = \mathbf{A}(k)\mathbf{x}(k-1) \oplus \mathbf{B}(k)\mathbf{u}(k), &(43)\\ &\quad \mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) \oplus \mathbf{D}(k)\mathbf{u}(k). &(44) \end{aligned}$$

$\mathbf{x} \in \mathbb{R}_\varepsilon^n$ is called *state vector*,
$\mathbf{u} \in \mathbb{R}_\varepsilon^r$ is called *input vector* or *control vector* of the system,
$\mathbf{y} \in \mathbb{R}_\varepsilon^m$ is called *output vector* of the system,
$\mathbf{A} \in \mathbb{R}_\varepsilon^{n\times n}$ is called *system matrix*,
$\mathbf{B} \in \mathbb{R}_\varepsilon^{n\times r}$ is called *input matrix*,
$\mathbf{C} \in \mathbb{R}_\varepsilon^{m\times n}$ is called *output matrix*,

$\mathbf{D} \in \mathbb{R}_{\varepsilon}^{m\times r}$ is called *feedthrough (or feedforward) matrix.* In general case, for N–order time–invariant system is described by:

$$\mathbf{x}(k) = \bigoplus_{i=0}^{N} \mathbf{A}_i\mathbf{x}(k-i) \oplus \bigoplus_{i=0}^{N-1} \mathbf{B}_i\mathbf{u}(k-i), \tag{45}$$

$$\mathbf{y}(k) = \bigoplus_{i=0}^{N-1} \Big(\mathbf{C}_i\mathbf{x}(k-i) \oplus \mathbf{D}_i\mathbf{u}(k-i)\Big). \tag{46}$$

When we remove the $\mathbf{x}(k)$ on the right–side of (45) (if $\mathbf{A}_0^\star$ converge), and if we also define a new state vector, an augmented input vector and matrices:

$$\widetilde{\mathbf{x}}(k) = \big[\mathbf{x}(k)\mathbf{x}(k-1)\cdots\mathbf{x}(k-N+1)\big]^T, \tag{47}$$

$$\widetilde{\mathbf{u}}(k) = \big[\mathbf{u}(k)\mathbf{u}(k-1)\cdots\mathbf{u}(k-N+1)\big]^T, \tag{48}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0^\star\mathbf{A}_1 & \mathbf{A}_0^\star\mathbf{A}_2 & \dots & \dots & \mathbf{A}_0^\star\mathbf{A}_N \\ \mathbf{I} & \boldsymbol{\varepsilon} & \dots & \dots & \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} & & & & \\ \vdots & & \ddots & & \\ \boldsymbol{\varepsilon} & \dots & \boldsymbol{\varepsilon} & \mathbf{I} & \boldsymbol{\varepsilon} \end{bmatrix}, \tag{49}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{A}_0^\star\mathbf{B}_0 & \dots & \mathbf{A}_0^\star\mathbf{B}_{N-1} \\ \boldsymbol{\varepsilon} & \dots & \boldsymbol{\varepsilon} \\ \vdots & & \vdots \\ \boldsymbol{\varepsilon} & \dots & \boldsymbol{\varepsilon} \end{bmatrix}, \tag{50}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_0 & \dots \mathbf{C}_{N-1} \end{bmatrix}, \tag{51}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_0 & \dots \mathbf{D}_{N-1} \end{bmatrix}, \tag{52}$$

then eqns. (45), (46) can be written as time–invariant, 1–order model of eqns. (43), (44) where $\mathbf{I}$ and $\boldsymbol{\varepsilon}$ are appropriately sized $(\max,+)$-algebraic identity and zero matrices, respectively, and

$$\forall k \in \mathbb{N} : \mathbf{A}(k) = \mathbf{A},\ \mathbf{B}(k) = \mathbf{B},\ \mathbf{C}(k) = \mathbf{C},\ \mathbf{D}(k) = \mathbf{D}.$$

Table 9: Toolbox functions for state space models.

function	short description
`mp_system`	state fo an autonomous linear max–plus system

4.2 State space description of timed event graph

Timed event graphs (TEGs) are (timed) Petri-nets [Murata 1989], which are convenient to model (timed) synchronisation problems. They are characterised by the fact that every place has exactly one predecessor transition and one successor transition. Time constraints are modelled by so–called *holding times*, representing the minimum amount of time a token has to *spend* in a place before it can contribute to enable a *downstream transition*. Let $x_i(k)$ denote the time instant, when an *internal transition* i can fire for the k^{th} time, and $\mathbf{x}(k) = (x_i(k))$ the corresponding vector of firing times. Similarly, let $u_i(k)$ denote the firing times of *input transitions* which can be triggered by the outside world, and $y_i(k)$ the firing times of *output transitions* which carry information to the outside world. It is then straightforward, to read the (max, +) equations (45)–(46) from the timed event graph.

Example 4.2 An example of a TEG and its (max, +) description
Let us consider an exemplary timed event graph depicted on Fig. 4 and its state space representaion in the (max, +) domain.

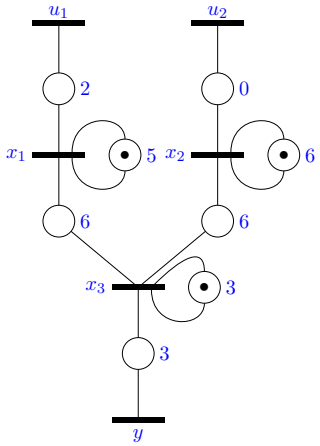

Figure 4: An exemplary TEG.

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{A}_0\mathbf{x}(k) \oplus \mathbf{A}_1\mathbf{x}(k-1) \oplus \mathbf{B}_0\mathbf{u}(k) \\ &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k), \end{aligned}$$

where:

$$\mathbf{A}_0 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ 6 & 6 & \varepsilon \end{bmatrix}, \mathbf{A}_1 = \begin{bmatrix} 5 & \varepsilon & \varepsilon \\ \varepsilon & 6 & \varepsilon \\ \varepsilon & \varepsilon & 3 \end{bmatrix}, \mathbf{B}_0 = \begin{bmatrix} 2 \\ 0 \\ \varepsilon \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \varepsilon & \varepsilon & 3 \end{bmatrix},$$

$$\mathbf{A} = \mathbf{A}_0^{\star}\mathbf{A}_1 = \begin{bmatrix} 5 & \varepsilon & \varepsilon \\ \varepsilon & 6 & \varepsilon \\ 11 & 12 & 3 \end{bmatrix}, \mathbf{B} = \mathbf{A}_0^{\star}\mathbf{B}_0 = \begin{bmatrix} 2 \\ 0 \\ 8 \end{bmatrix}.$$

The **Petri Net Toolbox**[2] is able to directly derive the (max, +) state space representation in form (45)–(46) from topology of a TEG.

4.3 Analysis of DES

The periodical behaviour of closed DESs, i.e. involving a set of repeatedly performed activities, can be totally characterised by solving an eigenvalue and eigenvector equation. Additionally, in the toolbox there is a set of functions for graphical illustration of DESs behaviour.

4.3.1 Graphical representation – Gantt charts

For generating Gantt charts[3] we defined two functions — see tab. 10

Table 10: Gantt charts.

function	short description
`mp_ganttr`	Gantt chart of resources occupation in time
`mp_ganttx`	Gantt chart of a state vector evolution in time

The Gantt charts can be saved as encapsulated postscript using the following expression:

```
>> print -depsc2 'fileName.eps'
```

or as a PDF:

```
>> print -dpdf 'fileName.pdf'
```

4.4 Examples of DES

In this part, we focus on two examples from domain (flexible) manufacturing.

4.4.1 A simple production system

Let consider a simple production system, presented in Fig. 6 [De Schutter 1996]. This production system consists of 3 processing units: M_1, M_2 and M_3. A raw material is fed to M_1 and M_2, where it is processed and sent to

[2] More information under http://www.ac.tuiasi.ro/pntool.

[3] At present, graphical functions work under Matlab® only.

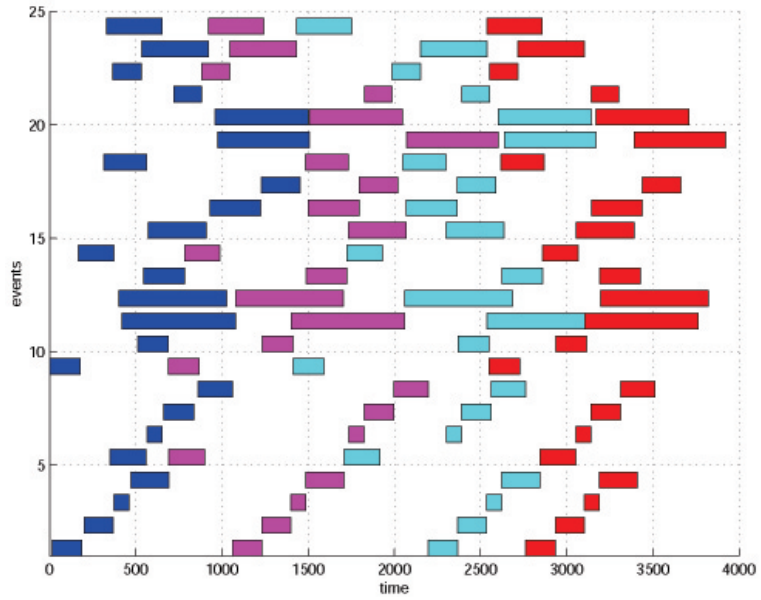

Figure 5: An exemplary Gantt chart generated by function `mp_ganttx`.

M_3, where assembly takes place. The processing times for M_1, M_2 and M_3 are respectively $d_1 = 5$, $d_2 = 6$ and $d_3 = 3$ time units. We assume that it takes $t_1 = 2$ time units for the raw material to get from the input source to M_1 and that it takes $t_3 = 1$ time unit for the finished products of processing unit M_1 to reach M_3. The other transportation times (t_2, t_4 and t_5) are assumed to be negligible. At the input of the system and between the processing units there are buffers with a capacity that is large enough to ensure that no over flow will occur. Initially all the buffers are empty and none of the processing units contains raw material or intermediate products. A processing unit can start working on a new product only after it has finished processing the previous one. We assume that each processing unit starts working as soon as all parts are available.

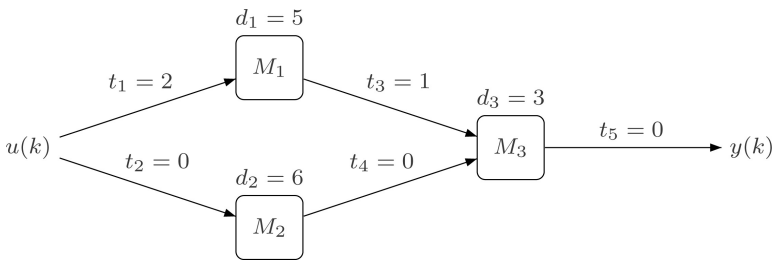

Figure 6: A simple production system.

We define:

- $u(k)$ — time instant at which raw material is feed to the system for the kth time,
- $x_i(k)$ — time instant at which the ith processing unit starts working for the kth time,
- $y(k)$ — time instant at which the kth finished product leaves the system.

Let us now determine the time instant at which processing unit M_1 starts working for the kth time. If we feed raw material to the system for the kth time, then this raw material is available at the input of processing unit M_1 at time $t = u(k) + 2$. However, M_1 can start working on the new batch of raw material only as soon as it has finished processing the previous, i.e. the $(k-1)$st batch. Since the processing time on M_1 is $d_1 = 5$ time units, the $(k-1)$st intermediate product will leave M_1 at time $t = x_1(k-1)+5$. Since M_1 starts working on a batch of raw material as soon as the raw material is available and the current batch has left the processing unit, this implies that we have

$$\forall k \in \mathbb{N}_0 : \qquad x_1(k) = \max\big(x_1(k-1) + 5, u(k) + 2\big). \tag{53}$$

The condition that initially processing unit M_1 is empty and idle corresponds to the initial condition $x_1(0) = \varepsilon$ and hence it follows from equation (53) that $x_1(1) = u(1) + 2$, i.e. the first batch of raw material that is fed to the system will be processed immediately (after a delay of 2 time units needed to transport the raw material from the input to M_1).
Using a similar reasoning we find the following expressions for the time instants at which M_2 and M_3 start working for the kth time and for the time instant at which the kth finished product leaves the system:

$$\begin{aligned}
\forall k \in \mathbb{N}_0 : \\
x_2(k) &= \max\big(x_2(k-1) + 6, u(k) + 0\big), \\
x_3(k) &= \max\big(x_1(k) + 5 + 1, x_2(k) + 6 + 0, x_3(k-1) + 3\big) \\
&= \max\big(x_1(k-1) + 11, x_2(k-1) + 12, x_3(k-1) + 3, u(k) + 8\big), \\
y(k) &= x_3(k) + 3 + 0.
\end{aligned} \tag{54}$$

The condition that initially all buffers are empty corresponds to the initial condition

$$x_1(0) = x_2(0) = x_3(0) = \varepsilon. \tag{55}$$

Let us now rewrite the evolution equations of the production system using the (max, +) notation:

$$\begin{aligned} x_1(k) &= 5x_1(k-1) \oplus 2u(k), \\ x_2(k) &= 6x_2(k-1) \oplus u(k), \\ x_3(k) &= 11x_1(k-1) \oplus 12x_2(k-1) \oplus 3x_3(k-1) \oplus 8u(k), \\ y(k) &= 3x_3(k), \end{aligned} \tag{56}$$

in matrix notation:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{u}(k) \\ &= \begin{bmatrix} 5 & \varepsilon & \varepsilon \\ \varepsilon & 6 & \varepsilon \\ 11 & 12 & 3 \end{bmatrix} \mathbf{x}(k-1) \oplus \begin{bmatrix} 2 \\ 0 \\ 8 \end{bmatrix} \mathbf{u}(k), \end{aligned} \tag{57}$$

$$\begin{aligned} \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) \\ &= \begin{bmatrix} \varepsilon & \varepsilon & 3 \end{bmatrix} \mathbf{x}(k), \end{aligned}$$

where $\mathbf{x}(k) = \begin{bmatrix} x_1(k) & x_2(k) & x_3(k) \end{bmatrix}^T$.

Now, we assume that after a machine has finished a sequence of products it starts with the next sequence. So, we have $\mathbf{u}(k) = \mathbf{y}(k-1)$ for $k > 0$. Hence, we obtain:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{u}(k) \\ &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{y}(k-1) \\ &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{C}\mathbf{x}(k-1) \\ &= \big(\mathbf{A} \oplus \mathbf{B}\mathbf{C}\big)\mathbf{x}(k-1) \\ &= \widehat{\mathbf{A}}\mathbf{x}(k-1) \end{aligned} \tag{58}$$

where $\widehat{\mathbf{A}} = \mathbf{A} \oplus \mathbf{B}\mathbf{C} = \begin{bmatrix} 5 & \varepsilon & 5 \\ \varepsilon & 6 & 3 \\ 11 & 12 & 11 \end{bmatrix}$.

We can see, that this cyclic production system can also be described by a model of the form (43), but now the model is autonomous i.e. there is no external input that controls the behaviour of the system. This example is included in file: `exSimpleProduction.m`.
Exemplary results for model (57) with $\mathbf{u}(k) = 0 = constant$:

```
% definition of matrices
>> A = [      5 mp_zero mp_zero
        mp_zero       6 mp_zero
             11      12       3];
>> B = [      2       0       8]';
>> C = [mp_zero mp_zero       3];
```

```
7
8  % determine initial conditions
9  >> x0 = mp_zeros(3, 1);
10 >> u0 = 0;
11
12 % calculate a sequence of state vector
13 >> X(:, 1)= mp_add(mp_multi(A, x0), mp_multi(B, u0));
14 >> Y(1)  = mp_multi(C, X(:, 1));
15 >> for i = 2:10
16       X(:, i) = mp_add(mp_multi(A, X(:, i-1)), mp_multi(B, u0));
17       Y(i) = mp_multi(C, X(:, i));
18    end
19 >> X, Y
20 X =
21      2     7    12    17    22    27    32    37    42    47
22      0     6    12    18    24    30    36    42    48    54
23      8    13    18    24    30    36    42    48    54    60
24 Y =
25     11    16    21    27    33    39    45    51    57    63
```

`X` is a sequence of state vector from $\mathbf{x}(1)$ to $\mathbf{x}(10)$, `Y` is an output value for appropriate state vector.
Results for model (57) with $\mathbf{u}(0) = 0$ and then $\mathbf{u}(k) = \mathbf{y}(k-1)$:

```
26 % calculate a sequence of state vector
27 >> X(:, 1)= mp_add(mp_multi(A, x0), mp_multi(B, u0));
28 >> Y(1)  = mp_multi(C, X(:, 1));
29 >> for i = 2:10
30       X(:, i) = mp_add(mp_multi(A, X(:,i-1)), mp_multi(B, Y(i-1)));
31       Y(i) = mp_multi(C, X(:, i));
32    end
33 >> X, Y
34 X =
35      2    13    24    35    46    57    68    79    90   101
36      0    11    22    33    44    55    66    77    88    99
37      8    19    30    41    52    63    74    85    96   107
38 Y =
39     11    22    33    44    55    66    77    88    99   110
```

4.4.2 Multi-product manufacturing system

The idea for this example has been taken from [Baccelli et al. 1992]. Consider a manufacturing system that consists of three machines (M_1, M_2 and M_3). In this manufacturing system three different types of parts (P_1, P_2 and P_3) are produced according to a certain product mix. The routes followed by the various types of parts are depicted in Fig. 7.
Parts of type P_1 first visit machine M_2 and then go to M_3. Parts of type P_2

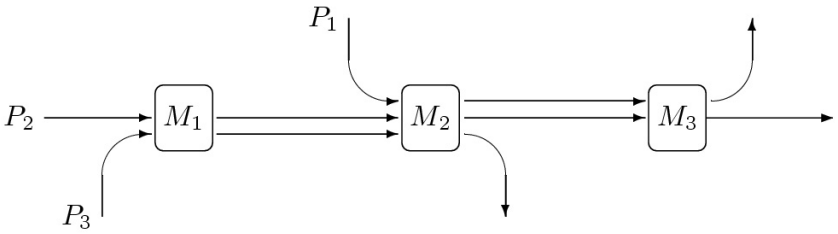

Figure 7: The routing of the various types of parts along the machines.

enter the system via machine M_1, then they go to machine M_2 and finally leave the system through machine M_3. Parts of type P_3 first visit machine M_1 and then go to M_2. It is assumed that:

- Parts are carried around on pallets. There is one pallet available for each type of part.
- It is assumed that the transportation times are negligible and that there are no set-up times on the machines when they switch from one part type to another.
- The sequencing of the various parts on the machines is known: on machine M_1 it is (P_2, P_3), i.e. the machine first processes a part of type P_2 and then a part of type P_3, on machine M_2 the sequence is (P_1, P_2, P_3), and (P_1, P_2) on machine M_3. We will call these sequences *local dispatching rules* and we will describe them as σ (i.e. σ_1 for the sequence on M_1, σ_2 for the sequence on M_2, and σ_3 for M_3).

The information about the sequencing and the duration of the various activities (processing times) is shown in Fig. 8. In this figure, the activities are represented by ordered pairs of the form (P_i, M_j) meaning that a part of type P_i is processed on machine M_j. The arcs represent the precedence constraints between activities. At the bottom right of each activity we have indicated its duration, e.g. (P_1, M_2) (activity 3) has duration $d_3 = 3$.

In order to simplify the process of deriving the evolution equations of this system, we shall first look at what happens in one cycle of the production process. We define:

- $u_i(k)$ — time instant at which machine M_i is available for the first activity that should be performed on it in the kth production cycle for $i = 1, 2, 3$;
- $u_j(k)$ — time instant at which the raw material for a part of type P_{j-3} is available in the kth production cycle for $j = 4, 5, 6$;
- $x_i(k)$ — time instant at which activity i starts in the kth production cycle for $i = 1, 2, \ldots, 7$;

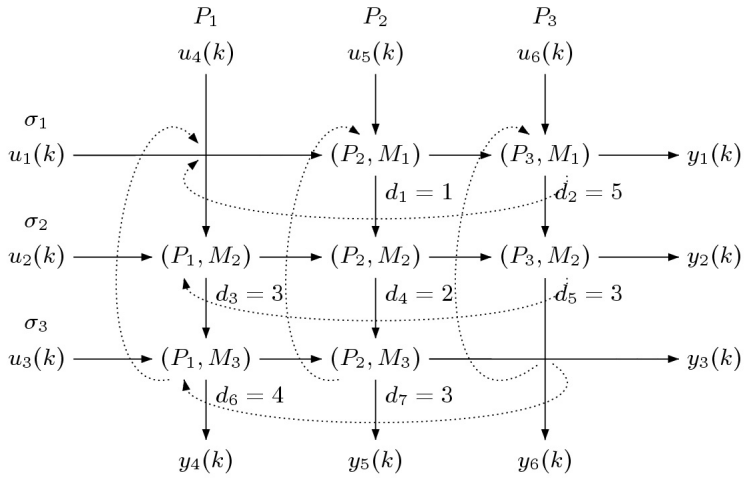

Figure 8: The sequence and the duration of the various activities.

- $y_i(k)$ — time instant at which machine M_i has finished processing the last part of the kth production cycle that should be processed on it for $i = 1, 2, 3$;
- $y_j(k)$ — time instant at which the finished product of type P_{j-3} of the kth production cycle has been completed for $j = 4, 5, 6$.

We have the following evolution equations:

$$\begin{aligned} x_1(k) &= 5x_2(k-1) \oplus 3x_7(k-1) \oplus u_1(k) \oplus u_5(k), \\ x_2(k) &= 1x_1(k) \oplus 3x_5(k-1) \oplus u_6(k), \\ &\vdots \end{aligned} \tag{59}$$

or, more compactly:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{A}_0\mathbf{x}(k) \oplus \mathbf{A}_1\mathbf{x}(k-1) \oplus \mathbf{B}_0\mathbf{u}(k), \\ &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{u}(k), \end{aligned} \tag{60}$$

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

In the cyclic (closed) systems, it is advisable to introduce a matrix $\mathbf{K} \in \mathbb{R}_\varepsilon^{r\times m}$, to describe the dynamics of restarting the system for the next cycle:

$$\mathbf{u}(k) = \mathbf{K}\mathbf{y}(k-1). \tag{61}$$

Hence:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{K}\mathbf{y}(k-1), && (62)\\ &= \mathbf{A}\mathbf{x}(k-1) \oplus \mathbf{B}\mathbf{K}\mathbf{C}\mathbf{x}(k-1), && (63)\\ &= (\mathbf{A} \oplus \mathbf{B}\mathbf{K}\mathbf{C})\mathbf{x}(k-1), && (64)\\ &= \hat{\mathbf{A}}\mathbf{x}(k-1). && (65) \end{aligned}$$

So, we obtain an autonomous model.

In the steady–state, the difference in the same state variables of two subsequent cycles is (in the conventional algebra):

$$\mathbf{x}(k) = \mathbf{x}(k-1) + T, \tag{66}$$

or in (max, +):

$$\mathbf{x}(k) = T\mathbf{x}(k-1), \tag{67}$$

where T is a cycle time.
Hence, from (65) and (67):

$$\mathbf{A}\mathbf{x}(k-1) = T\mathbf{x}(k-1), \tag{68}$$

where T is a (max, +) eigenvalue of $\mathbf{A}$.

An exemplary analysis of the considered system based on the Max–Plus Algebra Toolbox is shown below. It provides cycle time and consecutive states ($\mathbf{x}(k)$) of the system. It is assumed that $\mathbf{x}(0) = [\varepsilon]^{7\times 1}$, i.e. all machines are idle in the beginning, and $\mathbf{u}(1) = [e]^{6\times 1}$, i.e. all machines can be started without delay. This example is included in files `exMultiProduct.m` and `exGanttr.m`.

```
% operation times
>> d = [1 5 3 2 3 4 3];

% matrices definition
>> A0 = mp_zeros(7);
>> A0(2,1) = d(1); A0(4,1) = d(1); A0(4,3) = d(3); A0(5,2) = d(2);
>> A0(5,4) = d(4); A0(6,3) = d(3); A0(7,4) = d(4); A0(7,6) = d(6);

>> A1 = mp_zeros(7);
>> A1(1,2) = d(2); A1(1,7) = d(7); A1(2,5) = d(5);
>> A1(3,5) = d(5); A1(3,6) = d(6); A1(6,7) = d(7);

>> B0 = mp_zeros(7,6);
>> B0(1,1) = mp_one; B0(1,5) = mp_one; B0(2,6) = mp_one;
>> B0(3,2) = mp_one; B0(3,4) = mp_one; B0(6,3) = mp_one;

```

```
>> C = mp_zeros(6,7);
>> C(1,2) = d(2); C(2,3) = d(3); C(3,7) = d(7);
>> C(4,6) = d(6); C(5,7) = d(7); C(6,5) = d(5);

>> K = mp_eye(6);

% create the matrices A=(A0^*A1) and B=(A0^*B0)
>> A = mp_multi(mp_star(A0), A1);
>> B = mp_multi(mp_star(A0), B0);

% create matrix M = (A \oplus BKC)
>> M = mp_add(A, mp_multi(B, mp_multi(K, C)));

% determine initial conditions
x0 = mp_zeros(7, 1);
u1 = mp_ones(6, 1);

% calculate a sequence of a state vector
>> X(:, 1) = mp_add(mp_multi(A, x0), mp_multi(B, u1));
>> for i = 2:10
       X(:, i) = mp_multi(M, X(:, i-1));
   end
>> X
X =
     0    10    19    29    38    48    57    67    76    86
     1    11    20    30    39    49    58    68    77    87
     0     9    19    28    38    47    57    66    76    85
     3    12    22    31    41    50    60    69    79    88
     6    16    25    35    44    54    63    73    82    92
     3    12    22    31    41    50    60    69    79    88
     7    16    26    35    45    54    64    73    83    92

% calculate a sequence of an output vector
>> Y = mp_multi(C, X)
Y =
     6    16    25    35    44    54    63    73    82    92
     9    19    28    38    47    57    66    76    85    95
    10    19    29    38    48    57    67    76    86    95
     7    16    26    35    45    54    64    73    83    92
    10    19    29    38    48    57    67    76    86    95
     9    19    28    38    47    57    66    76    85    95

% cycle time
>> lambda = mp_mcm(M)
lambda = 9.5000

% does the system start in steady-state?
>> mp_isegv(M, X(:,1), lambda)
ans = 0
```

```
66
67  % this means NO, so, let us calculate new x0 for start in steady state
68  >> x0 = mp_egv1(M, mp_ones(7,1))
69  x0 =
70     14.5000
71     15.5000
72     14.0000
73     17.0000
74     20.5000
75     17.0000
76     21.0000
77
78  % let min(x0)==0
79  >> x0 = x0 - min(x0);
80
81  % calculate a new sequence of a state vector
82  >> X(:, 1) = x0;
83  >> for i = 2:10
84         X(:, i) = mp_multi(M, X(:, i-1));
85     end
86
87  >> X
88  X =
89     0.50  10.00  19.50  29.00  38.50  48.00  57.50  67.00  76.50  86.00
90     1.50  11.00  20.50  30.00  39.50  49.00  58.50  68.00  77.50  87.00
91        0   9.50  19.00  28.50  38.00  47.50  57.00  66.50  76.00  85.50
92     3.00  12.50  22.00  31.50  41.00  50.50  60.00  69.50  79.00  88.50
93     6.50  16.00  25.50  35.00  44.50  54.00  63.50  73.00  82.50  92.00
94     3.00  12.50  22.00  31.50  41.00  50.50  60.00  69.50  79.00  88.50
95     7.00  16.50  26.00  35.50  45.00  54.50  64.00  73.50  83.00  92.50
96
97  % calculate others preformance indices
98  % resources utilisation level (for M1, M2, M3 respectively):
99  >> ro = [d(1)+d(2) d(3)+d(4)+d(5) d(6)+d(7)] / lambda
100 ro =
101     0.6316    0.8421    0.7368
102
103 % processes execution level (P1, P2, P3):
104 >> eta = [d(3)+d(6) d(1)+d(4)+d(7) d(2)+d(5)] / lambda
105 eta =
106     0.7368    0.6316    0.8421
```

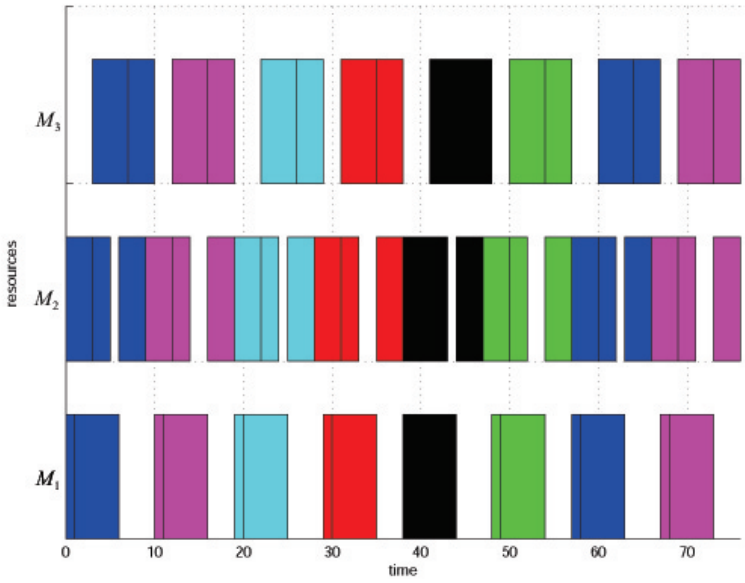

Figure 9: A usage of machines in time, the Gantt chart generated by function `mp_ganttr`.

5 Miscellaneous functions and data structures

5.1 Functions

Table 11: Miscellaneous functions.

function	short description
mp_mxconv	matrix conversion from the (max, +) to the (min, +) and vice versa
mp_mx2latex	matrix (or vector) conversion from the (max, +) to the LaTeX
mpm_mx2latex	matrix (or vector) conversion from the (min, +) to the LaTeX

5.2 Data structures

Definition 5.1 *resources–state–vector matrix* ***MPX_Rsv***
Let us define a matrix MPX_Rsv of resources–state–vector connection, $\texttt{MPX\_Rsv} \in \mathbb{N}_0^{p\times q}$, where p is a number of resources in the system, q is a maximal amount of the state–vector entries, which describe one resource, i.e. the i–th row define i–th resource, entries of this row are numbers of entries of the state–vector which are described this resource.

Example 5.2
Let us consider an example from § 4.4.2.
There are 3 resources M_1, M_2 and M_3. Resource M_1 is described by states x_1, x_2, M_2 by x_3, x_4, x_5, and M_3 by x_6, x_7. So

$$\texttt{MPX\_Rsv} = \begin{bmatrix} M_1 & = & \{x_1, & x_2\} & \\ M_2 & = & \{x_3, & x_4, & x_5\} \\ M_3 & = & \{x_6, & x_7\} & \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 5 \\ 6 & 7 & 0 \end{bmatrix}.$$

Rows 1 and 3 have 2 elements, the row 2 has 3 elements, thus we have to fill missing element by 0 to obtain the same length of all rows (add zeros in rows 1 and 3).
See function mp_ganttr or example exGanttr.m for details.

6 Toolbox function reference

All functions described below are listed in alphabetical order.

---

mp_add

(max, +) addition of scalars, vectors or matrices

Syntax

```
y = mp_add(n, m)
```

Description

$y = n \oplus m$

- If `n` and `m` are scalars, result is a (max, +) sum of `n` and `m`.
- If `n` (or `m`) is scalar and `m` (or `n`) is vector, result is a vector the same size as `m` (or `n`) where for every entries is (max, +) added `n` (or `m`).
- If `n` (or `m`) is scalar and `m` (or `n`) is matrix, result is a matrix the same size as `m` (or `n`) where for every entries is (max, +) added `n` (or `m`).
- If `n` and `m` are this same size matrices, result is an array the same size as `n` (and `m`) with the entries equal to (max, +) addition elements from `n` and `m`.

Example

```
>> mp_add(-5, 7)
ans =
        7

>> mp_add(3, [mp_zero 4])
ans =
        3     4

>> A = [1 6 8 ;  3 -Inf 4], B = [2 5 -Inf ; 3 1 3]
A =
        1        6        8
        3     -Inf        4

```

```
B =
        2        5     -Inf
        3        1        3

>> mp_add(A, B)
ans =
        2        6        8
        3        1        4
```

See also

mp_multi, mp_one, mp_ones, mp_zero, mp_zeros

---

mp_conv

scalar, vector or matrix conversion from the (max, +) to the (min, +) and vice versa

Syntax

```
Z = mp_conv(X)
```

Description

Function exchanges all ∞ and $-\infty$ to $-\infty$ and ∞ respectively. Entries with other values are not changing.

Example

```
>> A = [0        3      Inf        1
        1        2        2     -Inf
     -Inf      Inf        1        0];

>> mp_conv(A)
ans =
        0        3     -Inf        1
        1        2        2      Inf
      Inf     -Inf        1        0
```

---

mp_div

(max, +) division

Syntax

```
Z = mp_div(A, B)
[Z, err] = mp_div(A, B)
```

Description

$Z = A \oslash B$

- If `A` is a scalar
  - if `B` is a scalar, result is a scalar: (max, +) division of `A` by `B`;
  - if `B` is a vector (or matrix), result is a vector (or a matrix) where every entries are: `A` divided by apropriate entry of `B`.
- If `A` is a vector (or a matrix)
  - if `B` is a scalar, result is a vector (or a matrix) the same size as `A` where every entries of `A` are (max, +) divided by `B`;
  - if `B` is a vector (or a matrix) the same size as `A`: the result is (max, +) division `A` by `B`, i.e. `mp_multi(mp_inv(B), A)`, and when `B` is (max, +)-invertable `err = 0`, otherwise `err = 1`;
  - if `B` is a vector (or a matrix) different size than `A` — operation is not defined.
- Division by (max, +) zero (`-Inf`) is not defined - and returns `NaN`.

Example

```
>> mp_div(3, mp_one)
ans =
        3

>> mp_div(3, 3)
ans =
        0

>> mp_div(3, mp_zero)
ans =
      NaN

>> mp_div(3, [mp_zero 7 mp_one])
ans =
      NaN       -4        3
```

```
>> mp_div([mp_zero 7 mp_one], 3)
ans =
     -Inf        4       -3
```

(max, +) division for square matrices:

```
>> A = [mp_zero  1        mp_zero;
        2        mp_zero  mp_zero;
        mp_zero  mp_zero  3];
>> B = [3        mp_zero  mp_zero;
        mp_zero  mp_zero  4;
        mp_zero  5  mp_zero];
>> [C, err] = mp_div(A, B)
C =
     -Inf       -2     -Inf
     -Inf     -Inf       -2
       -2     -Inf     -Inf

err =
        0

>> mp_multi(B, C)
ans =
     -Inf        1     -Inf
        2     -Inf     -Inf
     -Inf     -Inf        3
```

and not square matrices:

```
>> A = [mp_zero  1  mp_zero;  2  mp_zero  mp_zero];
>> B = [3  mp_zero  mp_zero;  mp_zero  mp_zero  4];
>> [C, err] = mp_div(A, B)
C =
     -Inf       -2     -Inf
     -Inf     -Inf     -Inf
       -2     -Inf     -Inf

err =
        0

>> mp_multi(B, C)
ans =
     -Inf        1     -Inf
        2     -Inf     -Inf
```

See also

mp_inv, mp_multi, mp_one, mp_zero

---

mp_egv_bo93

eigenvector and eigenvalue of matrix A

Syntax

```
eigenvector = mp_egv_bo93(A, x0)
eigenvector = mp_egv_bo93(A, x0, r)
[eigenvector, eigenvalue] = mp_egv_bo93(A, x0)
[eigenvector, eigenvalue] = mp_egv_bo93(A, x0, r)
```

Description

Function returns an eigenvector and an eigenvalue of matrix A by [Braker and G.-J. Olsder 1993], see alg. 2, page 23.

- A must be a square matrix, $A \in \mathbb{R}_{\varepsilon}^{n \times n}$
- algorithm starts from vector x0 $\in \mathbb{R}^n$
- r (optional) is the maximum number of steps, after which the algorithm stops — default r = 1000
- if A has more than one eigenvalue, function returns only one
- if A has more than one eigenvector associated with the eigenvalue, function returns only one

Example

Let us consider example 2.43.

```
>> A  = [mp_zero        3  mp_zero        1;
                2  mp_zero        1  mp_zero;
                1        2        2  mp_zero;
          mp_zero  mp_zero        1  mp_zero];

>> x0 = [      0  mp_zero  mp_zero  mp_zero]';

>> [v, l] = mp_egv_bo93(A, x0)
v =
  10.0000
```

```
   9.5000
   9.0000
   7.5000

l =
   2.5000
```

See also

mp_egv_o91, mp_egv_pqc, mp_egv_sw001, mp_egv_sw002, mp_is_egv1, mp_is_egv2, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_egv_o91

(candidate) eigenvector and eigenvalue of matrix `A`

Syntax

```
eigenvector = mp_egv_o91(A, x0)
eigenvector = mp_egv_o91(A, x0, r)
[eigenvector, eigenvalue] = mp_egv_o91(A, x0)
[eigenvector, eigenvalue] = mp_egv_o91(A, x0, r)
```

Description

Function returns a (candidate) `eigenvector` and an `eigenvalue` of matrix `A` by [G.-J. Olsder 1991] — alg. 1, page 23.

- `A` must be a square matrix, $\mathtt{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$,
- algorithm starts from vector $\mathtt{x0} \in \mathbb{R}^n$,
- `r` (optional) is the maximum number of steps, after which the algorithm stops, default $\mathtt{r} = 1000$
- If `A` has more than one eigenvalue, function returns only one.
- If `A` has more than one eigenvector associated with the `eigenvalue`, function returns only one.

Example

Let us consider example 2.42.

```
>> A = [3 7; 2 4], x0 = [0 0]'
A =
```

```
         3         7
         2         4

x0 =
         0
         0

>> [v, l] = mp_egv_o91(A, x0)
v =
   9.0000
   6.5000

l =
   4.5000
```

See also

mp_egv_bo93, mp_egv_pqc, mp_egv_sw001, mp_egv_sw002, mp_is_egv1, mp_is_egv2, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_egv_pqc

eigenvalue of A calculated form components p, q and c

Syntax

```
l = mp_egv_pqc(p, q, c)
l = mp_egv_pqc([p q c])
```

Description

- Function calculates an eigenvalue of matrix A from components p,q and c (see theorem 2.40).
- The components can be calculated by mp_pqc.
- $p,q \in \mathbb{N}_0 : p > q \geqslant 0$, $c \in \mathbb{R}$

Example

Let us consider example 2.41.

```
>> A = [3 7; 2 4]
A =
```

```
         3         7
         2         4

>> x0 = [0 0]'
x =
         0
         0

>> mp_egv_pqc(mp_pqc(A, x0))
ans =
   4.5000
```

See also

mp_pqc, mp_egv_bo93, mp_egv_o91, mp_egv_sw001, mp_egv_sw002, mp_is_egv1, mp_is_egv2, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_egv_sw001

eigenvector and eigenvalue of matrix `A`

Syntax

```
eigenvector = mp_egv_sw001(A, x0)
eigenvector = mp_egv_sw001(A, x0, r)
[eigenvector, eigenvalue] = mp_egv_sw001(A, x0)
[eigenvector, eigenvalue] = mp_egv_sw001(A, x0, r)
```

Description

Function returns an `eigenvector` and an `eigenvalue` of matrix `A` by [Subiono and van der Woude 2000], see alg. 4, page 28.

- `A` must be a square matrix, $\mathtt{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$,
- algorithm starts from vector $\mathtt{x0} \in \mathbb{R}^n$,
- `r` (optional) is the maximum number of steps, after which the algorithm stops, default $\mathtt{r} = 1000$.
- If `A` has more than one eigenvalue, function returns only one.
- If `A` has more than one eigenvector associated with the `eigenvalue`, function returns only one.

Example

Let us consider example 2.46.

```
>> A = [3 7; 2 4], x0 = [0 0]'
A =
        3        7
        2        4

x0 =
        0
        0

>> [v, l] = mp_egv_sw001(A, x0)
v =
  11.5000
   9.0000

l =
   4.5000
```

See also

mp_egv_bo93, mp_egv_o91, mp_egv_pqc, mp_egv_sw002, mp_is_egv1, mp_is_egv2, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_egv_sw002

eigenvector and eigenvalue of matrix `A`

Syntax

```
eigenvector = mp_egv_sw002(A, x0)
eigenvector = mp_egv_sw002(A, x0, r)
[eigenvector, eigenvalue] = mp_egv_sw002(A, x0)
[eigenvector, eigenvalue] = mp_egv_sw002(A, x0, r)
```

Description

Function returns an `eigenvector` and an `eigenvalue` of matrix `A` by [Subiono and van der Woude 2000], see alg. 5, page **??**.

- `A` must be a square matrix, $\texttt{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$,
- algorithm starts from vector `x0` $\in \mathbb{R}^n$,
- `r` (optional) is the maximum number of steps, after which the algorithm stops, default `r` $= 1000$.
- If `A` has more than one eigenvalue, function returns only one.
- If `A` has more than one eigenvector associated with the `eigenvalue`, function returns only one.

Example

Let us consider example 2.45.

```
>> A  = [mp_zero        3  mp_zero        1;
               2  mp_zero        1  mp_zero;
               1        2        2  mp_zero;
         mp_zero  mp_zero        1  mp_zero];

>> x0 = [      0  mp_zero  mp_zero  mp_zero]';

>> [v, l] = mp_egv_sw002(A, x0)
v =
  12.5000
  12.0000
  11.5000
  10.0000

l =
   2.5000
```

See also

mp_egv_bo93, mp_egv_o91, mp_egv_pqc, mp_egv_sw001, mp_is_egv1, mp_is_egv2, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_ev_fw

eigenvectors of matrix `A`

Syntax

```
[eigenvectors] = mp_eg_fw(A)
```

Description

Function returns a set of eigenvectors of `A` from final Floyd–Warshall matrix (genenerated by `mp_mcm_fw` or `mp_mx_fw`), without recurrent eigenvectors, as well as without their linear combinations. Function uses the Floyd–Warshall algorithm Ahuja, Magnanti, and Orlin 1993 adapted by G.-J. Olsder, Roos, and van Egmond 1999 — see step 3 of alg. 5, page 25.

Example

Let us consider example 2.44.

```
>> A=[      4  mp_zero  mp_zero  mp_zero  mp_zero        4;
            2        5        1  mp_zero  mp_zero  mp_zero;
      mp_zero  mp_zero  mp_zero        6  mp_zero  mp_zero;
      mp_zero  mp_zero        8        3  mp_zero  mp_zero;
            9  mp_zero  mp_zero  mp_zero  mp_zero  mp_zero;
      mp_zero  mp_zero  mp_zero  mp_zero        8
         3];

>> F = mp_mx_fw(A, 7)
F =
        0     -Inf     -Inf     -Inf       -2       -3
       -5       -2       -6       -7       -7       -8
     -Inf     -Inf        0       -1     -Inf     -Inf
     -Inf     -Inf        1        0     -Inf     -Inf
        2     -Inf     -Inf     -Inf        0       -1
        3     -Inf     -Inf     -Inf        1        0

>> ev = mp_ev_fw(F)
ev =
        0        0       -2       -3       -2
       -5       -5       -6       -6       -7
        0       -1        0        0       -1
        1        0        1        1        0
        2        2        0       -1        0
        3        3        1        0        1
```

See also

`mp_mcm_fw`, `mp_mx_fw`

mp_eye

(max, +) identity matrix

Syntax

```
Y = mp_eye
Y = mp_eye(n)
Y = mp_eye(n, m)
```

Description

- `mp_eye` returns 0.
- `mp_eye(n)` or `mp_eye([n])` returns an `n`–by–`n` (max, +) identity matrix, i.e. with (max, +) units on the main diagonal and ε elsewhere.
- `mp_eye(n, m)` or `mp_eye([n m])` returns an `n`–by–`m` (max, +) identity matrix.

Example

```
>> mp_eye
ans =
        0

>> mp_eye(2)
ans =
        0     -Inf
     -Inf        0

>> mp_eye(2, 3)
ans =
        0     -Inf     -Inf
     -Inf        0     -Inf
```

See also

mp_one, mp_ones, mp_zero, mp_zeros, mp_randi

---

mp_ganttr

Gantt chart of resources occupation in time

Syntax

```
mp_ganttr(X, time, MPX_Rsv)
mp_ganttr(X, time, MPX_Rsv, xrange)
mp_ganttr(X, time, MPX_Rsv, ytick)
mp_ganttr(X, time, MPX_Rsv, xrange, ytick)
```

Description

A Gantt chart of resources occupation in time, where

`X`	a $n \times m$ matrix of state vectors, i.e. a collection of m successive state vectors
`time`	a $n \times 1$ vector of operation times for every entry in state vector (time of every event/operation), or a $n \times m$ matrix of operation times for m iterations (time of every event/operation in every iteration)
`MPX_Rsv`	a matrix of resources–state–vector connection for details see def. 5.1
`xrange`	(optional) enables to specify limits of the x axis, `xrange = [xmin xmax]`, by default, function finds the maximum and minimum of the data i.e. `xmin = min(X)`, `xmax = max(X+t)`
`ytick`	`ytick = 0|1` (optional, default `ytick = 1`) if `ytick = 1` then every ytick is marked along y axis

Example

```
% collections of state vectors
>> V = [0   10   19   29   38   48   57   67   76   86;
        1   11   20   30   39   49   58   68   77   87;
        0    9   19   28   38   47   57   66   76   85;
        3   12   22   31   41   50   60   69   79   88;
        6   16   25   35   44   54   63   73   82   92;
        3   12   22   31   41   50   60   69   79   88;
        7   16   26   35   45   54   64   73   83   92];

% operation times
>> op_times = [1 5 3 2 3 4 3]';

% matrix of resources–state–vector connection
>> R_V = [[1 2 0]
          [3 4 5]
          [6 7 0]];

>> mp_ganttr(X(:, 1:8), t, R);
```

A Gantt chart generated by the code presented above is shown in fig. 9.

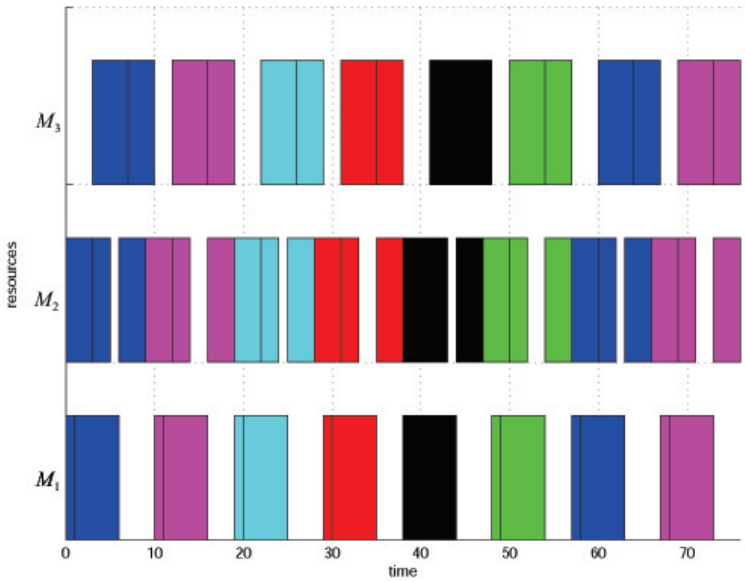

Figure 10: A usage of resources in time, the Gantt chart generated by function `mp_ganttr`.

See also

`mp_ganttx`

---

mp_ganttx

Gantt chart of a state vector evolution in time
Gantt chart of execution of operations

Syntax

```
mp_ganttx(X, time)
mp_ganttx(X, time, xrange)
mp_ganttx(X, time, ytick)
mp_ganttx(X, time, xrange, ytick)
```

Description

A Gantt chart of a state vector evolution in time (or execution of operations)

`X`	a $n \times m$ matrix of state vectors, i.e. a collection of m successive state vectors
`t`	a $n \times 1$ vector of operation times for every entry in state vector (time of every event/operation), or a $n \times m$ matrix of operation times for m iterations (time of every event/operation in every iteration)
`xrange`	(optional) enables to specify limits of the x axis, `xrange = [xmin xmax]`, by default, function finds the maximum and minimum of the data i.e. `xmin = min(X)`, `xmax = max(X+t)`
`ytick`	`ytick = 0|1` (optional, default `ytick = 1`) if `ytick = 1` then every ytick is marked along y axis

Example

A part of code from file `exGanttx.m` demonstrates use of `mp_ganttx`:

```
1  % collections of 4 state vectors
2  V = [    14        1060        2196        2762
3         201        1232        2368        2934
4         374        1400        2536        3102
```

⋮

```
25        331         921        1432        2537];
26
27 operation_times = [172;
28                    168;
29                     87;
```

⋮

```
50                    321];
51
52 mp_ganttx(V, operation_times, [0 4000], 0);
```

A Gantt chart generated by the code presented above is shown in fig. 11.

See also

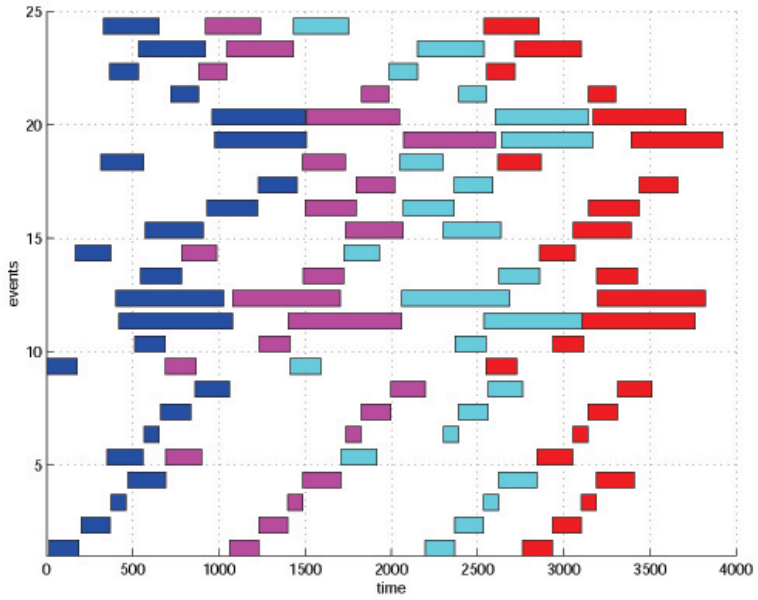

Figure 11: An exemplary Gantt chart generated by function `mp_ganttx`.

mp_ganttr

---

mp_inv

(max, +) matrix invertion

Syntax

```
Y = mp_inv(A)
[Y, err] = mp_inv(A)
```

Description

`Y` = $\mathtt{A}^{-1}$

- If `A` is a scalar then `Y = mp_power(A, -1)` and `err = 0`.
- If `A` is a square matrix then `Y` = $\mathtt{A}^{-1}$ and if `AY = YA = I` then `err = 0`, otherwise `err = 1`.
- If `A` not a square matrix then `Y = (I-A)'` and if `AY = I` then `err = 0`, otherwise `err = 1`.

Example

Let us consider a square matrix:

```
>> mp_inv(5)
ans =
       -5

>> A = [mp_zero  1        mp_zero;
        2        mp_zero  mp_zero;
        mp_zero  mp_zero  3];

>> [Y, err] = mp_inv(A)
Y =
     -Inf       -2     -Inf
       -1     -Inf     -Inf
     -Inf     -Inf       -3

err =
        0

>> mp_multi(A, Y)
ans =
        0     -Inf     -Inf
     -Inf        0     -Inf
     -Inf     -Inf        0
```

And now not a square matrix:

```
>> A = [mp_zero  1        mp_zero;
        2        mp_zero  mp_zero];

>> [Y, err] = mp_inv(A)
Y =
     -Inf       -2
       -1     -Inf
     -Inf     -Inf

err =
        0

>> mp_multi(A, Y)
ans =
        0     -Inf
     -Inf        0
```

See also

mp_power, mp_multi, mp_div, mp_eye

---

mp_is_egv1

checks, if vector x is an eigenvector of A

Syntax

```
y = mp_is_egv1(A, x)
y = mp_is_egv1(A, x, eigenvalue)
y = mp_is_egv1(A, x, eigenvalue, d)
```

Description

Function returns 1 if x is an eigenvector of A, otherwise 0.

- A must be a square matrix, $\mathtt{A} \in \mathbb{R}_{\varepsilon}^{n \times n}$,
- $\mathtt{x} \in \mathbb{R}^{n}$,
- eigenvalue of A (optional), $\mathtt{eigenvalue} \in \mathbb{R}$, default it is computed by mp_mcm(A),
- d (optional) a number of arcs in the maximum cycle mean of $\mathcal{G}(\mathtt{A})$, default $\mathtt{d} = 1$,
- if $\mathtt{d} \in \mathbb{N}$ is given, the function checks eq. (69) and returns 1 if (69) is fulfiled, 0 otherwise.

$$\mathtt{A}^{\mathtt{d}}\mathtt{x} = \mathtt{eigenvalue}^{\mathtt{d}}\mathtt{x}, \tag{69}$$

Example

```
>> A = [2 5; 3 3]; x = [1 0]';
>> mp_is_egv1(A, x)
ans =
        1

>> % let's check:
>> mp_multi(A, x)
ans =
        5
        4

>> mp_multi(mp_mcm(A), x)
ans =
        5
        4
```

See also

mp_egv_bo93, mp_egv_o91, mp_egv_pqc, mp_egv_sw001, mp_egv_sw002, mp_is_egv2, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_is_egv2

checks, if vector x is an eigenvector of A

Syntax

```
y = mp_is_egv1(x2, x1, eigenvalue)
y = mp_is_egv1(x2, x1, eigenvalue, d)
```

Description

Function returns 1 if eigenvalue $\otimes$ x1 = x2, 0 otherwise.

- x1, x2 $\in \mathbb{R}^n$, eigenvalue $\in \mathbb{R}$,
- if d $\in \mathbb{N}$ is given (optional, default d = 1) then
  - function returns 1 if eigenvalue$^{\text{d}}$ $\otimes$ x1 = x2, 0 otherwise.

Example

```
>> A = [2 5; 3 3]
A =
        2        5
        3        3
>> lambda = mp_mcm(A)
lambda =
        4

>> x = [1 0]'
x =
        1
        0

>> mp_multi(A, x)
ans =
        5
        4
```

```
18
19  >> mp_is_egv2(ans, x, lambda)
20  ans =
21          1
```

See also

mp_egv_bo93, mp_egv_o91, mp_egv_pqc, mp_egv_sw001, mp_egv_sw002, mp_is_egv1, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_is_pga

is a precedence graph $\mathcal{G}(\mathtt{A})$ acyclic?
has a matrix `A` got any eigenvalue?

Syntax

```
y = mp_is_pga(A)
```

Description

- Function returns 1:
  - if $\mathcal{G}(\mathtt{A})$ is acyclic ($\mathcal{G}(\mathtt{A})$ does not contain any circuit), or equivalently
  - if `A` has NOT got any eigenvalue,
- 0 otherwise.
- `A` must be a square matrix.

Example

Let us consider an acyclic digraph from Fig. 1, matrix `A` represents this graph.

```
1  >> A = [mp_zero        4  mp_zero;
2                2  mp_zero  mp_zero;
3          mp_zero        3        2];
4
5  >> mp_is_pga(A)
6  ans =
7          0
```

See also

mp_is_pgc, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_karp, mp_mcm_fw, mp_isegv, mp_isegvv, mp_compute_pqc, mp_lpqc, mp_FloydWarshallMx, mp_egvFloydWarshall, mp_egv1, mp_egv2, mp_egv3, mp_egv4

---

mp_is_pgc

is precedence graph $\mathcal{G}(\mathtt{A})$ connected?

Syntax

```
y = mp_is_pgc(A)
```

Description

- Function returns 1 if precedence graph $\mathcal{G}(\mathtt{A})$ is connected, 0 otherwise.
- `A` must be a square matrix.

Example

Let us consider a digraph from Fig. 1.

```
>> A = [mp_zero        4  mp_zero;
              2  mp_zero  mp_zero;
       mp_zero        3        2];

>> mp_is_pgc(A)
ans =
        1
```

See also

mp_is_pga, mp_is_pgsc1, mp_is_pgsc2

---

mp_is_pgsc1

is precedence graph $\mathcal{G}(\mathtt{A})$ strongly connected?
is matrix `A` irreducible?
has matrix `A` got exactly one eigenvalue?

Syntax

```
y = mp_is_pgsc1(A)
```

Description

- Function returns 1:
  - if $\mathcal{G}(\mathtt{A})$ is strongly connected, or equivalently
  - if matrix `A` is irreducible, or
  - if matrix `A` has got exactly one eigenvalue,
- 0 otherwise.
- `A` must be a square matrix.
- Result is calculated based directly on definition 2.27

Example

Let us consider a digraph from Fig. 1.

```
>> A = [mp_zero        4  mp_zero;
              2  mp_zero  mp_zero;
       mp_zero        3        2];

>> mp_is_pgsc1(A)
ans =
       0
```

See also

mp_is_pga, mp_is_pgc, mp_is_pgsc2, mp_mcm, mp_mcm_karp, mp_mcm_fw, mp_isegv, mp_isegvv, mp_compute_pqc, mp_lpqc, mp_FloydWarshallMx, mp_egvFloydWarshall, mp_egv1, mp_egv2, mp_egv3, mp_egv4

---

mp_is_pgsc2

is precedence graph $\mathcal{G}(\mathtt{A})$ strongly connected?
is matrix `A` irreducible?
has matrix `A` got exactly one eigenvalue?

Syntax

```
y = mp_is_pgsc2(A)
```

Description

Like mp_is_pgsc1, function checks whether $\mathcal{G}(\mathtt{A})$ is strongly connected (from theorem 2.29).

- Function returns 1:
  - if $\mathcal{G}(\mathtt{A})$ is strongly connected, or equivalently
  - if matrix `A` is irreducible, or
  - if matrix `A` has got exactly one eigenvalue,
- 0 otherwise.
- `A` must be a square matrix.

See also

mp_is_pga, mp_is_pgc, mp_is_pgsc1, mp_mcm, mp_mcm_karp, mp_mcm_fw, mp_isegv, mp_isegvv, mp_compute_pqc, mp_lpqc, mp_FloydWarshallMx, mp_egvFloydWarshall, mp_egv1, mp_egv2, mp_egv3, mp_egv4

---

mp_mcm

(max, +) eigenvalue of matrix `A`
maximum cycle mean of precedence graph $\mathcal{G}(\mathtt{A})$

Syntax

```
l = mp_mcm(A)
[l, d] = mp_mcm(A)
```

Description

- (max, +) eigenvalue of an irreducible matrix `A`
  - if `A` is not irreducible, it gives only one eigenvalue.
- Maximum cycle mean of precedence graph $\mathcal{G}(\mathtt{A})$
  - if $\mathcal{G}(\mathtt{A})$ is not strongly connected, it gives only one value.
- Function returns:
  - `l` — the eigenvalue of `A` or equivalently, the value of maximum cycle mean of precedence graph $\mathcal{G}(\mathtt{A})$, from (16)
  - `d` — the number of arcs in critical circuit.
- `A` must be a square matrix.

Example

Let us consider the digraph from Fig. 2.

```
>> A = [      5  mp_zero        5;
        mp_zero        6        3;
             11       12       11];

>> [l, d] = mp_mcm(A)
l =
       11
d =
        1
```

See also

mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm_fw, mp_mcm_karp, mp_isegv, mp_isegvv, mp_compute_pqc, mp_lpqc, mp_FloydWarshallMx, mp_egvFloydWarshall, mp_egv1, mp_egv2, mp_egv3, mp_egv4

---

mp_mcm_fw

(max, +) eigenvalue of matrix `A`
maximum cycle mean of precedence graph $\mathcal{G}$(`A`)
Floyd–Warshall's algorithm

Syntax

```
eigenvalue = mp_mcm_fw(A)
[eigenvalue, Afw] = mp_mcm_fw(A)
```

Description

- (max, +) eigenvalue of a matrix `A`
  - if `A` is not irreducible, it gives only one eigenvalue.
- Maximum cycle mean of precedence graph $\mathcal{G}$(`A`).
  - if $\mathcal{G}$(`A`) is not strongly connected, it gives only one value.
- Function returns:
  - `eigenvalue` — an eigenvalue of `A` (a maximum cycle mean of $\mathcal{G}$(`A`)),
  - `Afw` — a final Floyd–Warshall's matrix,
  - from `Afw` matrix can be obtained a set of eigenvectors by mp_egvFloydWarshall.
- `A` must be a square matrix.

Function uses the Floyd–Warshall procedure [Ahuja, Magnanti, and Orlin 1993] adapted by [G.-J. Olsder, Roos, and van Egmond 1999] — see steps 1–2 of alg. 5, page 25.

Example

Let us consider example 2.44

```
>> A=[4        mp_zero  mp_zero  mp_zero  mp_zero  4;
      2        5        1        mp_zero  mp_zero  mp_zero;
      mp_zero  mp_zero  mp_zero  6        mp_zero  mp_zero;
      mp_zero  mp_zero  8        3        mp_zero  mp_zero;
      9        mp_zero  mp_zero  mp_zero  mp_zero  mp_zero;
      mp_zero  mp_zero  mp_zero  mp_zero  8        3];

>> l = mp_mcm_fw(A)
l =
        7

>> v = [0 -5 0 1 2 3]';

>> mp_multi(A, v)' == mp_multi(l, v)'
ans =
        1        1        1        1        1        1
```

See also

mp_ev_fw, mp_mx_fw, mp_egv_bo93, mp_egv_o91, mp_egv_pqc, mp_egv_sw001, mp_egv_sw002, mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_is_egv1, mp_is_egv2, mp_mcm, mp_mcm_karp

---

mp_mcm_karp

(max, +) eigenvalue of matrix A
maximum cycle mean of precedence graph $\mathcal{G}$(A)
Karp's algorithm

Syntax

```
l = mp_mcm_karp(A)
```

Description

- (max, +) eigenvalue of an irreducible matrix A

  - if A is not irreducible, it gives only one eigenvalue.
- Maximum cycle mean of precedence graph $\mathcal{G}(\mathtt{A})$
  - if $\mathcal{G}(\mathtt{A})$ is not strongly connected, it gives only one value.
- Function returns:
  - the eigenvalue of A, or equivalently
  - the value of maximum cycle mean of precedence graph $\mathcal{G}(\mathtt{A})$.
- A must be a square matrix.

Karp's algorithm [Karp 1978] adapted by [Gaubert and Scilab 1998], see page 18.

Example

Let us consider the matrix **A** from Fig. 2.

```
>> A = [      5  mp_zero       5;
        mp_zero        6       3;
             11       12      11];

>> mp_mcm_karp(A)
ans =
       11
```

See also

mp_is_pga, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_isegv, mp_isegvv, mp_compute_pqc, mp_lpqc, mp_FloydWarshallMx, mp_egvFloydWarshall, mp_egv1, mp_egv2, mp_egv3, mp_egv4

---

mp_multi

(max, +) multiplication of scalars, vectors or matrices

Syntax

```
y = mp_multi(m, n)
```

Description

$\mathtt{y} = \mathtt{n} \otimes \mathtt{m}$

- If n and m are scalars, result is a (max, +) multiplication of n and m.

- If `n` (or `m`) is scalar and `m` (or `n`) is vector, result is a vector the same size as `m` (or `n`) where for every entries is (max, +) multiplication by `n` (or `m`).
- If `n` (or `m`) is scalar and `m` (or `n`) is matrix, result is a matrix the same size as `m` (or `n`) where for every entries is (max, +) multiplication by `n` (or `m`).
- If `n` is an $p \times q$ matrix and `m` is an $q \times r$ matrix result is a (max, +) product `n` and `m` returns an $p \times r$ matrix.

Example

```
>> mp_multi(-5, 7)
ans =
        2

>> mp_multi(3, [mp_zero 4])
ans =
     -Inf        7

>> A=[1 6 2; 8 3 4],  B=[2 5; 3 3; 1 6]
A =
        1        6        2
        8        3        4

B =
        2        5
        3        3
        1        6

>> mp_multi(A, B)
ans =
        9        9
       10       13
```

See also

mp_add, mp_div, mp_inv, mp_power, mp_one, mp_ones, mp_zero, mp_zeros

---

mp_mx_fw

final Floyd–Warshall's matrix

Syntax

`F = mp_mx_fw(A, eigenvalue)`

Description

Floyd–Warshall algorithm [Ahuja, Magnanti, and Orlin 1993] adapted by [G.-J. Olsder, Roos, and van Egmond 1999] — see alg. 3, page 25 for details.

- Function returns a final Floyd–Warshall matrix `F` from square matrix `A` and its `eigenvalue`.
- From this matrix can be obtained set of eigenvectors of `A` by `mp_ev_fw`.

Example

Let us consider example 2.44.

```
>> A=[      4  mp_zero  mp_zero  mp_zero  mp_zero        4;
            2        5        1  mp_zero  mp_zero  mp_zero;
      mp_zero  mp_zero  mp_zero        6  mp_zero  mp_zero;
      mp_zero  mp_zero        8        3  mp_zero  mp_zero;
            9  mp_zero  mp_zero  mp_zero  mp_zero  mp_zero;
      mp_zero  mp_zero  mp_zero  mp_zero        8
         3];

>> F = mp_mx_fw(A, 7)
F =
        0     -Inf     -Inf     -Inf       -2       -3
       -5       -2       -6       -7       -7       -8
     -Inf     -Inf        0       -1     -Inf     -Inf
     -Inf     -Inf        1        0     -Inf     -Inf
        2     -Inf     -Inf     -Inf        0       -1
        3     -Inf     -Inf     -Inf        1        0

>> ev = mp_ev_fw(F)
ev =
        0        0       -2       -3       -2
       -5       -5       -6       -6       -7
        0       -1        0        0       -1
        1        0        1        1        0
        2        2        0       -1        0
```

```
24          3        3        1        0        1
```

See also

mp_ev_fw, mp_mcm_fw

---

mp_mx2latex

matrix conversion from the Matlab (max, +) description to the LaTeX

Syntax

```
mp_mx2latex(X)
mp_mx2latex(X, 'fileName')
```

Description

(max, +) matrix (or vector) conversion from the Matlab®/Octave notation to the LaTeX source code.

- `X` is a matrix (or vector) to conversion;
- `fileName` (optional) is a name of file in which a LaTeX source code of `X` will be saved, default it is `mp_mx.tex`.

Example

```
>> A = [ 0     3   Inf     1
         1     2     2  -Inf
      -Inf   Inf     1     0];

>> mp_mx2latex(A, 'A.tex')
```

LaTeX source of A.tex:

```
\left[
\begin{array}{*{20}c}
0 & 3 & \infty & 1 \\
1 & 2 & 2 & \varepsilon \\
\varepsilon & \infty & 1 & 0 \\
\end{array}
\right]
```

LaTeX result:

$$\begin{bmatrix} 0 & 3 & \infty & 1 \\ 1 & 2 & 2 & \varepsilon \\ \varepsilon & \infty & 1 & 0 \end{bmatrix}$$

See also

mp_mx2latex, mp_conv

---

mp_one

(max, +) unit (0), neutral element for oprtation $\otimes$

Syntax

```
y = mp_one
```

Description

`mp_one` returns 0.

Example

```
>> mp_one
ans =
        0
```

See also

mp_ones, mp_zero, mp_zeros, mp_eye, mp_multi

---

mp_ones

(max, +) ones (0's) matrix, vector or scalar

Syntax

```
y = mp_ones
y = mp_ones(n)
y = mp_ones(n, m)
```

Description

- `mp_ones` returns 0.
- `mp_ones(n)` or `mp_ones([n])` is an `n`–by–`n` matrix of 0–s.
- `mp_ones(n, m)` or `mp_ones([n, m])` is an `n`–by–`m` matrix of 0–s.

Example

```
>> mp_ones
ans =
        0
>> mp_ones(2)
ans =
        0        0
        0        0

>> mp_ones(2, 3)
ans =
        0        0        0
        0        0        0
```

See also

mp_one, mp_zero, mp_zeros, mp_eye, mp_multi

---

mp_power

(max, +) raising to a power

Syntax

```
y = mp_power(x, n)
Y = mp_power(X, n)
```

Description

(max, +) `n`–th power of `x`, $y = x^n$
(max, +) `n`–th power of square matrix `X`, $Y = X^n$

- If `x` is a scalar than from definition 2.2:
  - $x \in \mathbb{R}_\varepsilon, n \in \mathbb{R} : x^n = x \times n$ (in conventional algebra).
  - If $n = 0$ then $x^0 = 0$.
  - If $x = \varepsilon$ and $n > 0$ then $\varepsilon^n = \varepsilon$.
  - If $x = \varepsilon$ and $n < 0$ then ε^n is not defined.

  - If `x` $= \varepsilon$ and `n` < 0 then $\varepsilon^0 = 0$ by definition.
- If `n` is a vector (or a matrix) then result is a vector (or a matrix) this same size, where every entries is (max, $+$) power of `x` to the proper element from `n`.
- If `X` is a square matrix then `n` must belong to $\mathbb{N}_0 \cup \{-1\}$, see definition 2.15.
  - If `n` $\notin \mathbb{N}_0 \cup \{-1\}$ operation is not defined.
- If `X` is not a scalar nor square matrix operation is not defined.

Example

```
>> mp_power(3, 3)
ans =
        9
>> mp_power(3, -2)
ans =
       -6

>> mp_power(3, 1/4)
ans =
   0.7500

>> mp_power(2, [1/4 1 4 -4])
ans =
  0.50000  2.00000  8.00000  -8.00000

>> mp_power([1 6; -Inf 3], 3)
ans =
        3       12
     -Inf        9
```

See also

mp_multi, mp_inv, mp_div, mp_eye, mp_one, mp_zero

---

mp_pqc

computes components of an eigenvalue for mp_egv_pqc

Syntax

```
[p q c] = mp_pqc(A, x0)
[p q c] = mp_pqc(A, x0, r)
```

Description

- Function calculates the components `p, q, c` of an eigenvalue of matrix `A` (see theorem 2.40).
- Algorithm starts from vector `x0`.
- `r` (optional, default `r = 1000`) is the maximum number of steps, after which the algorithm stops.
- The eigenvalue of `A` from the results of `mp_pqc` can be calculated by `mp_egv_pqc`.
- `A` must be a square matrix.

Example

Let us consider example 2.41.

```
>> A = [3 7; 2 4]
A =
        3        7
        2        4

>> x0 = [0 0]'
x =
        0
        0

>> mp_pqc(A, x0)
ans =
        3        1        9

>> mp_egv_pqc(ans)
ans =
   4.5000
```

See also

`mp_egv_bo93`, `mp_egv_o91`, `mp_egv_pqc`, `mp_egv_sw001`, `mp_egv_sw002`, `mp_is_egv1`, `mp_is_egv2`, `mp_is_pga`, `mp_is_pgsc1`, `mp_is_pgsc2`, `mp_mcm`, `mp_mcm_fw`, `mp_mcm_karp`

mp_randi

(max, +) random integer and ε

Syntax

```
Y = mp_randi(IMAX)
Y = mp_randi(IMAX, N)
Y = mp_randi(IMAX, M, N)
Y = mp_randi([IMIN IMAX], ...)
```

Description

- `mp_randi` returns a random integers in the range 1:IMAX (or IMIN:IMAX) + `mp_zero`.
- Additional arguments determine the shape of the return matrix `Y`.
- For more info look at `randi`.

Example

```
>> mp_randi([0 10])
ans =
        2

>> mp_randi([0 10], 3, 3)
ans =
        0        1        8
        3     -Inf        4
        0       10        9
```

See also

mp_ones, mp_zeros

---

mp_solve_Axb

greatest subsolution of `Ax = b`
(max, +) residuation operation

Syntax

```
x = mp_solvAxb(A, b)
[x, err] = mp_solvAxb(A, b)
```

Description

(max, +) residuation

- The greatest subsolution of `Ax` = `b`, computes the largest `x` such that `Ax` $\preccurlyeq$ `b` (see theorem 2.32).
- For `A` and `b` finite scalars, `x` = `A` − `b` (in conventional algebra).
- If `x` is the solution of `Ax` = `b` then `err = 0`, otherwise `err = 1`.

Example

```
% first example
>> a = 1; b = 2;
>> x = mp_solve_Axb(a, b)
x =
        1

% let's check
>> mp_multi(a, x)
ans =
        2

% second example
>> A = [2 3; 4 5]
A =
        2        3
        4        5

>> b = [6 8]'
b =
        6
        8

>> [x, err] = mp_solve_Axb(A, b)
x =
        4
        3

err =
        0

>> mp_multi(A, x)
ans =
```

```
        6
        8
```

As we can see, this result is the solution of $\mathbf{Ax} = \mathbf{b}$, but depends on parameters, a result can be a subsolution only, e.g.

```
% example with subsolution
>> b = [6 7]'
b =
        6
        7

>> [x, err] = mp_solve_Axb(A, b)
x =
        3
        2

err =
        1

>> mp_multi(A, x)
ans =
        5
        7
```

See also

mp_star, mp_solve_xAxb

---

mp_solve_xAxb

(max, +) solution of $\mathtt{x} = \mathtt{Ax} \oplus \mathtt{b}$

Syntax

```
x = mp_solve_xAxb(A, b)
```

Description

- It solves $\mathbf{x} = \mathtt{Ax} \oplus \mathtt{b}$ in the (max, +) algebra, i.e. $\mathtt{x} = \mathtt{A}^{\star}\mathtt{b}$.
- When there is no circuits with positive weight in the precedence graph $\mathcal{G}(\mathtt{A})$, then $\mathtt{x} = (\mathtt{A}^0 \oplus \mathtt{A}^1 \oplus \cdots \oplus \mathtt{A}^{(m-1)}) \otimes \mathtt{b}$, where m denotes the order of the square matrix A.

- `A` must be a square matrix.

Example

```
>> x = mp_solve_xAxb(−5, 2)
x =
        2

>> x == mp_add(p_multi(−5, x), 2)
ans =
        1

>> A = [mp_zero 2 3; −2 −10 −1; −5 −2 mp_one]
A =
     −Inf        2        3
       −2      −10       −1
       −5       −2        0

>> b = [1 2 3]'
b =
        1
        2
        3

>> x = mp_solve_xAxb(A, b)
x =
        6
        4
        3

>> x == mp_add(mp_multi(A, x), b)
ans =
        1
        1
        1
```

See also

mp_star, mp_solve_Axb

---

mp_star

(max, +) star operator

Syntax

```
B = mp_star(A)
[B, n] = mp_star(A)
```

Description

$\mathtt{B} = \mathtt{A}^\star = \mathtt{A}^0 \oplus \mathtt{A}^1 \oplus \ldots$

- It solves $\mathbf{x} = \mathtt{A}\mathbf{x} \oplus \mathbf{b}$ in the (max, +) algebra (i.e. $\mathbf{x} = \mathtt{A}^\star\mathbf{b}$).
- When there is no circuits with positive weight in the precedence graph $\mathcal{G}(\mathtt{A})$, then $\mathtt{A}^\star = \mathtt{A}^0 \oplus \mathtt{A}^1 \oplus \cdots \oplus \mathtt{A}^{(m-1)}$, where m denotes the order of the square matrix `A`.
- Function returns:
  - `star` $= \mathtt{A}^\star$
  - `n` — a minimal value for what all entries in $\mathtt{A}^{\mathtt{n}}$ are equal to ε.
- `A` must be a square matrix.

Example

```
>> mp_star(2)
ans =
      Inf
>> mp_star(-1)
ans =
        0

>> mp_star(mp_zero)
ans =
        0

>> mp_star(mp_one)
ans =
        0

>> mp_star(mp_zeros(2, 2))
ans =
        0     -Inf
     -Inf        0

>> [B, n] = mp_star([mp_zero mp_zero; 6 mp_zero])
B =
```

```
        0     -Inf
        6        0

n =
        2

>> A = [mp_zero 2 3; -2 -10 -1; -5 -2 mp_one]
A =
     -Inf        2        3
       -2      -10       -1
       -5       -2        0

>> [B, n] = mp_star(A)
B =
        0        2        3
       -2        0        1
       -4       -2        0

n =
      Inf

>> [B, n] = mp_star([2 3; mp_zero -1])
B =
      Inf      Inf
     -Inf        0

n =
      Inf
```

Lets find the minimal solution of the eq. (25):

```
>> a = -1; b = 2;
>> x = mp_multi(mp_star(a), b)
x =
        2

>> mp_add(mp_multi(a, x), b) == x
ans =
        1
```

for matrices:

```
>> A = [mp_zero   mp_one    mp_zero;
        mp_zero   mp_zero        -1;
        mp_one    mp_zero   mp_zero];
```

```
>> b = [     10;  mp_zero;  mp_zero];
>> x = mp_multi(mp_star(A), b)
x =
       10
        9
       10
```

Is it correct? Let us compare results from eqns. (25) and (26):

```
>> mp_add(mp_multi(A, x), b) == x
ans =
        1
        1
        1
```

See also

mp_solve_xAxb, mp_solve_Axb, mp_is_pga, mp_is_pgc, mp_is_pgsc1, mp_is_pgsc2, mp_mcm, mp_mcm_fw, mp_mcm_karp

---

mp_system

state fo an autonomous linear max–plus system

Syntax

```
X = mp_system(A, x0, k)
X = mp_system(A, x0, k, one_only)
```

Description

States of an autonomous linear max–plus system

$$x(k) = Ax(k-1),$$

where:

`A`	a (max, +) system matrix;
`k`	a cycle index (iteration number);
`x0`	an initial state vector, `x0 = x(0)`;
`one_only`	(optional, default `one_only = 1`) if is set to 1, the result is a state vector `x(k)`, if `one_only = 0`, the result is a matrix of a collection of state vectors from `x(0)` to `x(k)`.

Example

```
>> A = [3 7; 2 4]
A =
        3        7
        2        4

>> x0 = [1 0]'
x0 =
        1
        0

>> mp_system(A, x0, 8)
ans =
       25
       22

>> mp_system(A, x0, 5, 0)
ans =
        1        7       11       16       20       25
        0        4        9       13       18       22
```

See also

mp_multi

---

mp_trace

(max, +) trace of a matrix

Syntax

```
y = mp_trace(A)
```

Description

the (max, +) trace of matrix `A`, i.e.
the (max, +) sum of main diagonal of `A` (or equivalently a maximal element from main diagonal of `A`).

Example

```
>> A = [5 mp_zero 5; mp_zero 6 3; 11 12 11]
A =
        5     -Inf        5
```

```
    -Inf     6     3
      11    12    11

>> mp_trace(A)
ans =
      11
```

See also

mp_add

---

mp_zero

(max, +) zero ($-\infty$), i.e. ε, neutral element for operation $\oplus$

Syntax

```
y = mp_zero
```

Description

`mp_zero` returns `-Inf`

Example

```
>> mp_zero
ans =
    -Inf
```

See also

mp_zeros, mp_one, mp_ones, mp_eye, mp_add

---

mp_zeros

(max, +) zeros matrix, vector or scalar

Syntax

```
y = mp_zeros
y = mp_zeros(n)
y = mp_zeros(n, m)
```

Description

- `mp_zeros` returns ε, i.e. `-Inf`.
- `mp_zeros(n)` or `mp_zeros([n])` returns an `n`–by–`n` matrix of ε–s.
- `mp_zeros(n, m)` or `mp_zeros([n m])` returns an `n`–by–`m` matrix of ε–s.

Example

```
>> mp_zeros
ans =
     -Inf

>> mp_zeros(2)
ans =
     -Inf     -Inf
     -Inf     -Inf

>> mp_zeros(2, 3)
ans =
     -Inf     -Inf     -Inf
     -Inf     -Inf     -Inf
```

See also

mp_zero, mp_one, mp_ones, mp_eye, mp_add

---

mpm_add

(min, +) addition of scalars, vectors or matrices

Syntax

`y = mpm_sum(n,m)`

Description

`y` = `n` $\vee$ `m`

- If `n` and `m` are scalars, result is a (min, +) sum of `n` and `m`.
- If `n` (or `m`) is scalar and `m` (or `n`) is vector, result is a vector the same size as `m` (or `n`) where for every entries is (min, +) added `n` (or `m`).

- If `n` (or `m`) is scalar and `m` (or `n`) is matrix, result is a matrix the same size as `m` (or `n`) where for every entries is (min, +) added `n` (or `m`).
- If `n` and `m` are this same size matrices, result is an array the same size as `n` (and `m`) with the entries equal to (min, +) addition elements from `n` and `m`.

Example

```
>> mpm_add(-5, 7)
ans =
      -5

>> mpm_add(3, [mpm_zero 4])
ans =
        3        3
>> A = [1 6; 8 3], B = [2 5; 3 3]
A =
        1        6
        8        3

B =
        2        5
        3        3

>> ans =

        1        5
        3        3
```

See also

mpm_multi, mpm_one, mpm_ones, mpm_zero, mpm_zeros

---

mpm_div

(min, +) division

Syntax

```
Z = mpm_div(A, B)
[Z, err] = mpm_div(A, B)
```

Description

`Z = A ⊘ B`

- If `A` is a scalar
  - if `B` is a scalar, result is a scalar: (min, +) division of `A` by `B`;
  - if `B` is a vector (or matrix), result is a vector (or a matrix) where every entries are: `A` divided by apropriate entry of `B`.
- If `A` is a vector (or a matrix)
  - if `B` is a scalar, result is a vector (or a matrix) the same size as `A` where every entries of `A` are (min, +) divided by `B`;
  - if `B` is a vector (or a matrix) the same size as `A`: the result is (min, +) division `A` by `B`, i.e. `mpm_multi(mpm_inv(B), A)`, and when `B` is (min, +)-invertable `err = 0`, otherwise `err = 1`;
  - if `B` is a vector (or a matrix) different size than `A` — operation is not defined.
- Division by (min, +) zero (`Inf`) is not defined - and returns `NaN`.

Example

```
>> mpm_div(3, 3)
ans =
        0

>> mpm_div(3, mpm_one)
ans =
        3

>> mpm_div(3, mpm_zero)
ans =
      NaN

>> mpm_div([mpm_zero 7 mpm_one], 3)
ans =
      Inf        4       -3

>> mpm_div(3, [mpm_zero 7 mpm_one])
ans =
      NaN       -4        3

>> A = [mpm_zero         1  mpm_zero;
```

```
               2  mpm_zero  mpm_zero;
       mpm_zero  mpm_zero         3];

>> B = [       3  mpm_zero  mpm_zero;
       mpm_zero  mpm_zero         4;
       mpm_zero         5  mpm_zero];

>> [C, err] = mpm_div(A, B)
C =
      Inf       -2      Inf
      Inf      Inf       -2
       -2      Inf      Inf

err = 0

>> mpm_multi(B, C)
ans =
      Inf        1      Inf
        2      Inf      Inf
      Inf      Inf        3
```

See also

mpm_inv, mpm_multi, mpm_one, mpm_zero

---

mpm_eye

(min, +) identity matrix

Syntax

```
Y = mpm_eye
Y = mpm_eye(n)
Y = mpm_eye(n, m)
```

Description

- `mpm_eye` returns 0.
- `mpm_eye(n)` or `mpm_eye([n])` returns an `n`–by–`n` (min, +) identity matrix with 0–s on the main diagonal and `Inf`'s elsewhere.
- `mpm_eye(n, m)` or `mpm_eye([n m])` returns an `n`–by–`m` (min, +) identity matrix.

Example

```
>> mpm_eye
ans =
        0

>> mpm_eye(2)
ans =
        0      Inf
      Inf        0

>> mpm_eye(2, 3)
ans =
        0      Inf      Inf
      Inf        0      Inf
```

See also

mpm_one, mpm_ones, mpm_zero, mpm_zeros, mpm_randi

---

mpm_inv

(min, +) matrix invertion

Syntax

```
Y = mpm_inv(A)
[Y, err] = mpm_inv(A)
```

Description

$Y = A^{-1}$

- If A is a scalar then Y = mpm_power(A, -1) and err = 0.
- If A is a square matrix then $Y = A^{-1}$ and if AY = YA = I then err = 0, otherwise err = 1.
- If A is not a square matrix then Y = (I-A)' and if AY = I then err = 0, otherwise err = 1.

Example

Let us consider a square matrix:

```
>> mpm_inv(5)
ans =
       -5

>> A = [mpm_zero         1  mpm_zero;
               2  mpm_zero  mpm_zero;
        mpm_zero  mpm_zero         3];

>> [Y, err] = mpm_inv(A)
Y =
      Inf       -2      Inf
       -1      Inf      Inf
      Inf      Inf       -3

err =
        0

>> A = [mpm_zero         1  mpm_zero;
               2  mpm_zero  mpm_zero];

>> [Y, err] = mpm_inv(A)
Y =
      Inf       -2
       -1      Inf
      Inf      Inf
err =
        0

>> mpm_multi(A, Y)
ans =
        0      Inf
      Inf        0
```

See also

mpm_power, mpm_multi, mpm_div, mpm_eye

---

mpm_multi

(min, +) multiplication of scalars, vectors or matrices

Syntax

```
y = mpm_multi(m, n)
```

Description

$\mathtt{y} = \mathtt{n} \wedge \mathtt{m}$

- If `n` and `m` are scalars, result is a (min, +) multiplication of n and m.
- If `n` (or `m`) is scalar and `m` (or `n`) is vector, result is a vector the same size as `m` (or `n`) where for every entries is (min, +) multiplication by `n` (or `m`).
- If `n` (or `m`) is scalar and `m` (or `n`) is matrix, result is a matrix the same size as `m` (or `n`) where for every entries is (min, +) multiplication by `n` (or `m`).
- If `n` is an $p \times q$ matrix and `m` is an $q \times r$ matrix result is a (min, +) product `n` and `m` returns an $p \times r$ matrix.

Example

```
>> mpm_multi(-5, 7)
ans =
        2

>> mpm_multi(3, [mpm_zero 4])
ans =
      Inf        7

>> A = [1 6 2; 8 3 4],  B = [2 5; 3 3; 1 6]
A =
        1        6        2
        8        3        4

B =
        2        5
        3        3
        1        6

mpm_multi(A, B)
ans =
        3        6
        5        6
```

See also

mpm_add, mpm_div, mpm_inv, mpm_power, mpm_one, mpm_ones, mpm_zero, mpm_zeros

---

mpm_mx2latex

matrix conversion from the Matlab (min, +) description to the LaTeX

Syntax

```
mpm_mx2latex(X)
mpm_mx2latex(X, 'fileName')
```

Description

(min, +) matrix (or vector) conversion from the Matlab®/Octave notation to the LaTeX source code.

- `X` is a matrix (or vector) to conversion;
- `fileName` (optional) is a name of file in which a LaTeX source code of `X` will be saved, default it is `mpm_mx.tex`.

Example

```
>> A = [ 0     3   Inf     1
         1     2     2  -Inf
      -Inf   Inf     1     0];

>> mpm_mx2latex(A, 'A.tex')
```

LaTeX source of A.tex:

```
\left[
\begin{array}{*{20}c}
0 & 3 & \varepsilon & 1 \\
1 & 2 & 2 & -\infty \\
-\infty & \varepsilon & 1 & 0 \\
\end{array}
\right]
```

LaTeX result:

$$\left[\begin{array}{*{20}c} 0 & 3 & \varepsilon & 1 \\ 1 & 2 & 2 & -\infty \\ -\infty & \varepsilon & 1 & 0 \\ \end{array}\right]$$

See also

mpm_mx2latex, mp_conv

---

mpm_one

(min, +) unit (0), neutral element for oprtation $\wedge$

Syntax

```
y = mpm_one
```

Description

`mpm_one` returns 0.

Example

```
>> mpm_one
ans =
        0
```

See also

mpm_ones, mpm_zero, mpm_zeros, mpm_eye, mpm_multi

---

mpm_ones

(min, +) ones (0's) matrix, vector or scalar

Syntax

```
y = mpm_ones
y = mpm_ones(n)
y = mpm_ones(n,m)
```

Description

- `mpm_ones` returns 0.
- `mpm_ones(n)` or `mpm_ones([n])` is an `n`–by–`n` matrix of 0–s.
- `mpm_ones(n,m)` or `mpm_ones([n,m])` is an `n`–by–`m` matrix of 0–s.

Example

```
>> mpm_ones
ans =
         0

>> mpm_ones(2)
ans =
        0        0
        0        0

>> mpm_ones(2, 3)
ans =
        0        0        0
        0        0        0
```

See also

mpm_one, mpm_zero, mpm_zeros, mpm_eye, mpm_multi

---

mpm_plus

(min, +) plus operator
shortest path matrix

Syntax

```
Y = mpm_plus(A)
[Y, n] = mpm_plus(A)
```

Description

$$\mathtt{Y} \quad = \quad \mathtt{A}^{+} = \mathtt{A}^{1} \vee \mathtt{A}^{2} \vee \ldots$$
$$\mathtt{A}^{+} \vee \mathtt{A}^{0} = \mathtt{A}^{\star}$$

When there is no circuits with negative weight in the precedence graph $\mathcal{G}(\mathtt{A})$, then

$$\mathtt{A}^{+} = \mathtt{A}^{1} \vee \mathtt{A}^{2} \vee \cdots \vee \mathtt{A}^{(m-1)}$$

where m denotes the order of the square matrix A.

Function returns:

- `Y` = A^{+}
- `n` — a minimal value for what all entries in A^{n} are equal to ε.

`A` must be a square matrix.

Example

```
>> mpm_plus(2)
ans =
        2

>> mpm_add(mpm_one, mpm_plus(2)) == mpm_star(2)
ans =
        1
```

Let us consider a shortest path problem — example 3.3.

```
>>A=[mpm_zero mpm_zero mpm_zero mpm_zero mpm_zero mpm_zero;
     5        mpm_zero mpm_zero mpm_zero mpm_zero mpm_zero;
     3        mpm_zero mpm_zero mpm_zero mpm_zero mpm_zero;
     mpm_zero 2        mpm_zero mpm_zero 5        mpm_zero;
     mpm_zero 1        4        mpm_zero mpm_zero mpm_zero;
     mpm_zero mpm_zero 8        2        4        mpm_zero]

>> mpm_plus(A)
ans =
      Inf      Inf      Inf      Inf      Inf      Inf
        5      Inf      Inf      Inf      Inf      Inf
        3      Inf      Inf      Inf      Inf      Inf
        7        2        9      Inf        5      Inf
        6        1        4      Inf      Inf      Inf
        9        4        8        2        4      Inf
```

See also

mpm_star, mp_is_pga, mp_is_pgc

---

mpm_power

(min, +) raising to a power of scalar or a square matrix

Syntax

```
y = mpm_power(x, n)
Y = mpm_power(X, n)
```

Description

(min, +) `n`–th power of `x`, $\mathtt{y} = \mathtt{x}^n$
(min, +) `n`–th power of square matrix `X`, $\mathtt{Y} = \mathtt{X}^n$

- If `x` is a scalar than:
  - $\mathtt{x} \in \mathbb{R}_\varepsilon, \mathtt{n} \in \mathbb{R} : \mathtt{x}^n = \mathtt{x} \times \mathtt{n}$ (in conventional algebra).
  - If $\mathtt{n} = 0$ then $x^0 = 0$.
  - If $\mathtt{x} = \varepsilon$ and $\mathtt{n} > 0$ then $\varepsilon^n = \varepsilon$.
  - If $\mathtt{x} = \varepsilon$ and $\mathtt{n} < 0$ then ε^n is not defined.
  - If $\mathtt{x} = \varepsilon$ and $\mathtt{n} < 0$ then $\varepsilon^0 = 0$ by definition.
- If `n` is a vector (or a matrix) then result is a vector (or a matrix) this same size, where every entries is (min, +) power of `x` to the proper element from `n`.
- If `X` is a square matrix then `n` must belong to $\mathbb{N}_0 \cup \{-1\}$,
  - If $\mathtt{n} \notin \mathbb{N}_0 \cup \{-1\}$ operation is not defined.
- If `X` is not a scalar nor square matrix operation is not defined.

Example

```
>> mpm_power(3, 3)
ans =
        9

>> mpm_power(3, -2)
ans =
       -6

>> mpm_power(3, 1/4)
ans =
   0.7500

>> mpm_power(2, [1/4 1 4 -4])
ans =
   0.5000   2.0000   8.0000  -8.0000

>> mpm_power([1 6; 8 3], 3)
ans =
        3        8
       10        9
```

See also

mpm_multi, mpm_inv, mpm_div, mpm_eye, mpm_one, mpm_zero

---

mpm_star

(min, +) star operator

Syntax

```
B = mpm_star(A)
[B, n] = mpm_star(A)
```

Description

$\mathtt{B} = \mathtt{A}^{\star} = \mathtt{A}^{0} \vee \mathtt{A}^{+} = \mathtt{A}^{0} \vee \mathtt{A}^{1} \vee \dots$ It solves $\mathbf{x} = \mathtt{A}\mathbf{x} \vee \mathbf{b}$ in the (min, +) algebra (i.e. $\mathbf{x} = \mathtt{A}^{\star}\mathbf{b}$).
When there is no circuits with negative weight in the precedence graph $\mathcal{G}(\mathtt{A})$, then

$$\mathtt{A}^{\star} = \mathtt{A}^{0} \vee \mathtt{A}^{1} \vee \cdots \vee \mathtt{A}^{(m-1)}$$

where m denotes the order of the square matrix A.

Function returns:

- $\mathtt{B} = \mathtt{A}^{\star}$
- n — a minimal value for what all entries in $\mathtt{A}^{\mathtt{n}}$ are equal to ε.

A must be a square matrix.

Example

```
>> mpm_star(2)
ans =
        0

>> mpm_star(-1)
ans =
     -Inf

>> mpm_star(mpm_zero)
ans =
        0
```

```
>> mpm_star(mpm_one)
ans =
        0

>> mpm_star(mpm_zeros(2, 2))
ans =
        0      Inf
      Inf        0

>> [B, n] = mpm_star([mpm_zero mpm_zero; 6 mpm_zero])
B =
        0      Inf
        6        0
n =
        2

>> [B, n] = mpm_star([mpm_zero −2 −3; 2 10 1; 5 2 mpm_one])
B =
        0       −2       −3
        2        0       −1
        4        2        0
n =
      Inf

>> [B, n] = mpm_star([2 3; mpm_zero −1])
B =
        0      Inf
      Inf     −Inf
n =
      Inf
```

Lets find the maximal solution of $\mathbf{x} = \mathtt{A}\mathbf{x} \vee \mathbf{b}$

```
>> a = 1; b = 2;
>> x = mpm_multi(mpm_star(a), b)
x =
        2

>> mpm_add(mpm_multi(a, x), b) == x
ans =
        1
```

for matrices:

```
>> A = [mpm_zero  mpm_one    mpm_zero;
        mpm_zero  mpm_zero   1;
        mpm_one   mpm_zero   mpm_zero];
>> b = [10;       mpm_zero;  mpm_zero];
>> x = mpm_multi(mpm_star(A), b)
x =
       10
       11
       10
```

Is it correct?

```
>> mpm_add(mpm_multi(A, x), b) == x
ans =
        1
        1
        1
```

See also

mpm_plus, mp_is_pga, mp_is_pgc

---

mpm_zero

(min, +) zero ($-\infty$)
(min, +) neutral element for operation $\vee$

Syntax

```
y = mpm_zero
```

Description

`mpm_zero` returns `Inf`.

Example

```
>> mpm_zero
ans =
      Inf
```

See also

mpm_zeros, mpm_one, mpm_ones, mpm_eye, mpm_add

---

mpm_zeros

(min, +) zeros matrix, vector or scalar

Syntax

```
y = mpm_zeros
y = mpm_zeros(n)
y = mpm_zeros(n, m)
```

Description

- `mpm_zeros` returns `Inf`.
- `mpm_zeros(n)` or `mpm_zeros([n])` returns an `n`–by–`n` matrix of `Inf`-s.
- `mpm_zeros(n, m)` or `mpm_zeros([n m])` returns an `n`–by–`m` matrix of `Inf`-s.

Example

```
>> mpm_zeros
ans =
      Inf

>> mpm_zeros(2)
ans =
      Inf      Inf
      Inf      Inf

>> mpm_zeros(2, 3)
ans =
      Inf      Inf      Inf
      Inf      Inf      Inf
```

See also

mpm_zero, mpm_one, mpm_ones, mpm_eye, mpm_add

Bibliography

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin (1993). *Network Flows. Theory, Algorithms, and Applications*. Prentice Hall. ISBN: 0-13-617549-X.

Aldous, Joan M. and Robin J. Wilson (2000). *Graphs and Application. An Introductory Approach*. Springer-Verlag. ISBN: 1-85233-259-X.

Baccelli, Francois et al. (1992). *Synchronisation and Linearity. An Algebra for Discrete Event Systems*. Wiley. ISBN: 0-471-93609-X. URL: http://www-rocq.inria.fr/metalau/cohen/SED/book-online.html.

Bapat, R.B. (1998). "A max version of the Perron–Frobenius theorem". In: *Linear Algebra and its Applications* 275–276, pp. 3–18.

Blyth, T.S. and M.F. Janowitz (1972). *Residuation Theory*. Pergamon Press.

Braker, Johannes G. (1993). "Algorithms and Applications in Timed Discrete Event Systems". PhD Thesis. Department of Technical Mathematics and Informatics, Delft University of Technology. ISBN: 90-9006669-1. URL: http://repository.tudelft.nl/view/ir/uuid%3Adf4c3874-1255-485f-a3d8-a0e1bf123914/.

Braker, Johannes G. and Geert-Jan Olsder (1993). "The Power Algorithm in Max Algebra". In: *Linear Algebra and its Applications* 182, pp. 67–89. DOI: 10.1016/0024-3795(93)90492-7.

Cassandras, Christos G. and Stéphane Lafortune (2007). *Introduction to Discrete Event Systems*. Second Edition. Springer.

Cohen, Guy, Stéphane Gaubert, and Jean-Pierre Quadrat (1999). "Max–plus algebra and system theory. Where we are and where to go now". In: *Annual Reviews in Control* 23, pp. 207–219.

Cuninghame–Green, Raymond (1979). *Minimax Algebra*. Vol. 166. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag. ISBN: 3-540-09113-0.

Dasdan, Ali and Rajesh K. Gupta (1998). "Faster Maximum and Minimum Mean Cycle Algorithms for System Performance Analysis". In: *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems* 17, pp. 889–899.

De Schutter, Bart (1996). "Max-Algebraic System Theory for Discrete Event Systems". PhD Thesis. Leuven, Belgium: Faculty of Applied Sciences, K.U.Leuven. URL: http://www.dcsc.tudelft.nl/~bdeschutter/pub/rep/phd.pdf.

De Schutter, Bart and Ton van den Boom (2008). "Max-plus algebra and max-plus linear discrete event systems: an introduction". In: *Proceedings of the 9th International Workshop on Discrete Event Systems (WODES'08)*, pp. 36–42.

Floyd, Robert W (1962). "Algorithm 97: Shortest Path". In: *Communications of the ACM* 5, p. 345. DOI: 10.1145/367766.368168.

Gaubert, Stéphane (1992). *MAX: a Maple package for the (max,+) algebra*. URL: http://www.cmap.polytechnique.fr/~gaubert/PAPERS/MAX.html.

— (1997). *Methods and Applications of (max, +) Linear Algebra*. Research Report RR-3088. INRIA. URL: https://hal.inria.fr/inria-00073603.

Gaubert, Stéphane and Max P. Scilab (1998). "Max–plus Linear Algebra with Scialb". In: *ALAPEDES Max–Plus Software Workshop*. INRIA. URL: http://www.cmap.polytechnique.fr/~gaubert/TPALGLIN.pdf.

Gross, Donald et al. (2008). *Fundamentals of Queueing Theory*. Fourth Edition. Wiley.

Gruet, B. et al. (2015). *C++ MinMax library*. URL: http://perso-laris.univ-angers.fr/~hardouin/outils.html.

Heidergott, B. and R. de Vries (2001). "Towards a (max,+) Control Theory for Public Transportation Networks". In: *Discrete Event Dynamic Systems* 11.4, pp. 371–398.

Heidergott, Bernd, Geert Jan Olsder, and Jacob van der Woude (2005). *Max Plus at Work: Modeling and Analysis of Synchronized Systems. A Course on Max-Plus Algebra and Its Applications*. Princeton University Press.

Karp, Richard M. (1978). "A characterization of the minimum cycle mean in a digraph". In: *Discrete Mathematics* 23, pp. 309–311.

Komenda, J., A. El Moudni, and N. Zerhouni (2001). "Input-Output Relation and Time-Optimal Control of a Class of Hybrid Petri Nets Using (min,+) Semiring". In: *Discrete Event Dynamic Systems* 11.1, pp. 59–75.

Limnios, Nikolaos and G Oprişan (2013). *Semi–Markov Processes and Reliability*. Springer.

Maia, C.A. et al. (2003). "Optimal Closed-Loop Control of Timed Event Graphs in Dioids". In: *IEEE Transactions on Automatic Control* 48.12, pp. 2284–2287.

MaxPlus Working Group, INRIA (2003). *MaxPlus Toolbox for Scilab*. URL: http://www.cmap.polytechnique.fr/~gaubert/.

Menguy, E. et al. (2000). "A First Step Towards Adaptive Control for Linear Systems in Max Algebra". In: *Discrete Event Dynamic Systems* 10.4, pp. 347–367.

Murata, Tadao (1989). "Petri Nets: Properties, Analysis and Applications". In: *Proceedings of the IEEE* 77, pp. 541–580.

Olsder, Geert-Jan (1991). "Eigenvalues of dynamic max-min systems". In: *Discrete Event Dynamic Systems* 1, pp. 177–207. DOI: 10.1007/BF01805562.

Olsder, Geert-Jan, Kees Roos, and Robert-Jan van Egmond (1999). "An efficient algorithm for critical circuits and finite eigenvectors in the max–plus algebra". In: *Linear Algebra and its Applications* 295, pp. 231–240.

Ramadge, Peter J. and W. Murray Wonham (1989). "The control of discrete event systems". In: *Proceedings of the IEEE* 77, pp. 81–98.

Stańczyk, Jarosław (2016). *Max–Plus Algebra Toolbox for Matlab*. Version 1.7. URL: http://gen.up.wroc.pl/stanczyk/mpa/.

Stańczyk, Jarosław, Eckart Mayer, and Jörg Raisch (2004). "Modelling and performance evaluation of DES — a Max–Plus Algebra Toolbox for Matlab". In: 1^{st} Int. Conf. *Informatics in Control, Automation and Robotics, ICINCO'04*. Vol. 3. Setúbal, Portugal, pp. 270–275. DOI: 10.5220/0001132102700275.

Subiono and Jacob van der Woude (2000). "Power Algorithms for (max, +)- and Bipartite (min, max, +)-Systems". In: *Discrete Event Dynamic Systems* 10.4, pp. 369–389. DOI: 10.1023/A:1008315821604.

Tarjan, Robert E. (1972). "Depth–first search and linear graph algorithms". In: *SIAM J. Comput.* 1, pp. 146–160.

van den Boom, T. and B. De Schutter (2002). "Properties of MPC for Max–Plus–Linear Systems". In: *European Journal of Control* 8.5, pp. 453–462.

van den Boom, Ton J.J. et al. (2003). "Adaptive model predictive control using max-plus-linear input-output models". In: *Proceedings of the American Control Conference* 2, pp. 933–938. DOI: 10.1109/ACC.2003.1239706.

Notation

Here we list some of the acronyms and symbols that occur frequently in this contribution and with which the reader might not be familiar. The numbers in the last column refer to the page on which the symbol or concept is defined.

Acronyms

DES	Discrete Event System	

Sets

$\mathbb{N}$	set of the natural numbers, $\mathbb{N} = \{1, 2, 3, \dots\}$	
$\mathbb{N}_0$	set of the nonnegative integers	
$\mathbb{R}$	set of the real numbers	
$\mathbb{R}_\varepsilon$	$\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$	
	in the (max, +) $\varepsilon = -\infty$	9
	in the (min, +) $\varepsilon = +\infty$	30
$\mathbb{R}_\varepsilon^n$	set of the column vectors with n components in $\mathbb{R}_\varepsilon$, i.e. $\mathbb{R}_\varepsilon^n \equiv \mathbb{R}_\varepsilon^{n\times 1}$	
$\mathbb{R}_\varepsilon^{m\times n}$	set of the m–by–n matrices with entries in $\mathbb{R}_\varepsilon$	

Matrices and Vectors

$\mathbf{b}$	column vector	
$(\mathbf{b})_i$	i–th element of the column vector $\mathbf{b}$	
$\mathbf{A}^T$	transpose of the matrix $\mathbf{A}$	
$(\mathbf{A})_{ij}$	entry of the matrix $\mathbf{A}$ on the i–th row and the j–th column	
$\mathbf{I}_n$	(max, +) identity matrix, $\mathbf{I}_n \in \mathbb{R}_\varepsilon^{n\times n}$	??
$\boldsymbol{\varepsilon}^{m\times n}$	m–by–n (max, +) zero matrix	??

(max, +) Algebra

$\oplus$	(max, +) algebraic addition	9, ??
$\otimes$	(max, +) algebraic multiplication	9, ??

$\oslash$	(max, +) algebraic division	**??**, 15
e	neutral element for $\otimes : e = 0$	9
ε	neutral element for $\oplus : \varepsilon = -\infty$	9
$\mathbf{A}^\star$	(max, +) star operator (for square matrix), $\mathbf{A}^\star = \mathbf{I} \oplus \mathbf{A}^1 \oplus \mathbf{A}^2 \oplus \dots$	21
$\mathbb{R}_{\max}$	(max, +) algebra: $\mathbb{R}_{\max} = (\mathbb{R}_\varepsilon, \oplus, \otimes)$	9
$\mathbb{R}_\varepsilon^n$	$\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$	9

(min, +) Algebra

$\vee$	(min, +) algebraic addition	30
$\wedge$	(min, +) algebraic multiplication	30
e	neutral element for $\wedge : e = 0$	30
ε	neutral element for $\vee : \varepsilon = +\infty$	30
$\mathbf{A}^+$	shortest path matrix (min, +) plus operator (for square matrix), $\mathbf{A}^+ = \mathbf{A}^1 \vee \mathbf{A}^2 \vee \dots$	30
$\mathbf{A}^\star$	(min, +) star operator (for square matrix), $\mathbf{A}^\star = \mathbf{I} \vee \mathbf{A}^+ = \mathbf{I} \vee \mathbf{A}^1 \vee \mathbf{A}^2 \vee \dots$	
$\mathbb{R}_{\min}$	(min, +) algebra: $\mathbb{R}_{\min} = (\mathbb{R}_\varepsilon, \vee, \wedge)$	30
$\mathbb{R}_\varepsilon^n$	$\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$	30

Miscellaneous

$\mathcal{G}(\mathbf{A})$	precedence graph of the matrix $\mathbf{A}$	17
$tr(\mathbf{A})$	trace of the matrix $\mathbf{A}$	15
λ	maximal cycle mean of $\mathcal{G}(\mathbf{A})$ (largest) eigenvalue of $\mathbf{A}$	18 21

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the

text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in

the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this

License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

GNU Affero General Public License

Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU Affero General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

(a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

(b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy

of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place

additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work

the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to

downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public Li-

cense.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN

OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the

exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see http://www.gnu.org/licenses/.

Index