MAXPLUS LINEAR SYSTEMS IN SCILAB AND APPLICATION TO PRODUCTION SYSTEMS

G. COHEN, S. GAUBERT, AND J.P. QUADRAT

ABSTRACT. We present a session showing the current functionality of the max-plus toolbox of Scilab. The available max-plus types and the overloading of the standard arithmetic operators are shown. On a flowshop example coming from a real production application, we illustrate the usefulness of such kind of toolboxes.

CONTENTS

The maxplus toolbox is an external contribution of Scilab which must be installed and loaded. For that we have to :

- create a directory called “contrib” at the first level of scilab,
- copy inside this directory the directory ‘ ‘maxplus1”,
- enter at the first level of the maxplus1 directory,
- execute the linux “make” command at the first level of the maxplus1 directory,
- enter in Scilab
- load the maxplus toolbox in scilab (by the instruction “exec contrib/maxplus1/mploader.sce”) available at the first level of the maxplus1 directory.

1. MAXPLUS SCALAR AND MATRICES

Let us show first the max-plus object of Scilab and its interaction with standard objects.

The standard real matrices have the internal type number 1.

```
-->a=2
 a  =
    2.
-->type(a)
 ans  =
    1.
```

A maxplus matrix is created by the instruction maxplus which has the abreviation #. The maxplus type number of full maxplus matrices is 257.

```
-->b=#(3)
 b  =
 3
-->type(b)
 ans  =
    257.
```

We can change a maxplus matrix in a standard matrix by the instruction plustimes.

```
-->plustimes(b)
 ans  =
    3.
-->type(ans)
 ans  =
    1.
```

The maxplus zero is $-\infty$ is printed with the character dot.

```
-->%0
 %0  =
-->type(%0)
 ans  =
    257.
-->%inf
 %inf  =
   Inf
-->type(%inf)
 ans  =
    1.
```

The maxplus unity is equal to O.

```
-->%1
 %1  =
 0
```

The maxplus operations overload the standard operations.

```
-->b
 b  =
 3
-->type(b)
 ans  =
    257.
-->b + %0
 ans  =
 3
-->b * %1
 ans  =
 3
-->b + b
 ans  =
```

```
 3
-->b * b
 ans  =
 6
-->b / b
 ans  =
 0
-->b & %0
 ans  =
-->b == b
 ans  =
  T
-->b <> b
 ans  =
  F
-->b >= b
 ans  =
  T
-->b > b
 ans  =
  F
```

In general (at least when the first argument is of maxplus type) the maxplus type dominate the standard type. The semantic is still not completely fixed.

```
-->b+3
 ans  =
 3
-->type(ans)
 ans  =
    257.
-->b*3
 ans  =
 6
```

We have different way to create maxplus matrices :

- from a max-plus scalar

  ```
  -->c=[b,4;5,6]
   c  =
  !3  4  !
  !      !
  !5  6  !
  -->type(c)
   ans  =
      257.
  ```

- from a standard matrix

  ```
  -->d=[1,2;3,4]
   d  =
  !   1.    2. !
  !   3.    4. !
  ```

```
-->type(d)
 ans  =
    1.
-->e=#(d)
 e  =
!1  2  !
!      !
!3  4  !
-->type(e)
 ans  =
    257.
```

- by extraction

```
-->f=e(1,:)
 f  =
!1  2  !
-->type(f)
 ans  =
    257.
```

- by insertion

```
-->e(5,5)=6
 e  =
!1  2  .  .  .  !
!               !
!3  4  .  .  .  !
!               !
!.  .  .  .  .  !
!               !
!.  .  .  .  .  !
!               !
!.  .  .  .  6  !
-->type(e)
 ans  =
    257.
```

There are special instructions to create important particular maxplus matrices.

```
-->%ones(2,5)
 ans  =
!0  0  0  0  0  !
!               !
!0  0  0  0  0  !
-->%eye(2,5)
 ans  =
!0  .  .  .  .  !
!               !
!.  0  .  .  .  !
-->g=%zeros(2,5)
 g  =
(    2,    5) zero sparse matrix
```

There exists sparce maxplus matrices.

```
-->type(g)
 ans  =

    261.
```

We can change a sparse matrix in a full one.

```
-->full(g)
 ans  =
!.  .  .  .  .  !
!               !
!.  .  .  .  .  !
-->type(ans)
 ans  =

    257.
```

We can change a full matrix in a sparse one.

```
-->%ones(2,5)
 ans  =
!0  0  0  0  0  !
!               !
!0  0  0  0  0  !
-->sparse(ans)
 ans  =
(    2,    5) sparse matrix

(    1,    1)        0.
(    1,    2)        0.
(    1,    3)        0.
(    1,    4)        0.
(    1,    5)        0.
(    2,    1)        0.
(    2,    2)        0.
(    2,    3)        0.
(    2,    4)        0.
(    2,    5)        0.
-->type(ans)
 ans  =
    261.
```

The standard operations on matrices are overloaded (be careful with & which means here min element wise).

```
-->c
 c  =
!3  4  !
!      !
!5  6  !
-->d
 d  =
```

```
!   1.    2. !
!   3.    4. !
-->c + d
 ans  =
!3  4  !
!      !
!5  6  !
-->c * c
 ans  =
!9   10  !
!        !
!11  12  !
-->c / c
 ans  =
!0  -2  !
!       !
!2  0   !
-->d & c
 ans  =
!   1.    2. !
!   3.    4. !
-->star(c)
 ans  =
!Inf  Inf  !
!          !
!Inf  Inf  !
-->c == c
 ans  =

! T T !
! T T !
-->c <> c
 ans  =
! F F !
! F F !
-->d > c
 ans  =
! F F !
! F F !
```

The standard scilab column concatenation is overloaded.

```
-->h=[e,e]
 h  =
!1  2  .  .  .  1  2  .  .  .  !
!                              !
!3  4  .  .  .  3  4  .  .  .  !
!                              !
!.  .  .  .  .  .  .  .  .  .  !
!                              !
```

```
!.  .  .  .  .  .  .  .  .  .  !
!                              !
!.  .  .  .  6  .  .  .  .  6  !
```

The row concatenation is overloaded.

```
-->i=[e;e]
 i  =
!1  2  .  .  .  !
!               !
!3  4  .  .  .  !
!               !
!.  .  .  .  .  !
!               !
!.  .  .  .  .  !
!               !
!.  .  .  .  6  !
!               !
!1  2  .  .  .  !
!               !
!3  4  .  .  .  !
!               !
!.  .  .  .  .  !
!               !
!.  .  .  .  .  !
!               !
!.  .  .  .  6  !
-->size(i)
 ans  =
!   10.    5. !
```

The standard extraction is overloaded.

```
-->i([1,3],:)
 ans  =
!1  2  .  .  .  !
!               !
!.  .  .  .  .  !
```

Spectral elements of a matrix can be computed efficiently by the Howard algorithm.

```
-->c
 c  =
!3  4  !
!      !
!5  6  !
-->[chi,v]=howard(c)
 v  =
!4  !
!   !
!6  !
 chi  =
!6  !
```

```
!    !
!6   !
```

chi is the eigenvalue, v is the eigenvector

```
-->chi(1)*v==c*v
 ans  =
! T !
! T !
```

These spectral elements give the asymptotic behaviour of maxplus dynamical system.

```
-->x=[%1;%0]
 x  =
!0   !
!    !
!.   !
-->[x,c*x,c*c*x,c*c*c*x,c*c*c*c*x]
 ans  =
!0  3  9   15  21  !
!                  !
!.  5  11  17  23  !
```

Howard is a ”linear” algorithm with the number of arcs in practise.

```
-->timer();
-->[chi,v]=howard(#(sprand(10000,10000,0.0005)+..
0.001*speye(10000,10000)));
-->timer()
 ans  =
    3.32
-->chi
 chi  =
!0.9173857  !
!           !
!0.9173857  !
!           !
!0.9173857  !
!           !
!0.9173857  !
!           !
!0.9173857  !
!           !
```

2. State space representation of maxplus linear systems

Dynamical system in implicit state form

$$X(n) = DX(n) + AX(n-1) + BU(n), \quad Y(n) = CX(n)$$

can be manipulated with the standard Scilab operators which are once more over loaded.

Let us create first max-plus dynamical linear systems.

```
-->s1=mpsyslin([1,2;3,4],[0;0],[0,0],%eye(2,2))
 s1  =
    | 0  . |    | 1  2 |    | 0 |
x = | .  0 |x + | 3  4 |x'+ | 0 |u

y = | 0  0 |x
-->s1('D')
 ans  =
!0  .  !
!      !
!.  0  !
-->s1('X0')
 ans  =
(    2,    1) zero sparse matrix

-->s1=full(s1)
 s1  =
    | 0  . |    | 1  2 |    | 0 |
x = | .  0 |x + | 3  4 |x'+ | 0 |u

y = | 0  0 |x
-->s1('X0')
 ans  =
!.  !
!   !
!.  !
-->explicit(s1)
 ans  =
    | 1  2 |    | 0 |
x = | 3  4 |x'+ | 0 |u

y = | 0  0 |x
-->s2=mpsyslin([1,2,3;4,5,6;7,8,9],[0;0;0],[0,0,0],%eye(3,3))
 s2  =
    | 0  .  . |    | 1  2  3 |    | 0 |
x = | .  0  . |x + | 4  5  6 |x'+ | 0 |u
    | .  .  0 |    | 7  8  9 |    | 0 |

y = | 0  0  0 |x
-->s2=sparse(s2);
```

The maxplus linear system operators have the same syntax as the matrix ones.

- Diagonal composition

```
-->s4=s1|s2
 s4  =
    | 0  .  .  .  . |    | 1  2  .  .  . |    | 0  . |
    | .  0  .  .  . |    | 3  4  .  .  . |    | 0  . |
x = | .  .  0  .  . |x + | .  .  1  2  3 |x'+ | .  0 |u
    | .  .  .  0  . |    | .  .  4  5  6 |    | .  0 |
```

```
         | .  .  .  .  0 |     | .  .  7  8  9 |     | .  0 |

         | 0  0  .  .  . |
     y = | .  .  0  0  0 |x
```

- Parallel composition

```
-->s3=s1+s2
 s3  =
         | 0  .  .  .  . |     | 1  2  .  .  . |     | 0 |
         | .  0  .  .  . |     | 3  4  .  .  . |     | 0 |
     x = | .  .  0  .  . |x +  | .  .  1  2  3 |x'+  | 0 |u
         | .  .  .  0  . |     | .  .  4  5  6 |     | 0 |
         | .  .  .  .  0 |     | .  .  7  8  9 |     | 0 |

     y = | 0  0  0  0  0 |x
```

- Series composition

```
-->s1*s2
 ans  =
         | 0  .  .  .  . |     | 1  2  .  .  . |     | 0 |
         | .  0  .  .  . |     | 3  4  .  .  . |     | 0 |
     x = | 0  0  0  .  . |x +  | .  .  1  2  3 |x'+  | . |u
         | 0  0  .  0  . |     | .  .  4  5  6 |     | . |
         | 0  0  .  .  0 |     | .  .  7  8  9 |     | . |

     y = | .  .  0  0  0 |x
```

- Input in common

```
-->[s1;s2]
 ans  =
         | 0  .  .  .  . |     | 1  2  .  .  . |     | 0 |
         | .  0  .  .  . |     | 3  4  .  .  . |     | 0 |
     x = | .  .  0  .  . |x +  | .  .  1  2  3 |x'+  | 0 |u
         | .  .  .  0  . |     | .  .  4  5  6 |     | 0 |
         | .  .  .  .  0 |     | .  .  7  8  9 |     | 0 |

         | 0  0  .  .  . |
     y = | .  .  0  0  0 |x
```

- Output addition

```
-->[s1,s2]
 ans  =
         | 0  .  .  .  . |     | 1  2  .  .  . |     | 0  . |
         | .  0  .  .  . |     | 3  4  .  .  . |     | 0  . |
     x = | .  .  0  .  . |x +  | .  .  1  2  3 |x'+  | .  0 |u
         | .  .  .  0  . |     | .  .  4  5  6 |     | .  0 |
         | .  .  .  .  0 |     | .  .  7  8  9 |     | .  0 |

     y = | 0  0  0  0  0 |x
```

- Feedback composition

```
-->s1/.s2
```

```
 ans  =
     | 0  .  0  0  0 |     | 1  2  .  .  . |     | 0 |
     | .  0  0  0  0 |     | 3  4  .  .  . |     | 0 |
 x = | 0  0  0  .  . |x + | .  .  1  2  3 |x'+ | . |u
     | 0  0  .  0  . |     | .  .  4  5  6 |     | . |
     | 0  0  .  .  0 |     | .  .  7  8  9 |     | . |

 y = | 0  0  .  .  . |x
```

- Extraction

```
-->s4=full(s4)
 s4  =
     | 0  .  .  .  . |     | 1  2  .  .  . |     | 0  . |
     | .  0  .  .  . |     | 3  4  .  .  . |     | 0  . |
 x = | .  .  0  .  . |x + | .  .  1  2  3 |x'+ | .  0 |u
     | .  .  .  0  . |     | .  .  4  5  6 |     | .  0 |
     | .  .  .  .  0 |     | .  .  7  8  9 |     | .  0 |

     | 0  0  .  .  . |
 y = | .  .  0  0  0 |x
-->s4(1,1)
 ans  =
     | 0  .  .  .  . |     | 1  2  .  .  . |     | 0 |
     | .  0  .  .  . |     | 3  4  .  .  . |     | 0 |
 x = | .  .  0  .  . |x + | .  .  1  2  3 |x'+ | . |u
     | .  .  .  0  . |     | .  .  4  5  6 |     | . |
     | .  .  .  .  0 |     | .  .  7  8  9 |     | . |

 y = | 0  0  .  .  . |x
```

We can simulate a maxplus linea system.

```
-->y=simul(s1,[1:10])
 y  =
!1  5  9  13  17  21  25  29  33  37  !
```

3. Application to production systems

Let us give an illustration of the usefulnes of these functionalities to simulate and optimize production systems.

Let us first define a flowshop by a matrix describing the resources used and the processing times. Each line is associated to a machine class and each columns to a piece class. The entries of the matrix are the processing times. If a piece class does not need a machine class the corresponding entry is $-\infty$. Because we consider only a flowshop the piece classes go on machine classes in sequence from the first to the last machine class.

```
-->PT=[#(2),3.9,0.95,1.1,0.7,1.4;
-->%0,%0,2,1.2,%0,1.7;
-->3.7,%0,2.2,%0,6.4,%0;
-->%0,%0,2,%0,1,1;
-->1.7,3.1,3,%0,1.3,%0;
```

```
-->0.5,3.2,4.3,1.9,1.6,0.4;
-->1,1,1,1,1,1;
-->1.5,1.5,1.5,1.2,1.2,1.2]
 PT  =
!2    3.9  0.95  1.1  0.7  1.4  !
!                               !
!.    .    2     1.2  .    1.7  !
!                               !
!3.7  .    2.2   .    6.4  .    !
!                               !
!.    .    2     .    1    1    !
!                               !
!1.7  3.1  3     .    1.3  .    !
!                               !
!0.5  3.2  4.3   1.9  1.6  0.4  !
!                               !
!1    1    1     1    1    1    !
!                               !
!1.5  1.5  1.5   1.2  1.2  1.2  !
-->[nmach,npiece]=size(PT)
 npiece  =
    6.
 nmach  =
    8.
```

Let us give the machine number in each class.

```
-->nm=ones(1,nmach)
 nm  =

!   1.    1.    1.    1.    1.    1.    1.    1. !
```

Let us give the piece number in each class.

```
-->np=ones(1,npiece)
 np  =
!   1.    1.    1.    1.    1.    1. !
```

Let us show a graphic representation of the corresponding cyclic flowshop.

```
-->[g,T,N]=flowshop_graph(PT,nm,np,50);
```

Let us compute the throughput by the flowshop by the Howard algorithm.

```
-->[chi,v]=semihoward(T,N);
-->chi'
 ans  =
         column  1 to 10
!16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  !
         column 11 to 20
!16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  !
         column 21 to 30
!16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  !
         column 31 to 40
!16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  !
```

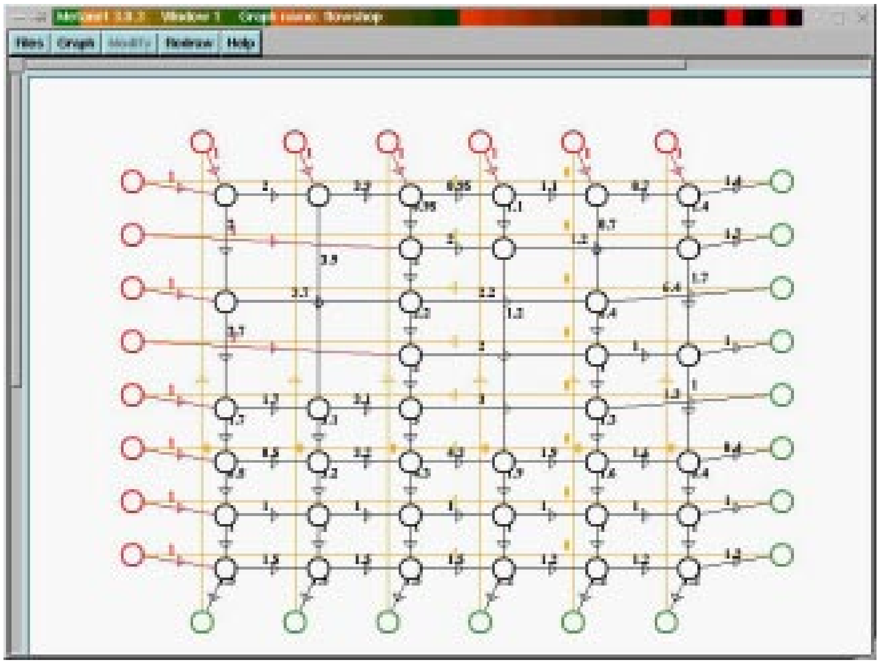

FIGURE 1. Flowshop.

```
         column 41 to 50
!16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  !
         column 51 to 60
!16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  16.95  !
         column 61 to 65
!16.95  16.95  16.95  16.95  16.95  !
-->v'
 ans  =
         column  1 to 11
!23.8  6.85  6.85  19.45  2.5  15.75  -1.2  11.45  -5.5  9.35  -7.6  !
         column 12 to 22
!8.35  -8.6  6.85  21.8  4.85  14.05  10.95  7.35  6.35  4.85  17.9  !
         column 23 to 33
!0.95  16.95  0  14.95  12.75  -4.2  10.75  7.75  3.45  2.45  0.95  !
         column 34 to 44
!10.7  -6.25  2.9  -1.6  -3.95  -4.95  -6.25  9.6  -7.35  8.9  1.1  !
         column 45 to 55
!0.1  -3.5  -5.15  -6.15  -7.35  8.25  -8.7  1.7  -3.2  -5.1  -6.4  !
         column 56 to 65
!-7.4  -8.7  6.85  0  2.5  -4.2  -1.2  -5.5  -7.6  -8.6  !
```

Let us show the critical circuit which is the bottleneck limiting the throughput.

```
-->show_cr_graph(g);
```

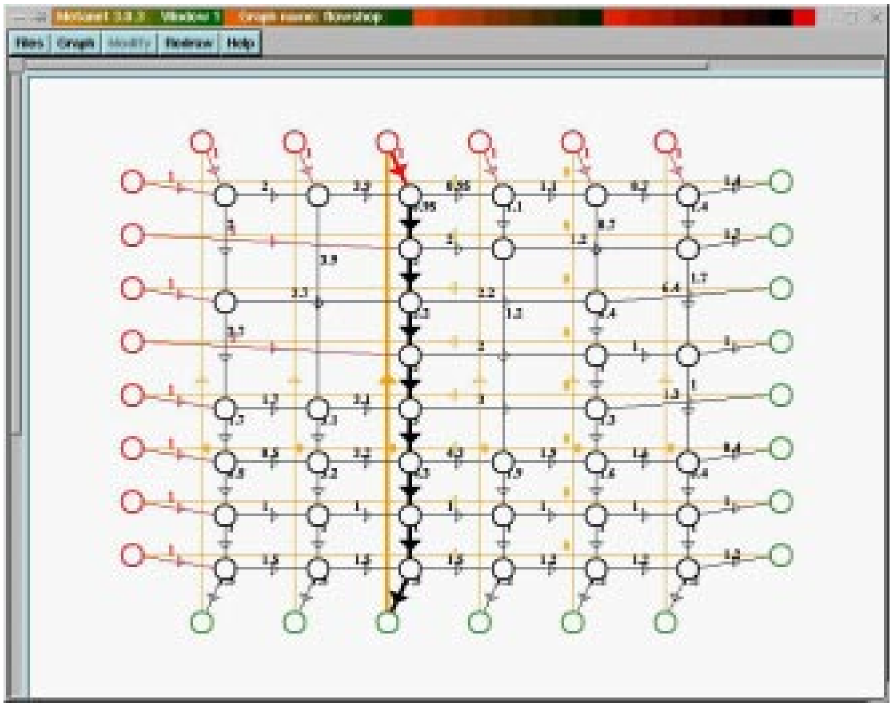

FIGURE 2. Critical circuit of the initial flowshop.

3.1. OPTIMISATION OF THE PALLET NUMBERS

Let us optimize by hand the number of pallet in the system. For that we add pallet in critical circuit with vertical arcs. The presence of such arcs means that machines are waiting for a piece.

First improvement of the pallet numbers.

```
-->pnb=[1,16,28,46,58,74];
-->g('edge_label')(pnb)=string([1,1,2,1,1,1]);
```

The new critical circuit

```
-->show_cr_graph(g);
```

The final critical circuit saturate the slowest machine.

```
-->g('edge_label')(pnb)=string([2,2,2,2,2,2]);
-->show_cr_graph(g);
```

See Figure 4.

3.2. SIMULATION OF THE FLOWSHOP

Implicit state representation of open-loop flowshop.

```
-->s=flowshop(PT)
 s  =
 x(n)=Dx(n)+Ax(n-1)+Bu(n)
 A=
(   48,   48) zero sparse matrix
 B=
(   48,   14) sparse matrix
```

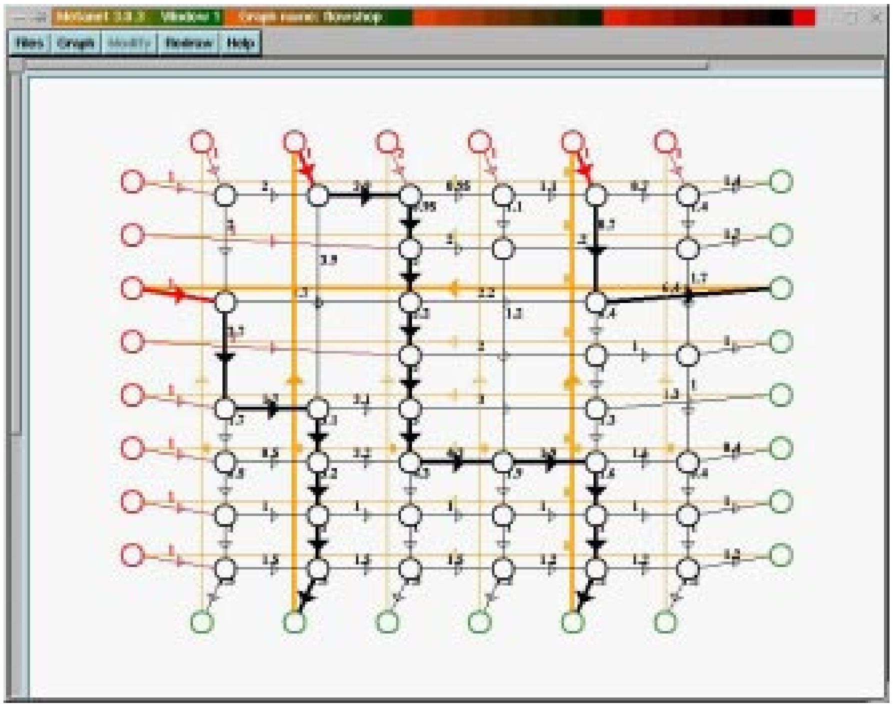

FIGURE 3. Critical circuit after the first improvement.

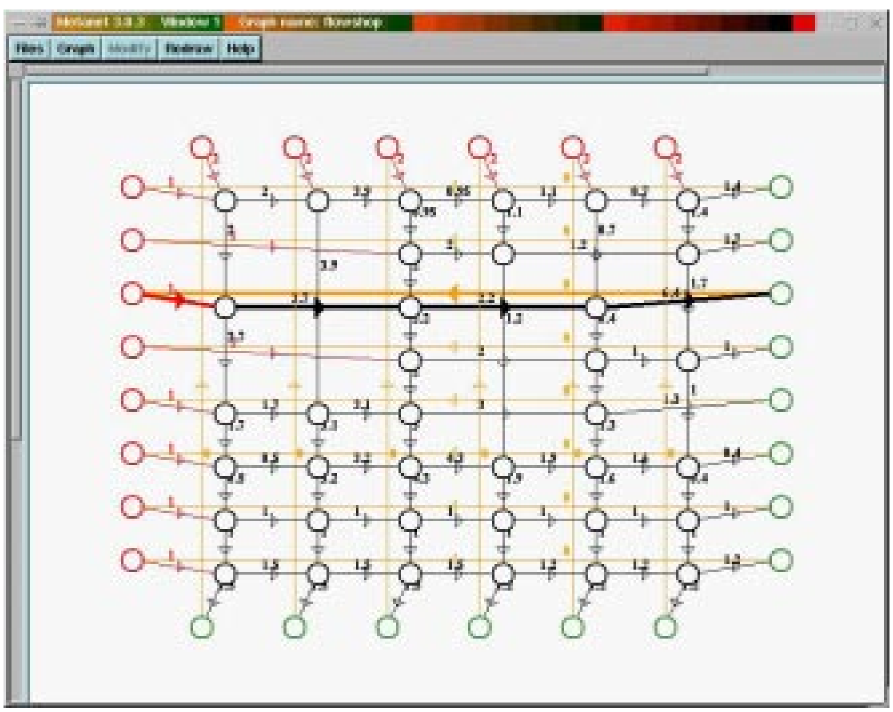

FIGURE 4. Final critical circuit.

```
(     1,     1)         0.
(     1,     7)         0.
(     2,     2)         0.
```

```
(    3,    3)        0.
(    4,    4)        0.
(    5,    5)        0.
(    6,    6)        0.
(    9,    8)        0.
(   13,    9)        0.
(   21,   10)        0.
(   25,   11)        0.
(   31,   12)        0.
(   37,   13)        0.
(   43,   14)        0.
 D=
(   48,   48) sparse matrix
(    2,    1)        2.
(    3,    2)        3.9
(    4,    3)        0.95
(    5,    4)        1.1
(    6,    5)        0.7
(    9,    3)        0.95
(   10,    4)        1.1
(   10,    9)        2.
(   12,    6)        1.4
(   12,   10)        1.2
(   13,    1)        2.
(   15,    9)        2.
(   15,   13)        3.7
(   17,    5)        0.7
(   17,   15)        2.2
(   21,   15)        2.2
(   23,   17)        6.4
(   23,   21)        2.
(   24,   12)        1.7
(   24,   23)        1.
(   25,   13)        3.7
(   26,    2)        3.9
(   26,   25)        1.7
(   27,   21)        2.
(   27,   26)        3.1
(   29,   23)        1.
(   29,   27)        3.
(   31,   25)        1.7
(   32,   26)        3.1
(   32,   31)        0.5
(   33,   27)        3.
(   33,   32)        3.2
(   34,   10)        1.2
(   34,   33)        4.3
(   35,   29)        1.3
(   35,   34)        1.9
```

```
(   36,   24)        1.
(   36,   35)        1.6
(   37,   31)        0.5
(   38,   32)        3.2
(   38,   37)        1.
(   39,   33)        4.3
(   39,   38)        1.
(   40,   34)        1.9
(   40,   39)        1.
(   41,   35)        1.6
(   41,   40)        1.
(   42,   36)        0.4
(   42,   41)        1.
(   43,   37)        1.
(   44,   38)        1.
(   44,   43)        1.5
(   45,   39)        1.
(   45,   44)        1.5
(   46,   40)        1.
(   46,   45)        1.5
(   47,   41)        1.
(   47,   46)        1.2
(   48,   42)        1.
(   48,   47)        1.2
 C=
(   14,   48) sparse matrix
(    1,   43)        1.5
(    2,   44)        1.5
(    3,   45)        1.5
(    4,   46)        1.2
(    5,   47)        1.2
(    6,   48)        1.2
(    7,    6)        1.4
(    8,   12)        1.7
(    9,   17)        6.4
(   10,   24)        1.
(   11,   29)        1.3
(   12,   36)        0.4
(   13,   42)        1.
(   14,   48)        1.2
```

The machine controller.

```
-->nm
 nm  =
!   1.    1.    1.    1.    1.    1.    1.    1. !
-->fbm=shift(nm(1),0) ;
-->for i=1:nmach-1, fbm=fbm|shift(nm(i),0) ; end ;
```

The pallet controller.

```
-->np
```

```
 np  =
!   1.    1.    1.    1.    1.    1. !
-->//
-->fbp=shift(np(1),0);
-->for i=1:npiece-1, fbp=fbp|shift(np(i),0) ; end ;
-->fbp
 fbp  =
    | .  0  .  .  .  .  .  .  .  .  .  .  . |    | .  .  .  .  .  .  |
    | .  .  .  .  .  .  .  .  .  .  .  .  . |    | 0  .  .  .  .  .  |
    | .  .  .  0  .  .  .  .  .  .  .  .  . |    | .  .  .  .  .  .  |
    | .  .  .  .  .  .  .  .  .  .  .  .  . |    | .  0  .  .  .  .  |
    | .  .  .  .  .  0  .  .  .  .  .  .  . |    | .  .  .  .  .  .  |
    | .  .  .  .  .  .  .  .  .  .  .  .  . |    | .  .  0  .  .  .  |
x = | .  .  .  .  .  .  .  0  .  .  .  .  . |x'+ | .  .  .  .  .  .  |u
    | .  .  .  .  .  .  .  .  .  .  .  .  . |    | .  .  .  0  .  .  |
    | .  .  .  .  .  .  .  .  .  0  .  .  . |    | .  .  .  .  .  .  |
    | .  .  .  .  .  .  .  .  .  .  .  .  . |    | .  .  .  .  0  .  |
    | .  .  .  .  .  .  .  .  .  .  .  0  . |    | .  .  .  .  .  .  |
    | .  .  .  .  .  .  .  .  .  .  .  .  . |    | .  .  .  .  .  0  |

    | 0  .  .  .  .  .  .  .  .  .  .  .  . |
    | .  .  0  .  .  .  .  .  .  .  .  .  . |
    | .  .  .  .  0  .  .  .  .  .  .  .  . |
y = | .  .  .  .  .  .  0  .  .  .  .  .  . |x
    | .  .  .  .  .  .  .  .  0  .  .  .  . |
    | .  .  .  .  .  .  .  .  .  .  0  .  . |
```

The complete feedback system.

```
-->sb=s/.(fbp|fbm);
```

Reducing a system and putting it in explicit form.

```
-->sbs=explicit(sb);
```

Simulation of the feedback system

```
-->u=ones(nmach+npiece,1)*(1:100);
-->y=simul(sbs,u);
-->y(:,100)'
 ans  =
         column 1 to 8
!1690.85  1693.75  1701.9  1703.5  1705.1  1706.3  1690.75  1692.45  !
         column  9 to 14
!1696.5  1698.5  1698.8  1703.3  1704.9  1706.3  !
```

Plotting the transient part of the ouputs without the stationary drift term.

- Periodicity 1 case.

```
-->chi=howard(sbs('A'));
-->chit=plustimes(chi(1))*[1:100];
-->y=plustimes(y)-ones(nmach+npiece,1)*chit;
-->xbasc(); plot2d(y(:,[1:15])');
```

- Periodicity 2 case.

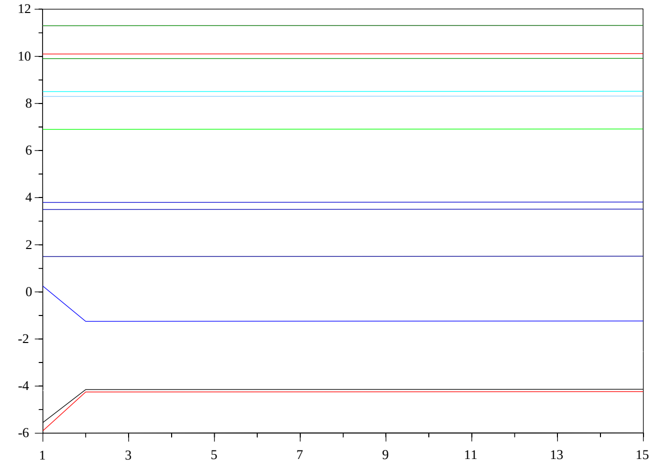

FIGURE 5. System of cyclicity 1.

```
-->np=2*ones(1,npiece); nm=3*ones(1,nmach);
-->xbasc(); [chi,y]=flowshop_simu(s,nm,np,u);
-->xbasc(); plot2d(y(:,[1:15])');
```

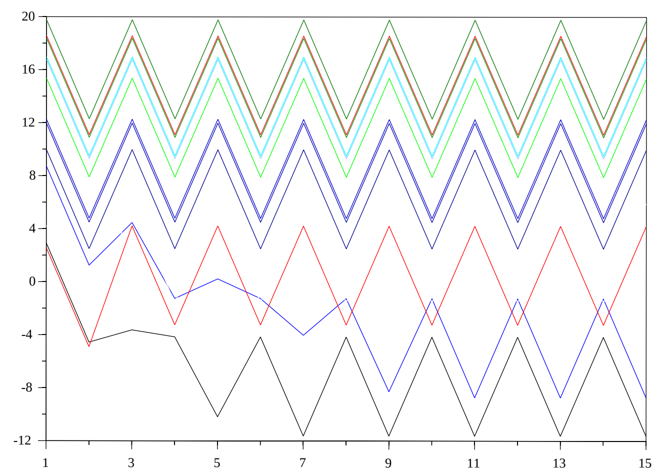

FIGURE 6. System of cyclicity 2.

- Periodicity 3 case.

```
-->np=3*ones(1,6); nm=3*ones(1,8);
-->xbasc(); [chi,y]=flowshop_simu(s,nm,np,u);
-->xbasc(); plot2d(y(:,[1:15])');
```

4. LINKS

Informations and articles about this max-plus algebra are available from the following web pages :

- http://cas.ensmp.fr/CAS/SED/index.html,
- http://amadeus.inria.fr/gaubert,
- http://www-rocq.inria.fr/scilab/quadrat,
- http://www.cs.rug.nl/ rein/alapedes/alapedes.html

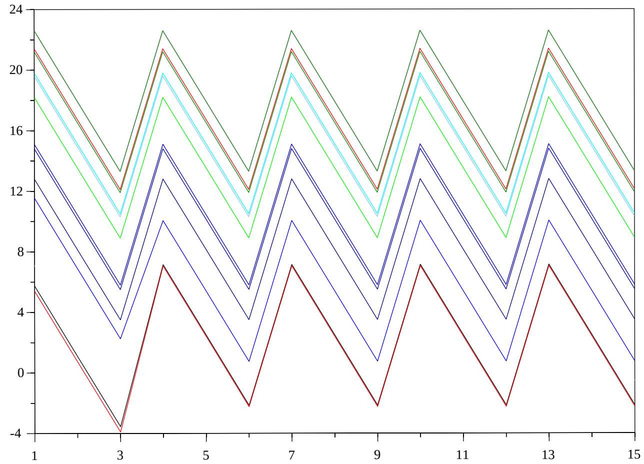

FIGURE 7. System of cyclicity 3.

In these articles a large bibliography is available.

G. COHEN, INRIA-ROCQUENCOURT, B.P. 105, 78153 LE CHESNAY CEDEX, FRANCE

S. GAUBERT, INRIA-ROCQUENCOURT, B.P. 105, 78153 LE CHESNAY CEDEX, FRANCE

J.P. QUADRAT, INRIA-ROCQUENCOURT, B.P. 105, 78153 LE CHESNAY CEDEX, FRANCE