

Discrete event system analysis using the  
max-plus-algebra

D.Wetjens

SE 420388

Master's Thesis

Supervisor: Prof. dr. ir. J.E. Rooda

Coaches: Dr. ir. A.A.J. Lefebber

Ir. J.A.W.M. van Eekelen

EINDHOVEN UNIVERSITY OF TECHNOLOGY  
DEPARTMENT OF MECHANICAL ENGINEERING  
SYSTEMS ENGINEERING GROUP

Eindhoven, June 2004



# Preface

Almost six years ago, September 1998, I started my study Mechanical Engineering at the Eindhoven University of Technology. My interests in formula one cars, airplanes, and large factories were the main reasons to choose for the world of technique. After three years of basic courses and group work, a choice of specialization had to be made. I decided to join the Systems Engineering Group (SE) of Professor Rooda. This group develops methods, techniques and tools for design, optimization and control of advanced industrial systems. After some courses with respect to the area of SE, I enjoyed two external internships. During the first internship, in cooperation with label factory Royal Sens in Rotterdam, I studied their product stream. During my second internship I stayed at the Nanyang Technological University (NTU) in Singapore. Beside working at my assignment of comparing two different dispatching rules I enjoyed to live in a multi-cultural (Asian) environment. Next to my life as a student I worked two summers as a travel guide in Spain for a company that organizes holidays for Dutch teenagers. Here, I learned how to organize, cooperate with all kinds of people, solve the most unbelievable problems and especially develop myself as a person.

This master's thesis is a result of a final assignment at the TU/e. During this last periode I have mentioned that besides my technological interests my interests in business topics have grown. Therefore, I have decided to start the Master of Science in Management Program of Nyenrode University, the Netherlands Business School.

I want to thank my coaches Erjen Lefeber and Joost van Eekelen for their pleasant and enthusiastic support and discussions, Ad Kock for reviewing my report, Professor Rooda for his support and advices with respect to my future plans, and Mieke Lousberg for her interests and nice conversations.

A special thank goes to my mom and dad. They have always supported and inspired me during my study. Their sober and realistic view on life guided me through relative tough times. I love you! I also want to thank my friends Roel, Simon, and Rob. Thanks guys, I enjoyed studying with you. Last but certainly not least I want to thank my girlfriend Majella for her support and patience during this busy recent period.

Dennis Wetjens  
Eindhoven, June 2004.



# Assignment

Student	D. Wetjens
Supervisor	Prof. dr. ir. J.E. Rooda
Advisors	Dr. ir. A.A.J. Lefeber Ir. J.A.W.M. van Eekelen
Start	September 2003
Finish	July 2004
Title	Discrete Event System analysis using the max-plus-algebra

## Subject

One approach to model and control manufacturing systems are Discrete Event Systems (DESs). For many years now, DESs, like flexible manufacturing systems, telecommunication networks, traffic control systems and logistic systems have been studied and analyzed. Therefore, many different modelling and analysis methods exist for DESs. Up to now, the most widely used technique to study DESs certainly is computer simulation, but sometimes analytic (mathematical) techniques can provide a better insight in the effect of parameter changes in DESs. Max-plus algebra is such a technique. The basic operations of the max-plus-algebra are maximization and addition. Using the max-plus-algebra, some properties of the system can fairly easily be derived, whereas in some cases brute force simulation might require a large amount of computation time. On the other side, the max-plus-algebra makes it possible to control DESs, by using extended model predictive control (MPC).

## Assignment

Analyse and try to understand the max-plus-algebra 'language' by literature investigation. Study (simple) manufacturing systems by using the max-plus-algebra; important

items are determination of properties of the system, controlling of the systems and the limits of the algebra. The manufacturing systems that are going to be used will have deterministic process times. Present the results in a report and provide recommendations for further research.

# Notation

In this report all sorts of symbols and abbreviations are used. It is possible that these are not familiar to the reader. These notations, and their short explanation, are represented here.

## List of symbols

$\mathbb{R}$	set of the real numbers
$\mathbb{Z}$	set of the integers
$(a, b)$	open interval in $\mathbb{Z}$ : $(a, b) = \{x \in \mathbb{Z}   a < x < b\}$
$\sim$	asymptotic equivalence
$\bar{x}(k)$	state vector at sample $k$
$\bar{u}(k)$	input vector at sample $k$
$\bar{y}(k)$	output vector at sample $k$
$\tilde{y}(k)$	predicted output vector at sample $k$
$\hat{y}(k + j   k)$	predicted output element of $\tilde{y}(k)$ at sample $k + j$ , based on information at sample $k$
$x^T$	transpose of vector $x$
$a_i$	$i$ th component of the vector $a$
$a_{ij}$	entry of the matrix $A$ on the $i$ th row and the $j$ th column
$ x $	1-norm of the vector $x$
$\oplus$	max-algebraic addition
$\otimes$	max-algebraic multiplication
$\varepsilon$	zero element for max-plus-algebra: $\varepsilon = -\infty$
$E_n$	$n$ by $n$ max-algebraic identity matrix
$\varepsilon_{m \times n}$	$m$ by $n$ max-algebraic zero matrix
$A^{\otimes n}$	$n$ th max-algebraic power of the matrix $A$
$P_i$	product of type $i$
$d_i$	deterministic process time of machine $i$
$d_{ij}$	deterministic process time of machine $i$ for product $P_j$

$J_{\text{in}}$	tracking error or output cost criterion
$J_{\text{out}}$	control effort or input cost criterion
$J$	objective function
$N_c$	length of control horizon
$N_p$	length of prediction horizon
$\lambda$ , and $\mu$	MPC weighting parameters
$x_0$	initial state vector ( $= x(0)$ )
$\rho$	average duration of a production cycle
$k_0$	length of impuls response
$G_k$	response matrices or Markov parameters
$\Delta s(k)$	$s(k) - s(k-1)$
$\bar{u}^*(k)$	optimal input sequence of the relaxed and the original optimization problem
$\bar{y}^*(k)$	optimal output sequence of the relaxed optimization problem
$\bar{y}^\#(k)$	optimal output sequence of the original optimization problem

## List of abbreviations

DES	Discrete Event System
MPC	Model Predictive Control
SISO	Single Input, Single Output
MIMO	Multi Input, Multi Output
FIFO	First In, First Out
LP	Linear Programming
SQP	Sequential Quadratic Programming
ELCP	Extended Linear Complementarity Problem



# Summary (in Dutch)

Bij academisch onderzoek en in de industrie gebruikt men sinds vele jaren simulatie methoden en analytische modellen om inzicht te verkrijgen in het gedrag van fabricage systemen. Het gedrag van deze systemen wordt middels deze modellen beschreven door wiskundige vergelijkingen. Deze vergelijkingen kunnen gebruikt worden voor het analyseren, optimaliseren en regelen van systemen. Sommige event gestuurde processen in de industrie kunnen worden geanalyseerd middels discrete modellen. Deze industriële systemen worden Discrete-Event Systemen (DES) genoemd. Typische voorbeelden van DES zijn transport systemen, communicatie netwerken, logistieke systemen en flexibele fabricage systemen. Het modelleren en analyseren van DES kan gedaan worden middels verschillende technieken. Computersimulatie is, tot nu toe, de meest gebruikte techniek om DES te bestuderen en vereist een hoge mate van abstractie van het systeem. Dit leidt tot een grote mate van overeenstemming tussen het model en het werkelijke systeem. Echter leidt computer simulatie niet altijd tot een duidelijke verklaring van het vertoonde gedrag van het systeem bij een parameter verandering. Een alternatief voor het analyseren van een DES is het gebruik van een wiskundig model.

De max-plus-algebra is zo'n wiskundige techniek die gebruikt kan worden voor het analyseren en modelleren van fabricage systemen. Systemen die gemodelleerd kunnen worden met alleen de max en de plus operatoren worden max-lineair in de max-plus-algebra en kunnen geschreven worden middels een max-algebraïsche toestandsvergelijking. De max-plus-algebra bestaat uit twee basis operatoren: maximalisatie of de max-plus optelling,  $\oplus$  en de optelling of max-plus vermenigvuldiging,  $\otimes$ . Vergeleken met de conventionele algebra, wordt de max vervangen door  $\oplus$  en de  $+$  door  $\otimes$ .

Het belangrijkste doel van dit project is om te bepalen of de max-plus-algebra een bruikbare techniek is voor het analyseren, modelleren en regelen van fabricage systemen. Twee belangrijke objecten in de industrie zijn machines en buffers met een (on)eindige capaciteit. Deze en andere basis elementen, zoals samenvoegen en batchprocessing, zijn gemodelleerd met behulp van de max-plus-algebra. De modellen van de basis elementen, zoals machines en buffers, hebben twee ingaande en twee uitgaande stromen. De ene ingang ontvangt informatie betreffende de beschikbaarheid van de gekoppelde structuur in stroomafwaartse richting, terwijl de andere ingang producten ontvangt van de structuur in stroomopwaartse richting. Een soortgelijke situatie geldt voor de twee uitgangen van het model. Het modelleren van een compleet fabricage systeem wordt

gedaan door de processen eerst individueel te modelleren en deze vervolgens te koppelen.

Om de geschiktheid van de max-plus-algebra te evalueren is een theoretische case geanalyseerd. In deze case is een compleet fabricage systeem bestaande uit verschillende structuren en verschillende soorten gedrag gemodelleerd met behulp van de max-plus-algebra en het formalisme  $\chi$ . Zowel de max-plus-algebra als  $\chi$  beschrijven het gedrag van een systeem exact. Tijdens het gebruik van de max-plus-algebra zijn een aantal zaken waargenomen. De elegante en overzichtelijke toestandsbeschrijving kan veel inzicht geven in het gedrag van een fabricage systeem. Dit inzicht vermindert echter als de systemen groter en complexer worden. Dit wordt veroorzaakt door het groter worden van de systeemmatrices. De max-plus-algebra is, tot nu toe, nog niet bruikbaar als simulatie gereedschap ter bepaling van de invloed van een bepaalde parameter. Hierbij moet bijvoorbeeld gedacht worden aan de invloed van het aantal machines of aantal buffer plaatsen met betrekking tot de prestaties van een fabricage systeem. Daarom wordt aanbevolen om automatisering door te voeren bij het gebruik van de max-plus-algebra om systemen te analyseren en te modelleren.

Een tweede doel van dit project is het gebruik van 'Model Predictive Control' (MPC) in combinatie met de max-plus-algebra. In dit onderzoek is MPC gebruikt om de uitgang van het fabricage systeem van de theoretische case te regelen met betrekking tot een bepaald referentie signaal. Deze regelmethode bepaalt een optimaalingangssignaal met betrekking tot een doelfunctie. Het optimaliseren is gedaan met behulp van Lineair Programmeren (LP).

Waargenomen is, gedurende de implementatie van de max-plus-algebra in combinatie met MPC, dat het gebruik van de max-plus-algebra en de conventionele algebra zorgt voor moeilijkheden. Vooral het gebruik van  $\varepsilon \stackrel{\text{def}}{=} -\infty$  en de max-algebraïsche toestandsbeschrijving in combinatie met de conventionele algebra zorgt voor een onordelijk geheel en is tijdrovend. Het gebruik van de elegante toestandsbeschrijving om het gedrag van een fabricage systeem te beschrijven zorgt voor een overzichtelijk en inzichtelijk geheel. Echter, dit voordeel vervaagt als de algebra wordt gebruikt in combinatie met de conventionele algebra.

Het fabricage systeem dat is gebruikt in de theoretische case bewijst geschikt te zijn als model en voor analyse. Het dominante gedrag van enkele processen in het systeem, de van te voren vastgestelde route van de producten door het systeem, en de deterministische bewerkingstijden maken dat het fabricage systeem helaas niet het vooraf gewenste dynamische gedrag vertoont. Hierdoor lijkt het regelen van een fabricage systeem met behulp van de max-plus-algebra en MPC een onordelijk en triviaal geheel. Een case studie met een fabricage systeem dat een meer dynamisch gedrag vertoont zou meer informatie kunnen geven over de kracht van de combinatie van de max-plus-algebra en MPC.

# Summary

Since many years, academic research and industrial practice use simulation- and analytical models to gain insight in the behavior of manufacturing systems. These models are sets of mathematical equations that describe the behavior of a system. They can be used for analysis, optimization and control. Some event-driven processes in industry can be analyzed by means of discrete models. These industrial systems are called Discrete Event Systems (DESs). Typical examples of DESs are transportation systems, communication networks, logistics systems, and (flexible) manufacturing systems. Modelling and analysis of DESs can be done using many different techniques. Up to now, the most widely used technique to study DESs certainly is computer simulation. Computer simulation can require a high degree of detail in the model which leads to a high degree of correspondence between the model and the real system. However, computer simulations do not always give a real understanding and explanation of the effects of parameter changes on the behavior of the system. An alternative way of analyzing DESs, is using mathematical models.

The max-plus-algebra is such a mathematical technique. It can be used to model and analyze manufacturing systems. Systems that can be modelled using only maximization (max) and addition (plus) become max-linear in the max-plus-algebra and can be written using the max-algebraic state-space description. The max-plus-algebra consists of two basic operators: maximization or the max-plus addition,  $\oplus$  and addition or the max-plus multiplication,  $\otimes$ . Compared to the conventional algebra, the max is replaced by  $\oplus$  and the  $+$  by  $\otimes$ .

The main objective of this project is to determine whether the max-plus-algebra is suitable as a technique to model, analyse and control manufacturing systems. In industry, main elements of manufacturing systems are machines and (in)finite buffers. These elements and some other basic structures, for instance merging and batching, have been modelled using the max-plus-algebra. In these models, main elements such as machines and buffers have two incoming streams and two outgoing streams. One input receives availability information of the coupled structure from the downstream direction, while the other input receives products from the coupled structure in the upstream direction. A similar approach is valid for the outputs of the system. Modelling an entire manufacturing system can be done best by individually modelling the processes and coupling these structures afterwards.

To evaluate the suitability of the max-plus-algebra a theoretical case has been analyzed. In this case, a manufacturing system with all sorts of structures and policies has been modelled using both the max-plus-algebra and the formalism  $\chi$ . Both tools describe the behavior of the system exactly. Using the max-plus-algebra, many observations are done. The elegant state-space description can provide good insight in the behavior of a system. This insight decreases if the system becomes larger and more complex due to the increased size of the system matrices. The max-plus-algebra is, up to now, not useful if simulations are required to determine the influence on certain parameters or the performance of a system with respect to e.g. the number of machines or the number of buffer places. Some automation in combination with the max-plus-algebra is recommended.

A second aim of this research project is to use Model Predictive Control (MPC) in combination with the max-plus-algebra. In this research, MPC is used to control the output sequence of the manufacturing system of the theoretical case with respect to a certain reference signal. This control approach finds an optimal input sequence with respect to a certain objective function. This optimization is done using Linear Programming (LP).

An observation obtained during the implementation of the max-plus model in combination with MPC, is that the use of both the max-plus-algebra and the conventional algebra causes some difficulties. Especially the use of  $\varepsilon \stackrel{\text{def}}{=} -\infty$  and the max-algebraic state-space description in combination with the conventional algebra results in a sometimes untidy and time-consuming approach as a whole. The elegant state-space description of the manufacturing system gains an orderly effect, but this advantage decreases if this algebra is used in combination with the conventional algebra.

The manufacturing system that is used as a theoretical case proved to be a good study for both modelling and analysis. Unfortunately the dominant behavior of certain processes and the fixed product route and deterministic process times did not turn out to have the diverse dynamic behavior as desired. Therefore, controlling a manufacturing system in combination with the max-plus-algebra and MPC seems to be untidy and trivial as a whole. However, a more dynamic case study might give some more information about the strength of this combination.

# Contents

<b>Preface</b>	<b>i</b>
<b>Assignment</b>	<b>iii</b>
<b>Notation</b>	<b>v</b>
<b>Summary (in Dutch)</b>	<b>vii</b>
<b>Summary</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Introduction to the max-plus-algebra</b>	<b>5</b>
2.1 Basic operations . . . . .	5
2.2 Main elements and standard matrices . . . . .	7
2.3 Summary . . . . .	9
<b>3 Modelling manufacturing systems using the max-plus-algebra</b>	<b>11</b>
3.1 Building a max-plus model . . . . .	11
3.2 General approach . . . . .	13
3.3 Introduction of a new structure . . . . .	14
3.4 Finite and infinite buffers . . . . .	17
3.5 Multi product structures . . . . .	19
3.6 Coupling structures . . . . .	22
3.7 Re-entrancy . . . . .	26

3.8	Model reduction . . . . .	31
3.9	Summary . . . . .	32
<b>4</b>	<b>Theoretical case</b>	<b>33</b>
4.1	General information . . . . .	33
4.2	Infinite buffer . . . . .	34
4.3	Machines 1 and 2 . . . . .	37
4.4	Finite buffer . . . . .	38
4.5	Batch machine . . . . .	42
4.6	Coupling . . . . .	45
4.7	Output explanation and validation . . . . .	46
4.8	Discussion . . . . .	51
<b>5</b>	<b>Control of a manufacturing system</b>	<b>55</b>
5.1	Model Predictive Control . . . . .	55
5.2	The standard MPC problem . . . . .	57
5.3	Tuning the MPC parameters . . . . .	64
5.4	MPC implementation and simulation results . . . . .	66
5.5	Discussion . . . . .	76
<b>6</b>	<b>Conclusions and recommendations</b>	<b>77</b>
6.1	Conclusions . . . . .	77
6.2	Recommendations for future research . . . . .	81
	<b>Bibliography</b>	<b>85</b>
<b>A</b>	<b>Algorithms to calculate system matrices</b>	<b>87</b>
<b>B</b>	<b>Details model reduction</b>	<b>89</b>
B.1	The minimal system order . . . . .	90
B.2	Minimal state-space realization . . . . .	90
B.3	Summary . . . . .	92

<b>C</b>	<b>System matrices theoretical case</b>	<b>93</b>
<b>D</b>	<b>Matlab model of the theoretical case</b>	<b>97</b>
<b>E</b>	<b><math>\chi</math> validation files</b>	<b>101</b>
E.1	Standard $\chi$ model . . . . .	101
E.2	Specified $\chi$ model . . . . .	106
E.3	Simulation results of both $\chi$ files . . . . .	113
<b>F</b>	<b>MPC implementation</b>	<b>117</b>
<b>G</b>	<b>Matlab file of MPC implementation</b>	<b>127</b>
G.1	mainfile.m . . . . .	127
G.2	calcHg.m and process.m . . . . .	133





# Chapter 1

## Introduction

### Background

According to the neo-Darwinistic view, evolution takes place through the creation of random combinations. Some of these evolutions will result in a struggle for existence. Some combinations will survive and proliferate, while other perish. This is a typical example of trial and error-elimination. In industry, this trial and error method is not desirable due to the fact that time and money would be wasted. Instead of trial and error, models are used to predict the behavior of processes in industry. Since many years, academic research and industrial practice use simulation- and analytical models. An important class of models consists of mathematical equations that describe the behavior of a system and they can be used for analysis, optimization and control. During the last decade, these models have become more complex due to the increased complexity of these systems.

Some event-driven processes in industry can be analyzed using discrete models. These industrial systems are called Discrete Event Systems (DESs). Typical examples of DESs are transportation systems, communication networks, logistics systems, (flexible) manufacturing systems, parallel processing systems, and traffic control systems. DESs are characterized by two main items. First, their dynamics are event-driven instead of time-driven, in other words, the behavior of a DES is governed by occurrences of different types of events over time rather than ticks of a clock [Sch96]. Compared to conventional time-driven systems, the expired time between event occurrences does not necessarily have a visible effect on the system. Second, at least some of the natural variables necessary to describe a DES are discrete [Cas95]. Examples of events are an arrival of a product at a machine, the completion of a product on a machine, an arrival of a product in a buffer or a machine breakdown. The intervals between events do not have to be identical, they can be deterministic or stochastic [Cas95]. The theory of DESs has been developed enormously the last decades.

## Modelling and analysis of DESs

A certain class of DESs exists where synchronization, but no concurrency takes place. Synchronization requires the availability of several resources (e.g. machines) at the same time. Concurrency appears when a choice has to be made between the use of several resources, or, in other words, the product flow through the systems is variable [Sch96]. These general DESs can be modelled using only the maximization (max) and the addition (plus) operations, which leads to a non-linear description in conventional algebra.

Modelling and analysis of DES can be done using many different techniques (Petri nets, finite state machines, automata, formal languages, process algebra, computer languages, etc). Up to now, the most widely used technique to study DESs certainly is computer simulation [Sch96], such as  $\chi$  formalism. Computer simulation can require a high degree of detail in the model which leads to a high degree of correspondence between the model and the real system. A computer simulation does not always give a real understanding of the effects of parameter changes on properties of the system. An alternative way of analyzing DESs, is mathematical modelling.

The max-plus-algebra is such a mathematical technique to analyse and model DESs. Systems that can be modelled using only maximization (max) and addition (plus), become 'linear', when formulated in the max-plus-algebra. Such a model is called max-linear. Compared to the linear system algebra, the max-plus-algebra has not been developed equally far. The max-plus-algebra consists of two main basic elements: maximization or the max-plus addition,  $\oplus$ , and addition or the max-plus multiplication,  $\otimes$ . Compared to the conventional linear algebra, the max is replaced by  $\oplus$  and the  $+$  by  $\otimes$ . Other properties (e.g. eigenvalues) can be translated from the conventional algebra to the max-plus-algebra.

To illustrate the use of the max-plus-algebra, consider a single lot machine with a known, deterministic process time that can always send finished products to a free output. In this simple example, the variables of interest are the arrival and the departure times of the products. The machine can start working if it received its raw material and if the previous (finished) product has been sent away. These two conditions reflect the synchronization feature. The start of the process is the maximum of the arrival time of the raw material and the time instant at which the previous product leaves the machine. Hence, the max operation is the basic operator through which variables interact. The departure time is the sum of the time instant at which a machine starts processing and the deterministic process time. There is no concurrency since the route of the product is fixed. This implies that an algebra exists in which DESs that do not involve concurrency can naturally be modelled as max-linear systems [Bac92]. This machine example shows that the max is the essential operation that captures the synchronization phenomenon. Start times are computed from arrival and departure times. This example also indicates that the conventional addition operation is necessary to add the deterministic process time to the time instant the machine starts working. Using these

main max-plus operations, entire manufacturing systems can be modelled using the max-plus-algebra.

## Previous research and objectives

Previous work of Baccelli, Cassandras and de Schutter [Bac92, Cas95, Sch96, Sch97] show that DESs can be modelled and analyzed using the max-plus-algebra. In general, all sorts of DESs have been modelled in the max-plus-algebra (e.g. railway systems and manufacturing systems). The modelled and analyzed manufacturing systems are rather small and simple and, up to now, not suitable to model entire manufacturing systems. In [Sch00a] and [Sch01] a Model Predictive Control (MPC) control framework is presented for max-plus-linear systems. This framework is only used on small manufacturing systems.

The main objective of this project, is to determine whether the max-plus-algebra is suitable as a tool to model, analyse and control manufacturing systems. Therefore, in this Master's thesis only the modelling of max-linear manufacturing systems is considered. A systematic approach will be presented to model large systems which contain all sorts of structures and policies. All different kinds of structures (and its policies) are modelled separately. The model of the entire manufacturing system is obtained by coupling all the small structures. Another objective is to use MPC to control the manufacturing systems.

## Outline

In this Master's thesis only DESs of manufacturing systems that can be described by max-linear time-invariant state-space models are considered. Here, a time-invariant system is a system that responses to a certain input sequence and that is not dependent on absolute time [Sch96]. These systems are modelled and analyzed, using the max-plus-algebra. In Chapter 2, an introduction of the max-plus-algebra is given. The theory is discussed using some examples to make the algebra clear. In Chapter 3, the modelling of manufacturing systems, using the max-plus-algebra is discussed. This is done by introducing structures of a machine and a(n) (in)finite buffer. Using this module a more extended manufacturing system can be modelled with a general approach. In this chapter, several structures are modelled using the max-plus-algebra. The resulting models of entire manufacturing lines unfortunately become large for simple structures. To reduce the number of states, at the end of this chapter a method of model reduction is discussed. In Chapter 4, a theoretical case containing several structures is worked out in detail to demonstrate how to use max-plus-algebra to model (large) manufacturing systems. In Chapter 5, the manufacturing system that is modelled in Chapter 4 is controlled using Model Predictive Control (MPC). Using this control method, the output

sequence of the system is controlled with respect to a certain reference signal. In the last chapter, conclusions and recommendations for future research are discussed.

## Chapter 2

# Introduction to the max-plus-algebra

Most readers are familiar with the conventional algebra. However, the conventional algebra differs significantly from the max-plus-algebra. In this chapter the basic operations, the main elements and the standard matrices, necessary to work with this new algebra, are discussed.

### 2.1 Basic operations

As explained in Chapter 1, the max-plus-algebra can be used as an analysis tool that exists for discrete event systems. With this technique, properties of DESs (eg. utilization, bottleneck, critical path) are derived easily, whereas in some cases brute force simulation might require a large amount of computation time [Sch96]. In this chapter the differences between the max-plus-algebra and the conventional algebra are clarified. Conventional algebra refers to the algebra of the real (or the complex) numbers with addition and multiplication as basic operations [Sch96]. Examples of these operations are addition (+), subtraction (−), multiplication (×) and division (÷). In the max-plus-algebra, only two main operations are known. The first operation is maximization or the so called max-plus addition. This operation is represented by  $\oplus$ . The  $\oplus$  can best be explained using two examples. Two scalars and two matrices are added in the max-plus domain.

$$1 \oplus 2 = \max(1, 2) = 2$$

$$\begin{aligned}
A \oplus B &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \oplus \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \\
&= \begin{pmatrix} a_{11} \oplus b_{11} & a_{12} \oplus b_{12} \\ a_{21} \oplus b_{21} & a_{22} \oplus b_{22} \end{pmatrix} \\
&= \begin{pmatrix} \max(a_{11}, b_{11}) & \max(a_{12}, b_{12}) \\ \max(a_{21}, b_{21}) & \max(a_{22}, b_{22}) \end{pmatrix} \\
&= \max(a_{ij}, b_{ij}).
\end{aligned}$$

Here  $a_{ij}$  and  $b_{ij}$  denote the elements in the  $i$ th row and the  $j$ th column of matrix  $A$  and  $B$  respectively.

The max-plus addition of two matrices can be compared to the matrix addition in the conventional algebra. The same calculation rules apply here. It should be noted that, as in the conventional algebra, the max-plus addition of two matrices  $A$  and  $B$  is defined only if  $A$  and  $B$  have the same number of rows and the same number of columns, in other words, if  $A$  and  $B$  are of equal size.

The second operation is addition or the so called max-plus multiplication. This operation is represented by  $\otimes$ . Similar to the max-plus multiplication, the  $\otimes$  can be explained best with two examples. Again two scalars and two matrices are used to make this operation more clear.

$$1 \otimes 2 = 1 + 2 = 3$$

$$\begin{aligned}
A \otimes B &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \\
&= \begin{pmatrix} (a_{11} \otimes b_{11}) \oplus (a_{12} \otimes b_{21}) & (a_{11} \otimes b_{12}) \oplus (a_{12} \otimes b_{22}) \\ (a_{21} \otimes b_{11}) \oplus (a_{22} \otimes b_{21}) & (a_{21} \otimes b_{12}) \oplus (a_{22} \otimes b_{22}) \end{pmatrix} \\
&= \begin{pmatrix} \max(a_{11} + b_{11}, a_{12} + b_{21}) & \max(a_{11} + b_{12}, a_{12} + b_{22}) \\ \max(a_{21} + b_{11}, a_{22} + b_{21}) & \max(a_{21} + b_{12}, a_{22} + b_{22}) \end{pmatrix}.
\end{aligned}$$

Here  $a_{ij}$  and  $b_{ij}$  denote the elements in the  $i$ th row and the  $j$ th column of matrix  $A$  and  $B$  respectively.

Observe that,  $A \otimes B \neq A + B$ .

As mentioned previously, matrix multiplication rules in the conventional algebra are equal to matrix multiplication rules in the max-plus-algebra. The difference here, is that in the max-plus-algebra,  $(+)$  and  $(\times)$  are replaced by the max-plus operators  $(\oplus)$

and  $(\otimes)$ . Observe, that, as in the conventional algebra, the product of  $A$  and  $B$  is defined only if the number of rows of  $B$  is identical to the number of columns of  $A$ .

The reason for using the above mentioned symbols, is the remarkable analogy between  $\oplus$  and addition and between  $\otimes$  and multiplication: many concepts and properties from conventional linear algebra (such as the eigenvectors and eigenvalues, etc.) can be translated to the max-plus-algebra [Sch96].

Now the basic operations,  $\oplus$  and  $\otimes$ , have been described, their power, that follows from these two operations can be explained. For scalars this is easy to understand:

$$\underbrace{x \otimes x \otimes x \otimes \cdots \otimes x}_n = x^{\otimes n} = nx.$$

This means that  $x^{\otimes 0} = 0$ . Multiplying identical matrices (or vectors), can be simplified using the power operator. This is shown below:

$$\underbrace{A \otimes A \otimes A \otimes \cdots \otimes A}_n = A^{\otimes n}.$$

The rules for the order of evaluation of the max-algebraic operators are similar to those of conventional algebra. So max-algebraic power has the highest priority, and max-algebraic multiplication has a higher priority than max-algebraic addition.

## 2.2 Main elements and standard matrices

Now the two basic operations and their represented symbols are known, the elements used in the max-plus-algebra can be defined. The elements of the max-plus-algebra are the real numbers ( $\mathbb{R}$ ) and  $\varepsilon \stackrel{\text{def}}{=} -\infty$ . This means that all real numbers,  $\varepsilon$ , the operations  $\oplus$  and  $\otimes$  together define the max-plus-algebra. Now,  $\mathbb{R} \cup \{\varepsilon\}$ ,  $\oplus$ , and  $\otimes$  define the max-plus-algebra.

There is another important difference between the conventional algebra and the max-plus-algebra. Part of the basics of the conventional algebra are the rules of equality. These rules give the properties of the zero-element, 0, and the one-element, 1.

For 0, the following equality rule is valid:

$$\text{for } 0: \quad 0 + x = x + 0 = x \quad (\forall x). \quad (2.1)$$

This equality does not exist in the max-plus-algebra due to the new operator  $\oplus$ . Therefore, 0 has to be replaced by an other element. In the max-plus-algebra, the 0 is replaced by  $\varepsilon$ . Now, the equality rule of the form (2.1) is valid in the max-plus-algebra:

$$\text{for } \varepsilon: \quad \varepsilon \oplus x = x \oplus \varepsilon = x \quad (\forall x).$$

A similar equality rule is valid for the one-element, 1.

$$\text{for } 1: \quad 1 \times x = x \times 1 = x \quad (\forall x). \quad (2.2)$$

In the max-plus-algebra, the equality rule of the form (2.2) can not be used, due to the max-plus operator  $\otimes$ . Therefore, in the max-plus domain, 1 is replaced by 0. This gives the following max-plus equality rule:

$$\text{for } 0: \quad 0 \otimes x = x \otimes 0 = x \quad (\forall x).$$

The replacements described above for the zero- and one-element, result in different zero- and identity matrices (so called standard matrices) when compared to the conventional algebra. The matrix  $E_n$  is the  $n \times n$  max-algebraic identity matrix:

$$\begin{aligned} (E_n)_{ii} &= 0 \quad \text{for } i = 1, 2, \dots, n, \\ (E_n)_{ij} &= \varepsilon \quad \text{for } i = 1, 2, \dots, n \text{ with } i \neq j. \end{aligned}$$

An example is an  $E_3$  identity matrix:

$$\begin{pmatrix} 0 & \cdot & \cdot \\ \cdot & 0 & \cdot \\ \cdot & \cdot & 0 \end{pmatrix}.$$

For reasons of clarity, the  $\varepsilon$ 's in this identity matrix are replaced by dots. The 'zero' power of a matrix, for example  $A^{\otimes 0}$  is equal to  $E_n$ .

The  $m \times n$  max-algebraic zero matrix is represented by  $\varepsilon_{m \times n}$ :

$$(E_{m \times n})_{ij} = \varepsilon \text{ for all } i, j$$

An example is an  $\varepsilon_{3 \times 2}$  zero matrix:

$$\begin{pmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{pmatrix}.$$



## 2.3 Summary

In this chapter, the max-plus-algebra has been introduced. The two main operations, maximization (max-plus-addition) and addition (max-plus multiplication) have been explained with examples together with the algebra's elements and standard matrices. Now,  $\mathbb{R} \cup \{\varepsilon\}$ ,  $\oplus$ , and  $\otimes$  define the max-plus-algebra.



## Chapter 3

# Modelling manufacturing systems using the max-plus-algebra

In Chapter 2, the basic operations, elements and standard elements have been discussed. This knowledge is necessary to build max-plus models of manufacturing systems. In general, DESs lead to a non-linear description in conventional algebra, but a subclass of DESs exists for which this model becomes 'linear' when it is formulated in max-plus-algebra [Sch96]. Such a model is called max-linear. In this chapter manufacturing systems are modelled using the max-plus-algebra. Different structures and situations are discussed. If possible, the max-plus models are reduced using model reduction.

### 3.1 Building a max-plus model

Using the conventional algebra for modelling DESs leads to non-linearities [Sch96]. In the max-plus-algebra, DESs lead to max-linear models and can be written in the following max-linear time-invariant state-space description, henceforth called state-space description:

$$x(k+1) = A \otimes x(k) \oplus B \otimes u(k) \quad (3.1a)$$

$$y(k) = C \otimes x(k) \oplus D \otimes u(k). \quad (3.1b)$$

To show that manufacturing systems can be modelled using the max-plus algebra, and specifically the state-space description, a single machine is modelled. Before this simple machine is modelled, a few assumptions have to be made:

- all process times are deterministic,
- the process time includes setup times,

- the process sequence in which products go through the manufacturing line is given and not variable,
- no machine failures take place,
- transportation times are negligible,
- the free output of the system can always receive a finished product.

Using these assumptions, the modelling of the machine can be started. A schematic representation of this machine is given in Figure 3.1.

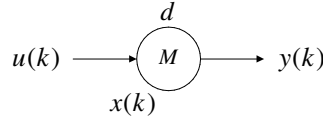


Figure 3.1: Single machine

In this figure,  $u(k)$ ,  $x(k)$ ,  $y(k)$ , and  $d$  are defined as follows:

- $u(k)$ : time instant at which raw material is fed to the input of the system for the  $(k+1)$ st time,
- $x(k)$ : time instant at which the processing unit starts working for the  $k$ th time,
- $y(k)$ : time instant at which a finished product leaves the system for the  $(k+1)$ st time,
- $d$ : deterministic process time.

Counter  $k$  is increased by 1 every time a new product is fed to the system. An important note here, is that  $k$  counts events rather than ticks of the clock. Input  $u$  is related to  $(k+1)$  to keep the state-space relations comprehensible. Output  $y$  is related to  $(k+1)$  to obtain easy coupling, which is further discussed in Section 3.6.

The machine can only start processing if two main conditions are true:

- condition 1: the raw material or (half)-product that has to be processed is available,
- condition 2: the machine has finished processing its previous product.

This implies that, when the machine has received the raw material or half-product, it can only start processing when the machine has finished the previous product. The opposite situation is possible as well: the machine has finished its previous product,

but the raw material or half-product that has to be processed is not yet available. In this situation, the machine can only start working on the next product when the raw material or next halfproduct is available. In both situations, the machine can start its next job if the above described two conditions are both true. The machine eventually starts if the last condition becomes true. The machine starts on the latest (= maximum) of both times the condition becomes true. Hence, the machine can start with its next operation,  $x(k+1)$ , if the raw material is available at  $u(k) + 0$ , and if the previous product is finished at  $x(k) + d$ . Here,  $x(k) + d$  is the moment a product leaves the system, which can be written as:

$$x(k+1) = \max(\overbrace{u(k) + 0}^{\text{raw material}}, \overbrace{x(k) + d}^{\text{previous product}}) \quad (3.2)$$

$$y(k) = x(k+1) + d. \quad (3.3)$$

This model would, because of the maximization, lead to non-linearity into the conventional algebra. In order to write the above equations in the state-space description (3.1), one extra step has to be done. If (3.2) is substituted in (3.3), this results in:

$$\begin{aligned} x(k+1) &= \max(u(k) + 0, x(k) + d) \\ y(k) &= \max(u(k) + d, x(k) + 2d). \end{aligned}$$

These equations can be written in state-space description, see (3.1). For that, the max has to be replaced by  $\oplus$  and the  $+$  has to be replaced by  $\otimes$ . This leads to the following equations:

$$\begin{aligned} x(k+1) &= d \otimes x(k) \oplus 0 \otimes u(k) \\ y(k) &= 2d \otimes x(k) \oplus d \otimes u(k). \end{aligned}$$

Here, the  $A$ ,  $B$ ,  $C$  and  $D$  matrices in (3.1) are equal to the scalars  $d$ ,  $0$ ,  $2d$  and  $d$ . The notation of the  $C$ -matrix,  $2d$ , is not correct in the max-plus domain. The correct notation is  $d^{\otimes 2}$ . For reasons of clarity, the notation of  $2d$  is used.

The simple model, described above, shows that it is possible to describe a manufacturing system using the max-plus-algebra.

## 3.2 General approach

In the previous section, a single machine has been modelled using the max-plus-algebra. This model shows that simple models can be modelled using the state-space description. In industry, many structures exist to produce all sorts of products. Examples are

splitting, merging, by-passing, back-tracking, batching, re-entrancy etc. In order to show the applicability of the max-plus-algebra to analyze and model DESs, all these structures have to be written down in this algebra.

A manufacturing system used in industry can contain many of the (complex) structures mentioned above. To model these systems using the max-plus-algebra, an algorithm has to be found to make the modelling understandable.

A commonly applied approach (e.g. systems theory), is to model each structure separately, which results in system matrices  $A$ ,  $B$ ,  $C$  and  $D$ . This model can be seen as a module which can be coupled to other modules (with other system matrices  $A$ ,  $B$ ,  $C$  and  $D$ ) to model a complete manufacturing system. The coupled structures with (small) system matrices lead to 4 large system matrices of the total manufacturing system. A schematic illustration of this approach can be seen in Figure 3.2,

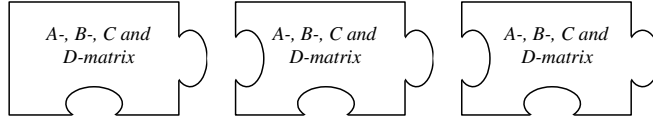


Figure 3.2: Coupling systematics

Important is that the structure of the original small system matrices does not change if the coupling takes place. The structure as described in Section 3.1 or represented in Figure 3.1 cannot be used, since the system matrices of the modules change if the modules are combined.

If a max-plus model is made of a manufacturing system, the assumptions as described in Section 3.1 are valid. If another (second) structure is coupled to this first structure, the assumption that the free output can always receive a product is not necessarily valid anymore for the first structure. If the second structure is not available, the first structure can not send its product, which affects the original state-space description (system matrices).

To use the general approach, a new structure has to be introduced. This is done in the next section.

### 3.3 Introduction of a new structure

The previous section explains why a new structure is needed to use the general approach; i.e. the necessity of availability information. An availability information 'stream' has to be added to extend the old model. This new structure is illustrated in Figure 3.3

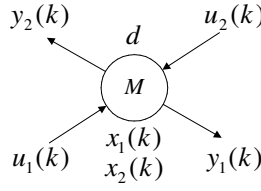


Figure 3.3: New structure

In this figure, the inputs, outputs and states etc. are defined as follows:

- $u_1(k)$ : time instant at which raw material is fed to the input of the processing unit for the  $(k + 1)$ st time,
- $y_1(k)$ : time instant at which a finished product leaves the processing unit for the  $(k + 1)$ st time,
- $u_2(k)$ : time instant at which the processing unit, directly coupled to this processing unit, can receive a product for the  $(k + 1)$ st time,
- $y_2(k)$ : time instant at which the processing unit becomes available for the  $(k + 1)$ st time ( $=y_1(k - 1)$ ),
- $x_1(k)$ : time instant at which the processing unit starts working on a product for the  $k$ th time,
- $x_2(k)$ : time instant at which the processing unit, directly coupled to this processing unit, can receive a product for the  $k$ th time ( $=u_2(k - 1)$ ).

Variables with index 2 are the additional definitions added to define the availability information.

Two main conditions that have to be true before a machine can start processing are defined in Section 3.1. Using the new structure, a third condition, linked to the availability information, is necessary.

- condition 3: the structure, directly coupled to the machine, has to be available to receive the finished product.

To explain this new structure, the machine of Section 3.1 is remodelled. The schematic representation can be seen in Figure 3.1.

The machine can start processing a new product for the  $(k + 1)$ st time if all three mentioned conditions become true: the raw material is fed to the machine for the  $(k + 1)$ st time, represented by  $u_1(k)$ . Another condition is that the machine has to be finished processing the  $k$ th product (this is the previous product). Taking into account the new

third condition, the start of the new process is dependent of the availability information. If the previous product has been finished, but the next structure is not available yet, the machine can not start processing the  $(k + 1)$ st product. The next structure has to be available to receive a product for the  $k$ th time. This means that the third element in the  $x_1(k + 1)$  equation is  $u_2(k - 1)$ . This results in the following equation:

$$x_1(k + 1) = \max(u_1(k), x_1(k) + d, u_2(k - 1)). \quad (3.4)$$

The last element of this equation,  $u_2(k - 1)$ , leads to the addition of an extra state,  $x_2(k)$ . This state is defined as:

$$x_2(k + 1) = u_2(k). \quad (3.5)$$

Substitution of (3.5) in (3.4) gives:

$$x_1(k + 1) = \max(u_1(k), x_1(k) + d, x_2(k)). \quad (3.6)$$

Now the two states are known, the first part of the state-space description of the form (3.1a), is given:

$$\bar{x}(k + 1) = \begin{pmatrix} d_1 & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{pmatrix} \otimes \bar{u}(k)$$

where:

$$\bar{x}(k) = \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} \quad \bar{u}(k) = \begin{pmatrix} u_1(k) \\ u_2(k) \end{pmatrix}.$$

The output of the system can be determined as follows. The machine can send a product to the other structure for the  $(k + 1)$ st time if the machine has finished the processing for the  $(k + 1)$ st time,  $x_1(k + 1) + d$ , and if the coupled structure can receive a product for the  $(k + 1)$ st time,  $u_2(k)$ . This all can be expressed in the following equation:

$$y_1(k) = \max(x_1(k + 1) + d, u_2(k)). \quad (3.7)$$

The substitution of (3.6) in (3.7) results in:

$$y_1(k) = \max(u_1(k) + d, x_1(k) + 2d, x_2(k) + d, u_2(k)). \quad (3.8)$$

The other output,  $y_2(k)$ , is defined as  $y_1(k - 1)$  (see definitions). Using (3.7)  $y_2(k)$  becomes:



$$\begin{aligned}
y_2(k) &= \max(x_1(k) + d, u_2(k-1)) \\
&= \max(x_1(k) + d, x_2(k)).
\end{aligned} \tag{3.9}$$

These two output equations ((3.8) and (3.9)) are captured in the second part of the state-space description of the form (3.1b). As the state equations, the  $C$  and  $D$  are matrices instead of the scalars used in the 'old' structure.

$$\bar{y}(k) = \begin{pmatrix} 2d & d \\ d & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{u}(k)$$

with:

$$\bar{y}(k) = \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix}.$$

Now the total max-plus model of a machine that processes one product per system feed can be seen below:

$$\bar{x}(k+1) = \begin{pmatrix} d & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{pmatrix} \otimes \bar{u}(k) \tag{3.10}$$

$$\bar{y}(k) = \begin{pmatrix} 2d & d \\ d & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{u}(k). \tag{3.11}$$

Compared to the model described in Section 3.1, the structure introduced in this section has a state-space that is twice as large. For large manufacturing systems this can lead to an explosion of the state-space description. The numbers of states might be reduced if necessary (see Section 3.8). This model reduction results in a decreased number of states, but also in the loss of information (e.g., the time instants at which machines start processing). This can be a disadvantage, if the models get larger, e.g. if complete manufacturing systems are modelled.

### 3.4 Finite and infinite buffers

In the previous section, a machine has been modelled in the max-plus-algebra. Next to machines, buffers are important objects of a DES. Buffers can have a finite or an infinite capacity, called respectively finite or infinite buffers.

The finite buffer has a finite capacity. This means that if the buffer is full, the buffer can not receive a product. This finite buffer is illustrated in a similar way as the machine. This all is illustrated in Figure 3.4. Here,  $n$  represents the number of buffer places.

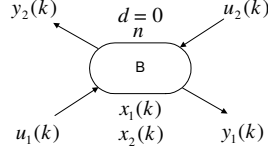


Figure 3.4:  $n$ -place finite buffer

Modelling a 1-place buffer using the max-plus-algebra is easy, because the model is equal to the model of the machine described above with a process time  $d = 0$ . The total model of this finite buffer with one buffer place, that receives one product per feed, can be seen below. Modelling a finite buffer with  $n$  buffer places can be done by coupling  $n$  of these one place finite buffers in sequence. This results in large system matrices and thus an increase in the number of dimensions. Coupling is discussed in Section 3.6.

$$\begin{aligned}\bar{x}(k+1) &= \begin{pmatrix} 0 & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k) &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{u}(k).\end{aligned}$$

Another buffer is the infinite buffer. This buffer has an infinite capacity and can always receive a product. An example of an infinite buffer is illustrated in Figure 3.5. Here,  $n$  has been removed due to the infinite number of buffer places.

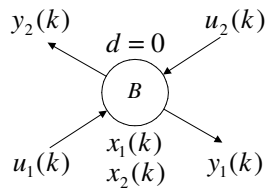


Figure 3.5: Infinite buffer

The model of this infinite buffer can be compared to the model of the finite buffer. The second state  $x_2(k)$  and the second output  $y_2(k)$  are added because of the needed availability stream. An infinite buffer can always receive products due to its infinite capacity. As mentioned before, an  $n$  place buffer can be modelled by coupling  $n$  finite buffers (coupling is described in Section 3.6). This increase of the number of states is caused by the events that occur inside the buffer. Information of the location of products inside the buffer have to be saved. This increase of the number of states does

not occur with infinite buffers because no information of the location of products has to be saved. The infinite buffer is always available, in other words,  $y_2(k) = \varepsilon (\forall k)$ . This results in the following model for the infinite buffer that receives one product per feed:

$$\begin{aligned}\bar{x}(k+1) &= \begin{pmatrix} 0 & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k) &= \begin{pmatrix} 0 & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{u}(k).\end{aligned}$$

Note, that the number of states of a finite buffer increases if the number of products per feed increase, and if the number of buffer places increases. The number of states of the infinite buffer, however, only increases if the number of products per feed increases. The significant difference between the infinite buffer and the finite buffer is that the infinite buffer does not need extra states to be modelled due to the amount of buffer places. The finite buffer needs to save information about the position of the product in the buffer. The infinite buffer does not have to save this information, which results in less states.

### 3.5 Multi product structures

The machine, finite and infinite buffer have been modelled in the max-plus-algebra for only one product per feed to the system. If, for example a situation in which splitting or merging has to be modelled, more than one product per feed of the system can be required. This means that the structures presented above have to be modelled for multi-products instead as well. Note, that a feed of 2 products does not necessarily mean that both products are fed to the system at the same time instant.

To start easily, the standard single machine machine is extended from one product per feed of the system to two products. This results in the model of Figure 3.6.

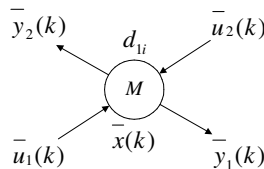


Figure 3.6: Multi product machine

with:

$$\bar{x}(k) = \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{pmatrix} \quad \bar{u}(k) = \begin{pmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \\ u_4(k) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ y_4(k) \end{pmatrix}.$$

This figure illustrates that the number of inputs, states and outputs doubles. This observation in combination with some more investigation show that the model of a machine and infinite buffer has dimension  $2n$  (=number of states) if  $n$  products are received per feed of the system. The process time of the machine of the  $n$  different products are not necessary similar. Therefore, the deterministic process time of multi product structures are indexed:  $d_{ij}$  (see Figure 3.6). Here,  $i$  and  $j$  represent respectively the machine- and product number.

The machine receives two products per feed, say product  $P_1$  and  $P_2$ . Or in other words, the product mix is  $P_1, P_2, P_1, P_2$  etc. The process time of  $P_1$  is called  $d_{11}$ , for  $P_2$ , this is  $d_{12}$ . The first two states are the time instants at which the machine starts working on the two products. This means that  $x_1(k)$  is the time instant at which the machine starts working on  $P_1$  for the  $k$ th time. The same definition is valid for  $x_2(k)$  and  $P_2$ .

The machine starts working on  $P_1$  for the  $(k+1)$ st time when it receives the raw material for  $P_1$  for the  $(k+1)$ st time,  $u_1(k)$ . Another condition is that the machine needs to have finished processing its previous product. This is not a product of type  $P_1$ , but a product of type  $P_2$ , according to the product mix, which results in  $x_2(k) + d_{12}$ . The coupled structure has to be available to receive  $P_2$  for the  $k$ th time,  $u_4(k-1)$ . This results into the following equation for  $x_1(k+1)$ :

$$x_1(k+1) = \max(u_1(k), x_2(k) + d_{12}, u_4(k-1)). \quad (3.12)$$

For product  $P_2$  a similar calculation can be done. The machine can start processing product  $P_2$  if it receives its raw material,  $u_2(k)$  and if the machine finished its previous product, in this case product  $P_1$  (for the  $(k+1)$ st time, according to the product mix),  $x_1(k+1) + d_{11}$ . The last condition that is needed before the machine can start processing  $P_2$  is the availability of the coupled structure to receive  $P_1$  for the  $(k+1)$ st time,  $u_3(k)$ . This results in the following equation:

$$x_2(k+1) = \max(u_2(k), x_1(k+1) + d_{11}, u_3(k)). \quad (3.13)$$

Substitution of (3.12) in (3.13) gives:

$$x_2(k+1) = \max(u_2(k), u_1(k) + d_{11}, x_2(k) + d_{11} + d_{12}, u_4(k-1) + d_{11}, u_3(k)). \quad (3.14)$$

The variable  $u_4(k-1)$  results in the addition of an extra state. Later in this section, it can be seen that the output equations  $y_3(k)$  and  $y_4(k)$  result in addition of another

extra state. Therefore, the two additional states are given here.

$$x_3(k+1) = u_3(k) \quad (3.15)$$

$$x_4(k+1) = u_4(k). \quad (3.16)$$

Substitution of (3.15) and (3.16) in (3.12) and (3.14) results in:

$$x_1(k+1) = \max(u_1(k), x_2(k) + d_{12}, x_4(k)) \quad (3.17)$$

$$x_2(k+1) = \max(u_2(k), u_1(k) + d_{11}, x_2(k) + d_{11} + d_{12}, x_4 + d_{11}, u_3(k)). \quad (3.18)$$

Combining (3.17) and (3.18) the first part of the state-space description of the form, (3.1a), can be obtained:

$$\bar{x}(k+1) = \begin{pmatrix} \cdot & d_{12} & \cdot & 0 \\ \cdot & d_{11} + d_{12} & \cdot & d_{11} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \cdot & \cdot & \cdot \\ d_{11} & 0 & 0 & \cdot \\ \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & 0 \end{pmatrix} \otimes \bar{u}(k). \quad (3.19)$$

Calculation of the system's output is similar to the deduction presented above. The machine can send product  $P_1$  for the  $(k+1)$ st time if it finished  $P_1$  for the  $(k+1)$ st time,  $x_1(k+1) + d_{11}$  and if the coupled structure can receive  $P_1$  for the  $(k+1)$ st time,  $u_3(k)$ . This results into the following equation for  $y_1(k)$ :

$$y_1(k) = \max(x_1(k+1) + d_{11}, u_3(k)). \quad (3.20)$$

Product  $P_2$  is sent to the coupled structure by the machine for the  $(k+1)$ st time, if the machine finished processing this product for the  $(k+1)$ st time,  $x_2(k+1)$ , and if the coupled structure can receive this product for the  $(k+1)$ st time,  $u_4(k)$ . This gives the following equation for  $y_2(k)$ :

$$y_2(k) = \max(x_2(k+1) + d_{12}, u_4(k)). \quad (3.21)$$

Substitution of (3.17) in (3.20) and (3.18) in (3.21) gives:

$$\begin{aligned} y_1(k) &= \max(u_1(k) + d_{11}, u_3(k), x_2(k) + d_{11} + d_{12}, x_4(k) + d_{11}) \\ y_2(k) &= \max(u_1(k) + d_{11} + d_{12}, u_2(k) + d_{12}, u_3(k) + d_{12}, u_4(k), \\ &\quad x_2(k) + d_{11} + 2d_{12}, x_4(k) + d_{11} + d_{12}). \end{aligned}$$

As discussed previously, the outputs of the availability information,  $y_3(k)$  and  $y_4(k)$ , equal  $y_1(k-1)$  and  $y_2(k-1)$ . This results in the following equations:

$$\begin{aligned} y_3(k) &= \max(x_1(k) + d_{11}, x_3(k)) \\ y_4(k) &= \max(x_2(k) + d_{12}, x_4(k)). \end{aligned}$$

The above calculation leads to the second part of the state-space description, (3.1b):

$$\bar{y}(k+1) = \begin{pmatrix} \cdot & d_{11} + d_{12} & \cdot & d_{11} \\ \cdot & d_{11} + 2d_{12} & \cdot & d_{11} + d_{12} \\ d_{11} & \cdot & 0 & \cdot \\ \cdot & d_{12} & \cdot & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d_{11} & \cdot & 0 & \cdot \\ d_{11} + d_{12} & d_{12} & d_{12} & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \bar{u}(k).$$

Summarized, the state-space description of a machine that processes two products per feed of the system can be seen below:

$$\begin{aligned} \bar{x}(k+1) &= \begin{pmatrix} \cdot & d_{12} & \cdot & 0 \\ \cdot & d_{11} + d_{12} & \cdot & d_{11} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \cdot & \cdot & \cdot \\ d_{11} & 0 & 0 & \cdot \\ \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & 0 \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k+1) &= \begin{pmatrix} \cdot & d_{11} + d_{12} & \cdot & d_{11} \\ \cdot & d_{11} + 2d_{12} & \cdot & d_{11} + d_{12} \\ d_{11} & \cdot & 0 & \cdot \\ \cdot & d_{12} & \cdot & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d_{11} & \cdot & 0 & \cdot \\ d_{11} + d_{12} & d_{12} & d_{12} & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \bar{u}(k). \end{aligned}$$

If more than one product is fed to the system per feed, ( $n > 1$ ), a general structure can be found to calculate system matrices  $A$ ,  $B$ ,  $C$  and  $D$ . This structure can be seen in Appendix A.

### 3.6 Coupling structures

The general approach as discussed in Section 3.2 and illustrated in Figure 3.2 can be applied using the structure as introduced Section 3.3. If, for instance, two structures are coupled, the output of the first structure has to be coupled to the input of the second

structure. Using this substitution, the max-plus model of the overall system can be calculated. This all can be explained best using an example.

Two single lot machines, as discussed in Section 3.3 are coupled. These machines only receive one product per system feed. A schematic representation of the coupling can be seen in Figure 3.7

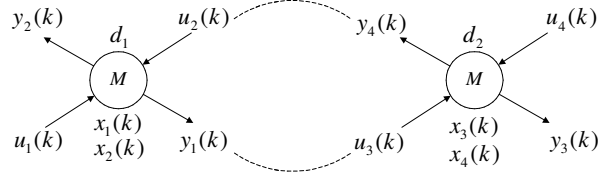


Figure 3.7: Coupling illustration

The state-space descriptions of both machines are identical. The only difference is the process time. For machine 1 the process time is  $d_1$ . Machine 2 has a process time of  $d_2$ . The state-space description now becomes:

$$\begin{aligned}\bar{x}(k+1) &= \begin{pmatrix} d_i & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k) &= \begin{pmatrix} 2d_i & d_i \\ d_i & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d_i & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{u}(k).\end{aligned}$$

The process times are indexed with  $i$ , identifying machine 1 or 2. If the 2 machines are coupled, it is obvious that the output of machine 1,  $y_1(k)$  is coupled to the input of machine 2,  $u_3(k)$ . The output of machine 2,  $u_4(k)$  is coupled to the input of machine 1,  $y_2(k)$ . This means that the coupling equations are:

$$\begin{aligned}u_2(k) &= y_4(k) \\ u_3(k) &= y_1(k).\end{aligned}$$

In order to ensure correct coupling, the structures, the input  $u$  and the output  $y$  both have to be related to  $(k+1)$ . This explains why  $y(k)$  is labelled to  $(k+1)$  instead of  $k$ .

Substitutions of  $y_1(k)$  and  $y_4(k)$ , lead to the following equation for  $u_2(k)$  and  $u_3(k)$ :

$$u_2(k) = d_2 \otimes x_3(k) \oplus 0 \otimes x_4(k) \quad (3.22)$$

$$\begin{aligned}u_3(k) &= 2d_1 \otimes x_1(k) \oplus d_1 \otimes x_2(k) \oplus d_2 \otimes x_3(k) \oplus 0 \otimes \\ &\quad x_4(k) \oplus d_1 \otimes u_1(k).\end{aligned} \quad (3.23)$$

The coupled structure consists of four states,  $x_1(k)$  to  $x_4(k)$ . Now the coupling is explained stepwise.

First the four (new) states are calculated. The first original two states, of the first machine are:

$$x_1(k+1) = d_1 \otimes x_1(k) \oplus 0 \otimes x_2(k) \oplus 0 \otimes u_1(k) \quad (3.24)$$

$$x_2(k+1) = 0 \otimes u_2(k). \quad (3.25)$$

Substitution of (3.22) in (3.25) gives:

$$x_1(k+1) = d_1 \otimes x_1(k) \oplus 0 \otimes x_2(k) \oplus 0 \otimes u_1(k)$$

$$x_2(k+1) = d_2 \otimes x_3(k) \oplus 0 \otimes x_4(k).$$

The original two states of the second machine are:

$$x_3(k+1) = d_2 \otimes x_3(k) \oplus 0 \otimes x_4(k) \oplus 0 \otimes u_3(k) \quad (3.26)$$

$$x_4(k+1) = 0 \otimes u_4(k). \quad (3.27)$$

Substitution of (3.23) in (3.26) gives:

$$x_3(k+1) = 2d_1 \otimes x_1(k) \oplus d_2 \otimes x_2(k) \oplus d_2 \otimes x_3(k) \oplus 0 \otimes x_4(k) \\ \oplus d_1 \otimes u_1(k)$$

$$x_4(k+1) = 0 \otimes u_4(k).$$

As has been seen before, a single machine has two outputs. Two coupled machines have two outputs as well. In this case, the outputs,  $y_1(k)$  and  $y_4(k)$  are connected and are no longer outputs of the coupled system. The outputs  $y_2(k)$  and  $y_3(k)$  are the outputs of the total system. Similarly, the inputs of the overall system are  $u_1(k)$  and  $u_4(k)$ . Observe that the first outputs in the output vector always contain the product flow information. Therefore, the sequence in the output vector becomes first  $y_3(k)$ , followed by  $y_2(k)$ .

$$\bar{u}(k) = \begin{pmatrix} u_1(k) \\ u_4(k) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_3(k) \\ y_2(k) \end{pmatrix}.$$

This gives the following first equation of the state-space description:

$$\bar{x}(k+1) = \begin{pmatrix} d_1 & 0 & . & . \\ . & . & d_2 & 0 \\ 2d_1 & d_2 & d_2 & 0 \\ . & . & . & . \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & . \\ . & . \\ d_1 & . \\ . & 0 \end{pmatrix} \otimes \bar{u}(k).$$



As mentioned before, the outputs of the coupled system are  $y_2(k)$  and  $y_3(k)$ :

$$y_3(k) = 2d_2 \otimes x_3(k) \oplus d_2 \otimes x_4(k) \oplus d_2 \otimes u_3(k) \oplus 0 \otimes u_4(k) \quad (3.28)$$

$$y_2(k) = d_1 \otimes x_1(k) \oplus 0 \otimes x_2(k). \quad (3.29)$$

Substitution of (3.23) in (3.28) gives:

$$\begin{aligned} y_3(k) &= 2d_1 \otimes x_1(k) \oplus d_1 \otimes x_2(k) \oplus 2d_2 \otimes x_3(k) \oplus d_2 \otimes x_4(k) \oplus \\ &\quad d_1 \otimes u_1(k) \oplus 0 \otimes u_4(k) \\ y_2(k) &= d_1 \otimes x_1(k) \oplus 0 \otimes x_2(k). \end{aligned}$$

Note that, if a substitution leads to, for example  $(d_1 \oplus 0) \otimes x_1(k)$ , this is equal to  $d_1 \otimes x_1(k)$ , because the process time is always greater than or equal to 0.

Now, the second part of the equation of the state-space description, of the form (3.1b), can be denoted:

$$\bar{y}(k) = \begin{pmatrix} 2d_1 & d_1 & 2d_2 & d_2 \\ d_1 & 0 & . & . \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d_1 & 0 \\ . & . \end{pmatrix} \otimes \bar{u}(k).$$

The total state-space description (3.1) of the coupled system is represented here:

$$\bar{x}(k+1) = \begin{pmatrix} d_1 & 0 & . & . \\ . & . & d_2 & 0 \\ 2d_1 & d_2 & d_2 & 0 \\ . & . & . & . \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & . \\ . & . \\ d_1 & . \\ . & 0 \end{pmatrix} \otimes \bar{u}(k) \quad (3.30)$$

$$\bar{y}(k) = \begin{pmatrix} 2d_1 & d_1 & 2d_2 & d_2 \\ d_1 & 0 & . & . \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d_1 & 0 \\ . & . \end{pmatrix} \otimes \bar{u}(k). \quad (3.31)$$

This section shows that the basic element, introduced as the new structure, enables usage of the general approach. All sort of structures in industry can be modelled using the basic element and the general approach. To illustrate this, a model is made for a theoretical case (see Chapter 4).

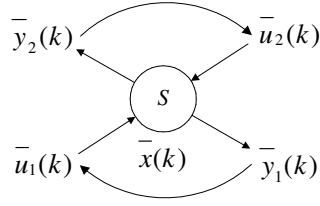


Figure 3.8: Representation re-entrancy

### 3.7 Re-entrancy

In the previous sections, different manufacturing situations have been described in the max-plus-algebra. In this section, a special situation is discussed. If products enter a manufacturing line more than once (re-enter) this is called re-entrancy. A schematic illustration of this situation can be seen in Figure 3.8. Here products enter system  $S$  with system matrices  $A$ ,  $B$ ,  $C$  and  $D$  more than one time. This way of processing can be described in the max-plus-algebra. In the situations discussed earlier, the products only enter the system in a certain sequence one time per system feed. After one feed, indicated with  $k$ , the next feed, indicated with  $(k + 1)$ , starts. The product flow of the  $(k + 1)$ th feed is identical to the flow of the  $k$ th feed, with the only difference that feed  $(k + 1)$  runs in a later time stadium. Consider a certain structure that is described in the max-plus-algebra using a state-space description. In other words, the system matrices of this structure,  $A$ ,  $B$ ,  $C$  and  $D$ , are known. These matrices are only valid in case the product enter the manufacturing system one time. If these products have to enter the same manufacturing system more than once, the system becomes re-entrant. Previously, a manufacturing system with system matrices  $A$ ,  $B$ ,  $C$  and  $D$  has been considered, where products enter the system only once. Now, the same manufacturing system becomes re-entrant, which results in a change of the system matrices. One way to model this new situation in the max-plus-algebra is to start all over. In other words, observe the new situation and determine the new state-space -description (system matrices). A more efficient way of determining the state-space description of this new process, is to use the known, 'old' system matrices  $A$ ,  $B$ ,  $C$  and  $D$ .

In this section, an algorithm is presented to calculate the new system matrices  $A$ ,  $B$ ,  $C$  and  $D$  from the 'old' matrices of the state-space description if products enter the manufacturing system  $n$  times ( $n > 1$ ). The machine described in Section 3.3 is used as an example to explain re-entrancy. Instead of entering the machine one time, the products enter the machine twice. The presented algorithm introduces a new state vector. The vector contains, all states, indexed from  $k$  to  $k + (n - 1)$  if the products enter the system  $n$  times. This results in the following state vector  $x$  of the machine in

a re-entrant situation:

$$\bar{x}(k) = \begin{pmatrix} x_1(k) \\ x_1(k+1) \\ x_2(k) \\ x_2(k+1) \end{pmatrix}.$$

This definition of the  $\bar{x}(k)$ -vector is not suitable for the re-entrant situation, because the definition of this vector changes per event. Therefore, it is decided that the state vector in case of a re-entrant situation in which products enter the manufacturing system two times becomes:

$$\bar{x}(k) = \begin{pmatrix} x_1(2k) \\ x_1(2k+1) \\ x_2(2k) \\ x_2(2k+1) \end{pmatrix}.$$

This also results in different definitions for  $u(k)$  and  $y(k)$ :

$$\bar{u}(k) = \begin{pmatrix} u_1(2k+1) \\ u_1(2k+2) \\ u_2(2k+1) \\ u_2(2k+2) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_1(2k+1) \\ y_1(2k+2) \\ y_2(2k+1) \\ y_2(2k+2) \end{pmatrix}.$$

Note that counter  $k$  in the vector notation  $\bar{x}(k)$  is not the same counter as in the elements of the vector. The counter  $k$  in the elements notation counts  $n$  times faster than the counter  $k$  in the vector notation if the products enter the system  $n$  times. Therefore, the vector elements of  $\bar{x}(k+1)$  have to be calculated as can be seen below. To make this all more clear, a representation is given in Figure 3.9. This has to be

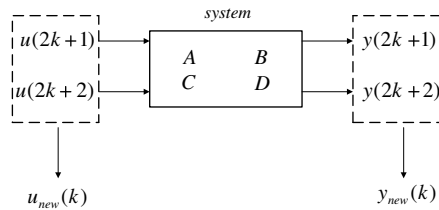


Figure 3.9: Representation new vectors

written in the standard state-space description of the form (3.1).

The new  $\bar{x}(k)$  results in the following  $\bar{x}(k+1)$ :

$$\bar{x}(k+1) = \begin{pmatrix} x_1(2 \cdot \{k+1\}) \\ x_1(2 \cdot \{k+1\} + 1) \\ x_2(2 \cdot \{k+1\}) \\ x_2(2 \cdot \{k+1\} + 1) \end{pmatrix} = \begin{pmatrix} x_1(2k+2) \\ x_1(2k+3) \\ x_2(2k+2) \\ x_2(2k+3) \end{pmatrix}.$$

All system matrices of the single lot machine are  $2 \times 2$  and have the following structure:

$$X = \left( \begin{array}{c|c} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array} \right).$$

The new elements in the state vector can be written as follows:

$$\begin{aligned} x_1(2k+2) &= A_{11} \otimes x_1(2k+1) \oplus A_{12} \otimes x_2(2k+1) \oplus B_{11} \otimes u_2(2k+1) \oplus \\ &\quad B_{12} \otimes u_2(2k+1) \end{aligned} \quad (3.32)$$

$$\begin{aligned} x_1(2k+3) &= A_{11} \otimes x_1(2k+3) \oplus A_{12} \otimes x_2(2k+2) \oplus B_{11} \otimes u_2(2k+2) \oplus \\ &\quad B_{12} \otimes u_2(2k+2) \end{aligned} \quad (3.33)$$

$$\begin{aligned} x_2(2k+2) &= A_{21} \otimes x_1(2k+1) \oplus A_{22} \otimes x_2(2k+1) \oplus B_{21} \otimes u_2(2k+1) \oplus \\ &\quad B_{22} \otimes u_2(2k+1) \end{aligned} \quad (3.34)$$

$$\begin{aligned} x_2(2k+3) &= A_{21} \otimes x_1(2k+2) \oplus A_{22} \otimes x_2(2k+2) \oplus B_{21} \otimes u_2(2k+2) \oplus \\ &\quad B_{22} \otimes u_2(2k+2). \end{aligned} \quad (3.35)$$

Substitution of (3.32) in (3.33) and (3.34) in (3.35) lead to the system matrices of the first state-space description:

$$\begin{aligned} A &= \begin{pmatrix} \cdot & A_{11} & \cdot & A_{12} \\ \cdot & A_{11}^{\otimes 2} \oplus (A_{12} \otimes A_{21}) & \cdot & (A_{11} \otimes A_{12}) \oplus (A_{12} \otimes A_{22}) \\ \cdot & A_{21} & \cdot & A_{22} \\ \cdot & (A_{21} \otimes A_{11}) \oplus (A_{22} \otimes A_{21}) & \cdot & A_{22}^{\otimes 2} \oplus (A_{21} \otimes A_{12}) \end{pmatrix} \\ B &= \begin{pmatrix} B_{11} & \cdot & B_{12} & \cdot \\ (A_{11} \oplus B_{11}) \oplus (A_{12} \otimes B_{21}) & B_{11} & (A_{11} \otimes B_{12}) \otimes (A_{12} \otimes B_{22}) & B_{12} \\ B_{21} & \cdot & B_{22} & \cdot \\ (A_{21} \otimes B_{11}) \oplus (A_{22} \otimes B_{21}) & B_{21} & (A_{21} \otimes B_{12}) \oplus (A_{22} \otimes B_{22}) & B_{22} \end{pmatrix}. \end{aligned}$$

With the output equation,  $y(k)$ , the same can be done:

$$\begin{aligned} y_1(2k+1) &= C_{11} \otimes x_1(2k+1) \oplus C_{12} \otimes x_2(2k+1) \oplus D_{11} \otimes u_1(2k+1) \oplus \\ &\quad D_{12} \otimes u_2(2k+1) \end{aligned} \quad (3.36)$$

$$\begin{aligned} y_1(2k+2) &= C_{11} \otimes x_1(2k+2) \oplus C_{12} \otimes x_2(2k+2) \oplus D_{11} \otimes u_1(2k+2) \oplus \\ &\quad D_{12} \otimes u_2(2k+2) \end{aligned} \quad (3.37)$$

$$\begin{aligned} y_2(2k+1) &= C_{21} \otimes x_1(2k+1) \oplus C_{22} \otimes x_2(2k+1) \oplus D_{21} \otimes u_1(2k+1) \oplus \\ &\quad D_{22} \otimes u_2(2k+1) \end{aligned} \quad (3.38)$$

$$\begin{aligned} y_2(2k+2) &= C_{21} \otimes x_1(2k+2) \oplus C_{22} \otimes x_2(2k+2) \oplus D_{21} \otimes u_1(2k+2) \oplus \\ &\quad D_{22} \otimes u_2(2k+2). \end{aligned} \quad (3.39)$$

After the substitution of (3.32) and (3.34) in (3.37) and (3.39), the system matrices of the second state-space description become:

$$C = \begin{pmatrix} \cdot & C_{11} & \cdot & C_{12} \\ \cdot & (C_{11} \otimes A_{11}) \oplus (C_{12} \otimes A_{21}) & \cdot & (C_{11} \otimes A_{12}) \oplus (C_{12} \otimes A_{22}) \\ \cdot & C_{21} & \cdot & C_{22} \\ \cdot & (C_{21} \otimes A_{11}) \oplus (C_{22} \otimes A_{21}) & \cdot & (C_{21} \otimes A_{12}) \oplus (C_{22} \otimes A_{22}) \end{pmatrix}$$

$$D = \begin{pmatrix} D_{11} & \cdot & D_{12} & \cdot \\ (C_{11} \otimes B_{11}) \oplus (C_{12} \otimes B_{21}) & D_{11} & (C_{11} \otimes B_{12}) \oplus (C_{12} \otimes B_{22}) & D_{12} \\ D_{21} & \cdot & D_{22} & \cdot \\ (C_{21} \otimes B_{11}) \oplus (C_{22} \otimes B_{21}) & D_{21} & (C_{21} \otimes B_{12}) \oplus (C_{22} \otimes B_{22}) & D_{22} \end{pmatrix}.$$

The model of the machine, (3.10) and (3.11), as described in Section 3.3 is used to extend from a normal to a re-entrant structure. As is described above, the system matrices become complex, even for a simple model with two dimensions. Therefore, the elements  $A_{11}$ ,  $A_{12}$  etc. are replaced by the values of the original elements of the system matrices. This results into the following simplified matrices (due to the  $\varepsilon$ -elements):

$$A = \begin{pmatrix} \cdot & d & \cdot & 0 \\ \cdot & 2d & \cdot & d \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \quad B = \begin{pmatrix} 0 & \cdot & \cdot & \cdot \\ d & 0 & 0 & \cdot \\ \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} \cdot & 2d & \cdot & d \\ \cdot & 3d & \cdot & 2d \\ \cdot & d & \cdot & 0 \\ \cdot & 2d & \cdot & d \end{pmatrix} \quad D = \begin{pmatrix} d & \cdot & 0 & \cdot \\ 2d & d & d & 0 \\ \cdot & \cdot & \cdot & \cdot \\ d & \cdot & 0 & \cdot \end{pmatrix}.$$

In a re-entrant manufacturing line, the machine is fed for the  $(k+1)$ th time by the finished products of feed  $k$ . In other words,  $u_1(k+1) = y_1(k)$ . The same is valid for the availability information,  $u_2(k+1) = y_2(k)$ . In the re-entrant situation, these coupling equations become:

$$\begin{aligned} u_1(2k+2) &= y_1(2k+1) \\ &= 2d \oplus x_1(2k+1) \oplus d \otimes x_2(2k+1) \oplus d \oplus u_1(2k+1) \oplus \\ &\quad 0 \otimes u_2(2k+1) \\ u_2(2k+2) &= y_2(2k+1) \\ &= d \otimes d \otimes x_1(2k+1) \oplus 0 \otimes x_2(2k+1). \end{aligned}$$

Using the coupling equations, the inputs, labelled with  $(2k+2)$  are eliminated. This results into two inputs instead of four. The same can be seen with the number of states

used in the state-space description:

$$A = \begin{pmatrix} . & d & . & 0 \\ . & 2d & . & d \\ . & . & . & . \\ . & d & . & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & . & . & . \\ d & . & 0 & . \\ . & . & 0 & . \\ . & . & . & . \end{pmatrix}$$

$$C = \begin{pmatrix} . & 2d & . & d \\ . & 3d & . & 2d \\ . & d & . & 0 \\ . & 2d & . & d \end{pmatrix} \quad D = \begin{pmatrix} d & . & 0 & . \\ 2d & . & d & . \\ . & . & . & . \\ d & . & 0 & . \end{pmatrix}.$$

Obviously, the columns one and three of matrices  $A$  and  $C$  only contain  $\varepsilon$ 's. This means that these states,  $x_1(2k)$  and  $x_2(2k)$  are not used. Clearly, the number of states can be reduced. Model reduction is further discussed in Section 3.8.

$$\bar{u}(k) = \begin{pmatrix} u_1(2k+1) \\ u_1(2k+2) \\ u_2(2k+1) \\ u_2(2k+2) \end{pmatrix} \quad A = \begin{pmatrix} . & d & . & 0 \\ . & 2d & . & d \\ . & . & . & . \\ . & d & . & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & . \\ d & 0 \\ . & 0 \\ . & . \end{pmatrix}$$

$$C = \begin{pmatrix} . & 2d & . & d \\ . & 3d & . & 2d \\ . & d & . & 0 \\ . & 2d & . & d \end{pmatrix} \quad D = \begin{pmatrix} d & 0 \\ 2d & d \\ . & . \\ d & 0 \end{pmatrix}.$$

If the system matrices of a manufacturing system, with  $m$  states and  $p$  in- and outputs, are known and the products go through the line  $n$  times, the  $x$ -,  $u$ - and  $y$ -vector become:

$$\bar{x}(k) = \begin{pmatrix} x_1(nk) \\ x_1(nk+1) \\ \vdots \\ x_1(nk+n-1) \\ x_2(nk) \\ \vdots \\ x_m(nk+n-1) \end{pmatrix}, \quad \bar{u}(k) = \begin{pmatrix} u_1(nk+n-1) \\ u_1(nk+n) \\ \vdots \\ u_1(nk+2n-2) \\ u_2(nk+n-1) \\ \vdots \\ u_p(nk+2n-2) \end{pmatrix}, \quad \bar{y}(k) = \begin{pmatrix} y_1(nk+n-1) \\ y_1(nk+n) \\ \vdots \\ y_1(nk+2n-2) \\ y_2(nk+n-1) \\ \vdots \\ y_p(nk+2n-2) \end{pmatrix}$$

$$\begin{aligned} u_1(nk+n) &= y_1(nk+n-1) \\ u_1(nk+n+1) &= y_1(nk+n) \\ &\vdots \\ u_1(nk+2n-2) &= y_1(nk+2n-3) \\ u_2(nk+n) &= y_2(nk+n-1) \\ &\vdots \\ u_p(nk+2n-2) &= y_p(nk+2n-3). \end{aligned}$$

### 3.8 Model reduction

In this chapter, manufacturing structures have been modelled using the max-plus-algebra. The number of states is equivalent to the size of the  $x$ -vector. The general modelling problem invariably involves a trade-off between complexity and accuracy of models. Simple models are preferred to models which contain many details and accurate models are preferred to models which have poor descriptive power. In order to obtain high accuracy models, one usually needs to resort to high detail models, while simple, low complexity models are generally inaccurate.

The complexity of max-linear time-invariant models is generally defined as the dimension of the state vector of any minimal state-space representation of the system, or the order of the system. Depending on the states of interest (e.g. the input-output behavior) most of the max-linear time-invariant systems, given in this chapter, can be reduced.

The minimal realization problem in linear system theory can be solved very efficiently. Some examples are state truncations, modal truncations, balanced truncations and Hankel norm reductions [Wei03]. Nevertheless, there still are no efficient algorithms to solve the minimal state-space realization problem in the max-plus-algebra [Sch97]. Many investigations were performed to find the minimal system order [Sch97, Gau92, Gau94] or minimal state-space realizations [Sch97, Sch96]. Details can be seen in Appendix B.

In Appendix B one can see that the presented methods to reduce the number of states of a max-plus model has major disadvantages. Because of these difficulties, the models, constructed in this chapter, can be reduced using the conventional way. If a state can be written in a max-plus-linear combination of two other states, this leads to reduction of the number of states. In this section, no algorithm is given for model reduction. This is an item for future work. Note that reduction leads to the loss of certain states. Most of the states are time instants a buffer or machine starts buffering or processing a product. If this information is needed, and the model is reduced to the minimal number of states, a transformation from the new states into the 'old' states is needed to calculate the time instants at which machines start working.

To show that the discussed models can be reduced, a simple example is given, as in Section 3.6, of two coupled machines who receive one product per feed.

If these machines are coupled, the number of states becomes four (recall (3.30)):

$$\begin{aligned}
 x_1(k+1) &= d_1 \otimes x_1(k) \oplus 0 \otimes x_2(k) \oplus 0 \otimes u_1(k) \\
 x_2(k+1) &= d_2 \otimes x_3(k) \oplus 0 \otimes x_4(k) \\
 x_3(k+1) &= 2d_1 \otimes x_1(k) \oplus d_1 \otimes x_2(k) \oplus d_2 \otimes x_3(k) \oplus 0 \otimes x_4(k) \oplus \\
 &\quad d_1 \otimes u_1(k) \\
 x_4(k+1) &= 0 \otimes u_4(k).
 \end{aligned}$$

Searching for max-plus-linear combinations, it can be seen that  $x_3(k+1)$  is a max-plus-linear combination of  $x_1(k+1)$  and  $x_2(k+1)$ :

$$x_3(k+1) = d_1 \otimes x_1(k+1) \oplus x_2(k+1).$$

Using this equation, the number of states can be reduced from four to three as follows:

$$\begin{aligned} x_1(k+1) &= d_1 \otimes x_1(k) \oplus 0 \otimes x_2(k) \oplus 0 \otimes u_1(k) \\ x_2(k+1) &= (d_1 \otimes d_2) \otimes x_1(k) \oplus d_2 \otimes x_2(k) \oplus 0 \otimes x_3(k) \\ x_3(k+1) &= 0 \otimes u_4(k). \end{aligned}$$

The number of in- and outputs remains the same. This means that if  $n$  machines are coupled, without model reduction the dimension of this coupled structure is  $2n$ . After some more research can be concluded that the model reduction described above can lead to a  $n+1$  dimensional model. Compared to the model as described in Section 3.1, the structure, with availability information stream, introduced in Section 3.3 in combination of the above mentioned model reduction leads to a small increase in the number of states.

### 3.9 Summary

In this chapter, manufacturing systems are modelled using the max-plus-algebra. A basic element is given which contains two streams. The first one is a product flow stream, the other stream contains availability information. Models of machines and buffers can be made using a number of assumptions, correct definitions, and some main conditions before the machine/buffer can start processing/buffering. Where DESs written in conventional algebra lead to non-linearity, machines, finite buffers and infinite buffers can be modelled and written into a max-linear time-invariant state-space description. Using the four system matrices  $A$ ,  $B$ ,  $C$  and  $D$  of the state-space description and a general (coupling) approach, entire manufacturing systems (which contain all sorts of structures) can be build. The system matrices of the individual structures form four new, large system matrices. These matrices can be used in a new state-space description. A special structure such as re-entrancy, where products go through the manufacturing system more than one time is discussed. Depending on the states of interest, the models as discussed in this chapter can be reduced.



## Chapter 4

# Theoretical case

Chapter 3 illustrates how max-plus models of a manufacturing system can be built. A basic element of a machine (or finite buffer) and an infinite buffer is introduced in Section 3.3. Using this element and the general approach, as described in Section 3.2, makes it possible to model (large) manufacturing systems that contain all sorts of structures, using the max-plus-algebra. In this chapter an analytical case is worked out. After this, simulations (using Matlab) and verification are done. The case is simulated using Matlab and validated.

### 4.1 General information

In the theoretical case, different structures are used to illustrate the expressiveness of the max-plus-algebra. The case contains the following structures: one infinite buffer, two single lot machines, one finite buffer with three buffer places and a batch machine with batch size two. Besides this, the case contains the structures splitting and merging.

The number of products that the entire system receives per feed, depends on the diverse structures in the manufacturing system. The products that enter the manufacturing system that is worked out in this section are split, merged, buffered in a 3-place buffer and processed in batches of two according to a certain product mix. Due to splitting, merging and batching two products are fed to the system per system feed. The product mix is chosen to be  $P_1, P_2, P_1$ , etc. Here  $P_1$  stands for product type one and  $P_2$  stands for product type two. The total structure of the case can be seen using a simplified illustration in Figure 4.1.

If the two products are fed to the system, they are split. Product  $P_1$  is sent to machine one for processing. The other product type,  $P_2$ , is processed at machine two. Both machines send the products to the finite buffer. This buffer has three buffer places but can only send and receive products in a certain product mix. This product mix is similar to the mix of the total system:  $P_1, P_2, P_1$ , etc. The last operation before

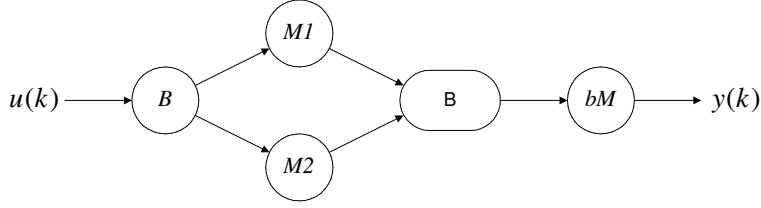


Figure 4.1: Case structure

the products leave the system is done by the batch machine. This machine needs two products, one of each product type, before it can start processing. If the products are processed by the batch machine they leave the system one by one. Now each structure with its policies are discussed stepwise.

The states of the structures are indexed in increasing order. The states of the infinite buffer are indexed with the lowest numbers and the states of the batch machine are indexed with higher numbers.

As in the rest of this report, the  $\varepsilon$ 's in the state-space descriptions are replaced by dots for reasons of clarity.

## 4.2 Infinite buffer

In Section 3.4 the infinite buffer has been discussed. Here the buffer only receives one product per system feed. This has to be extended to two products per feed. A schematical representation can be seen in Figure 4.2

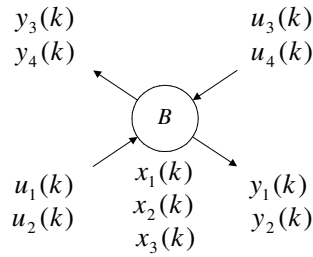


Figure 4.2: Infinite buffer

with:

- $u_1(k)$ : time instant at which product  $P_1$  is fed to the buffer for the  $(k+1)$ st time,
- $u_2(k)$ : time instant at which product  $P_2$  is fed to the buffer for the  $(k+1)$ st time,

- $u_3(k)$ : time instant at which the processing unit, directly coupled to the buffer, can receive product  $P_1$  for the  $(k + 1)$ st time,
- $u_4(k)$ : time instant at which the processing unit, directly coupled to the buffer, can receive product  $P_2$  for the  $(k + 1)$ st time,
- $y_1(k)$ : time instant at which product  $P_1$  leaves the buffer for the  $(k + 1)$ st time,
- $y_2(k)$ : time instant at which product  $P_2$  leaves the buffer for the  $(k + 1)$ st time,
- $y_3(k)$ : time instant at which the buffer becomes available for the  $(k + 1)$ st time to receive product  $P_1$ ,
- $y_4(k)$ : time instant at which the buffer becomes available for the  $(k + 1)$ st time to receive product  $P_2$ ,
- $x_1(k)$ : time instant at which the buffer starts buffering product  $P_1$  for the  $k$ th time,
- $x_2(k)$ : time instant at which the buffer starts buffering product  $P_2$  for the  $k$ th time,
- $x_3(k)$ : time instant at which product  $P_2$  leaves the buffer for the  $(k)$ th time ( $= y_2(k - 1)$ ).

As mentioned previously, the max-plus models of (in)finite buffers are almost similar to the machine models with process time  $d = 0$  (for the differences, see Section 3.4). The model of the infinite buffer is built up as a machine, with a process time that equals 0. Due to one of the main conditions (see Section 3.1), the deterministic process time has to be added in certain situations, to a time instant (state  $x$  or input  $u$ ). However, the addition of the process time is ignored in this section, since  $d = 0$ . The buffer receives and sends the products according to the system product mix,  $P_1, P_2, P_1$ , etc.

The buffer starts buffering  $P_1$  for the  $(k + 1)$ st time,  $x_1(k + 1)$ , if the product is fed to the system for the  $(k + 1)$ st time,  $u_1(k)$ , and if, according to the product mix, the second product,  $P_2$ , has been buffered for the  $k$ th time,  $x_2(k)$ . This results in the following equation for  $x_1(k + 1)$ :

$$x_1(k + 1) = \max(u_1(k), x_2(k)). \quad (4.1)$$

The same can be done for the second state. The buffer starts buffering  $P_2$  for the  $(k + 1)$ st time if this product has been fed to the system for the  $(k + 1)$ st time,  $u_2(k)$  and if product  $P_1$  has been buffered for the  $(k + 1)$ st time,  $x_1(k + 1)$ . This results in the following equation for  $x_2(k + 1)$ :

$$x_2(k + 1) = \max(u_2(k), x_1(k + 1)). \quad (4.2)$$

Substitution of (4.1) in (4.2) gives:

$$x_2(k+1) = \max(u_1(k), u_2(k), x_2(k)). \quad (4.3)$$

Product  $P_1$  leaves the system for the  $(k+1)$ st time if it is fed to the system for the  $(k+1)$ st time,  $u_1(k)$ , if it is buffered for the  $(k+1)$ st time,  $x_1(k+1)$ , if, according to the product mix,  $P_2$  has left the buffer for the  $(k)$ th time,  $y_2(k-1)$ , and if machine one can receive  $P_1$  for the  $(k+1)$ st time,  $u_3(k)$ . This all results, using (4.1):

$$y_1(k) = \max(u_1(k), u_3(k), x_2(k), y_2(k-1)). \quad (4.4)$$

The same can be done for the second product,  $P_2$ . This product leaves the system for the  $(k+1)$ st time if it is fed to the system for the  $(k+1)$ st time,  $u_2(k)$ , if it is buffered for the  $(k+1)$ st time,  $x_2(k+1)$ , if, according to the product mix,  $P_1$  has left the buffer for the  $(k+1)$ st time,  $y_1(k)$ , and if machine 1 can receive  $P_2$  for the  $(k+1)$ st time,  $u_4(k)$ . This all, including the substitution of (4.3) and (4.4), results in the following equation for  $y_2(k)$ :

$$y_2(k) = \max(u_1(k), u_2(k), u_3(k), u_4(k), x_2(k), y_2(k-1)). \quad (4.5)$$

Due to the last term in (4.4) and (4.5),  $y_2(k-1)$ , information of the previous step has to be saved to determine the two outputs  $y_1(k)$  and  $y_2(k)$ . Therefore, an extra state has to be added:

$$\begin{aligned} x_3(k+1) &= y_2(k) \\ &= \max(u_1(k), u_2(k), u_3(k), u_4(k), x_2(k), x_3(k)). \end{aligned} \quad (4.6)$$

Using (4.6), (4.4) and (4.5) become:

$$\begin{aligned} y_1(k) &= \max(u_1(k), u_3(k), x_2(k), x_3(k)) \\ y_2(k) &= \max(u_1(k), u_2(k), u_3(k), u_4(k), x_2(k), x_3(k)). \end{aligned}$$

The infinite buffer can always receive a product, which implies that the value of the outputs  $y_3(k)$  and  $y_4(k)$  (availability information stream) are equal to  $\varepsilon$  for all  $k$ 's:

$$\begin{aligned} y_3(k) &= \varepsilon \quad \forall k \\ y_4(k) &= \varepsilon \quad \forall k. \end{aligned}$$

Now the entire state-space description becomes:

$$\begin{aligned}\bar{x}(k+1) &= \begin{pmatrix} \cdot & 0 & \cdot \\ \cdot & 0 & \cdot \\ \cdot & 0 & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & 0 & 0 \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k) &= \begin{pmatrix} \cdot & 0 & 0 \\ \cdot & 0 & 0 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \cdot & 0 & \cdot \\ 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \bar{u}(k)\end{aligned}$$

where:

$$\bar{x}(k) = \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{pmatrix} \quad \bar{u}(k) = \begin{pmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \\ u_4(k) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ y_4(k) \end{pmatrix}.$$

### 4.3 Machines 1 and 2

The machines used in this case, are equal to the machines described in Section 3.3. They receive one product per feed. The process times are named  $d_1$  for machine 1 and  $d_2$  for machine 2. A schematic representation of the two machines can be seen in Figure 4.3.

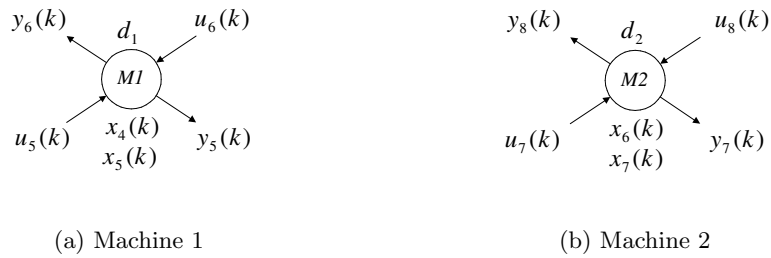


Figure 4.3: Single lot machines 1 and 2

The definitions can be seen in Section 3.3. The increased numbering of the states results in  $x_4(k)$  and  $x_5(k)$  for machine 1 and  $x_6(k)$  and  $x_7(k)$  for machine 2. The state-space description of the machine can be seen below. Here,  $d_i$  is the process time of machine  $i$ .

$$\begin{aligned}\bar{x}(k+1) &= \begin{pmatrix} d_i & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k) &= \begin{pmatrix} 2d_i & d_i \\ d_i & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d_i & 0 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes \bar{u}(k)\end{aligned}$$

with (for machine 1):

$$\bar{x}(k) = \begin{pmatrix} x_4(k) \\ x_5(k) \end{pmatrix} \quad \bar{u}(k) = \begin{pmatrix} u_5(k) \\ u_6(k) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_5(k) \\ y_6(k) \end{pmatrix}$$

and with (for machine 2):

$$\bar{x}(k) = \begin{pmatrix} x_6(k) \\ x_7(k) \end{pmatrix} \quad \bar{u}(k) = \begin{pmatrix} u_7(k) \\ u_8(k) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_7(k) \\ y_8(k) \end{pmatrix}.$$

#### 4.4 Finite buffer

As mentioned in Section 4.1 the finite buffer can only send and receive according to a certain product mix. This product mix is  $P_1$ ,  $P_2$ ,  $P_1$  etc. Due to this additional policy, the finite buffer as discussed in Section 3.4 cannot be used here. A new buffer has to be modelled. A schematic representation of this finite buffer can be seen in Figure 4.4.

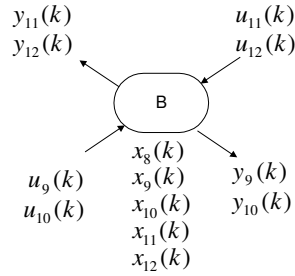


Figure 4.4: Finite buffer

with:

- $u_9(k)$ : time instant at which product  $P_1$  is fed to the buffer for the  $(k+1)$ st time,
- $u_{10}(k)$ : time instant at which product  $P_2$  is fed to the buffer for the  $(k+1)$ st time,

- $u_{11}(k)$ : time instant at which the processing unit, directly coupled to this processing unit, can receive product  $P_1$  for the  $(k + 1)$ st time,
- $u_{12}(k)$ : time instant at which the processing unit, directly coupled to this processing unit, can receive product  $P_2$  for the  $(k + 1)$ st time,
- $y_9(k)$ : time instant at which product  $P_1$  leaves the buffer for the  $(k + 1)$ st time,
- $y_{10}(k)$ : time instant at which product  $P_2$  leaves the buffer for the  $(k + 1)$ st time,
- $y_{11}(k)$ : time instant at which the buffer becomes available for the  $(k + 1)$ st time to receive product  $P_1$ ,
- $y_{12}(k)$ : time instant at which the buffer becomes available for the  $(k + 1)$ st time to receive product  $P_2$ ,
- $x_8(k)$ : time instant at which the buffer starts buffering product  $P_1$  for the  $k$ th time,
- $x_9(k)$ : time instant at which the buffer starts buffering product  $P_2$  for the  $k$ th time,
- $x_{10}(k)$ : time instant at which product  $P_1$  leaves the buffer for the  $(k)$ th time ( $= y_9(k - 1)$ ),
- $x_{11}(k)$ : time instant at which product  $P_2$  leaves the buffer for the  $(k)$ th time ( $= y_{10}(k - 1)$ ),
- $x_{12}(k)$ : time instant at which product  $P_2$  leaves the buffer for the  $(k - 1)$ st time ( $= y_{10}(k - 2)$ ).

The finite buffer is modelled using a similar way of modelling that is used to model a machine (Section 3.3), with the difference that the buffer has no process time. In other words,  $d_i = 0$ . Due to one of the main conditions (see Section 3.1), the deterministic process time has to be added in a certain situation, to a time instant (state). However, the addition of the process time is ignored in this section, since  $d_i = 0$ .

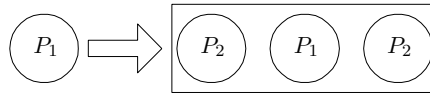


Figure 4.5: Finite buffer, situation 1

Due to the product mix, the determination of the states is more complicated than in the previously modelled structures. To calculate the first state,  $x_8(k)$ , a schematic representation of the situation is needed. If the buffer has to receive a product of type 1, the situation in the buffer can be seen in Figure 4.5. In Figure 4.5 it can be seen,

that two products of type 2 and one product of type 1 are inside the buffer. The buffer can only start buffering a product of type 1 for the  $(k + 1)$ st time,  $x_8(k + 1)$ , if its raw material has been fed to the system for the  $(k + 1)$ st time,  $u_9(k)$ , if the buffer has buffered product type 2, for the  $k$ th time,  $x_9(k)$  (due to its product mix) and if the last product in the buffer at that moment,  $P_2$ , see Figure 4.5, leaves the buffer for the  $(k - 2)$ nd time,  $y_{10}(k - 2)$ . Now,  $x_8(k + 1)$  becomes:

$$x_8(k + 1) = \max(u_9(k), x_9(k), y_{10}(k - 2)). \quad (4.7)$$

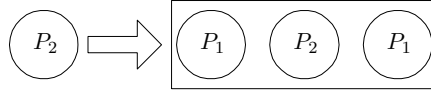


Figure 4.6: Finite buffer, situation 2

The same can be done for the second product. The buffer can only start buffering a product of type 2 for the  $(k + 1)$ st time,  $x_9(k + 1)$ , if its raw material has been fed to the system for the  $(k + 1)$ st time,  $u_{10}(k)$ , if the buffer has buffered product type 1, for the  $(k + 1)$ st time,  $x_8(k + 1)$  (due to its product mix) and if the last product in the buffer at that moment,  $P_1$ , see Figure 4.6, leaves the buffer for the  $(k - 1)$ st time,  $y_9(k - 1)$ . Now:

$$x_9(k + 1) = \max(u_9(k), u_{10}(k), x_9(k), y_{10}(k - 2), y_9(k - 1)). \quad (4.8)$$

Product  $P_1$  leaves the finite buffer for the  $(k + 1)$ st time,  $y_9(k)$  if it is fed to the buffer for the  $(k + 1)$ st time,  $u_9(k)$ , if it is buffered for the  $(k + 1)$ st time,  $x_8(k + 1)$ , if the batch machine can receive  $P_1$  for the  $(k + 1)$ st time,  $u_{11}(k)$  and if, according to its product mix,  $P_2$  has been sent away for the  $k$ th time,  $y_{10}(k - 1)$ . This all, including the substitution of (4.7) results in:

$$y_9(k) = \max(u_9(k), u_{11}(k), x_9(k), y_{10}(k - 1), y_{10}(k - 2)). \quad (4.9)$$

A similar approach can be used to calculate output  $y_{10}(k)$ . Product  $P_2$  leaves the finite buffer for the  $(k + 1)$ st time,  $y_{10}(k)$ , if it is fed to the buffer for the  $(k + 1)$ st time,  $u_{10}(k)$ , if it is buffered for the  $(k + 1)$ st time,  $x_9(k + 1)$ , if the batch machine can receive  $P_2$  for the  $(k + 1)$ st time,  $u_{12}(k)$  and if, according to its product mix,  $P_1$  has been sent away for the  $(k + 1)$ st time,  $y_9(k)$ . With substitution of (4.8) and (4.9), this gives:

$$y_{10}(k) = \max(u_9(k), u_{10}(k), u_{11}(k), u_{12}(k), x_9(k), y_{10}(k - 1), y_{10}(k - 2), y_9(k - 1)). \quad (4.10)$$



Due to the last terms in (4.7) to (4.10),  $y_{10}(k-2)$  and  $y_9(k-1)$ , three extra states are needed to save the information of previous steps:

$$\begin{aligned} x_{10}(k+1) &= y_9(k) \\ &= \max(u_9(k), u_{11}(k), x_9(k), x_{11}(k), x_{12}(k)) \end{aligned} \quad (4.11)$$

$$\begin{aligned} x_{11}(k+1) &= y_{10}(k) \\ &= \max(u_9(k), u_{10}(k), u_{11}(k), u_{12}(k), x_9(k), x_{10}(k), \\ &\quad x_{11}(k), x_{12}(k)) \end{aligned} \quad (4.12)$$

$$x_{12}(k+1) = x_{11}(k). \quad (4.13)$$

Substitution of (4.11) to (4.13) in (4.7) to (4.10) results in :

$$\begin{aligned} x_8(k+1) &= \max(u_9(k), x_9(k), x_{11}(k)) \\ x_9(k+1) &= \max(u_9(k), u_{10}(k), x_9(k), x_{10}(k), x_{12}(k)). \end{aligned}$$

and:

$$\begin{aligned} y_9(k+1) &= \max(u_9(k), u_{11}(k), x_9(k), x_{11}(k), x_{12}(k)) \\ y_{10}(k+1) &= \max(u_9(k), u_{10}(k), u_{11}(k), u_{12}(k), x_9(k), x_{10}(k), \\ &\quad x_{11}(k), x_{12}(k)). \end{aligned}$$

Now, the availability information has to be determined. The finite buffer can receive a product of type 1 for the  $(k+1)$ st time,  $y_{11}(k)$  if a product of type 2 has been sent away for the  $(k-2)$ nd time,  $y_{10}(k-2)$  (see Figure 4.5). The buffer can receive a product of type 2 for the  $(k+1)$ st time,  $y_{12}(k)$ , if a product of type 1 has been sent away for the  $(k-1)$ st time,  $y_9(k-1)$  (see Figure 4.6). This results in the following equations:

$$\begin{aligned} y_{11}(k) &= y_{10}(k-2) = x_{12}(k) \\ y_{12}(k) &= y_9(k-1) = x_{10}(k). \end{aligned}$$

Now the entire state-space description of the form (3.1) becomes:

$$\begin{aligned}\bar{x}(k+1) &= \begin{pmatrix} . & 0 & . & . & 0 \\ . & 0 & 0 & . & 0 \\ . & 0 & . & 0 & 0 \\ . & 0 & 0 & 0 & 0 \\ . & . & . & 0 & . \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & . & . & . \\ 0 & 0 & . & . \\ 0 & . & 0 & . \\ 0 & 0 & 0 & 0 \\ . & . & . & . \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k) &= \begin{pmatrix} . & 0 & . & 0 & 0 \\ . & 0 & 0 & 0 & 0 \\ . & . & . & . & 0 \\ . & . & 0 & . & . \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & . & 0 & . \\ 0 & 0 & 0 & 0 \\ . & . & . & . \\ . & . & . & . \end{pmatrix} \otimes \bar{u}(k)\end{aligned}$$

where:

$$\bar{x}(k) = \begin{pmatrix} x_8(k) \\ x_9(k) \\ x_{10}(k) \\ x_{11}(k) \\ x_{12}(k) \end{pmatrix} \quad \bar{u}(k) = \begin{pmatrix} u_9(k) \\ u_{10}(k) \\ u_{11}(k) \\ u_{12}(k) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_9(k) \\ y_{10}(k) \\ y_{11}(k) \\ y_{12}(k) \end{pmatrix}.$$

## 4.5 Batch machine

The last machine in the manufacturing system is the batch machine. As mentioned in Section 4.1, the batch size of this machine is two. The process time equals  $d_3$  time units. The machine produces the products  $P_1$  and  $P_2$  in one batch. After the products have been processed, the machine sends them individually, in the standard product mix,  $P_1, P_2, P_1$ , etc., to the output of the system. The batch machine can only receive new products if both products have left the machine. The output can always receive products. A schematic illustration can be seen in Figure 4.7.

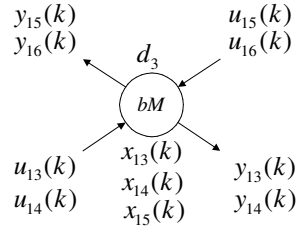


Figure 4.7: Batch machine

with:

- $u_{13}(k)$ : time instant at which product  $P_1$  is fed to the batch machine for the  $(k+1)$ st time,
- $u_{14}(k)$ : time instant at which product  $P_2$  is fed to the batch machine for the  $(k+1)$ st time,
- $u_{15}(k)$ : time instant at which the free output can receive product  $P_1$  for the  $(k+1)$ st time,
- $u_{16}(k)$ : time instant at which the free output can receive product  $P_2$  for the  $(k+1)$ st time,
- $y_{13}(k)$ : time instant at which finished product  $P_1$  leaves the batch machine for the  $(k+1)$ st time,
- $y_{14}(k)$ : time instant at which finished product  $P_2$  leaves the batch machine for the  $(k+1)$ st time,
- $y_{15}(k)$ : time instant at which the batch machine becomes available for the  $(k+1)$ st time to receive product  $P_1$ ,
- $y_{16}(k)$ : time instant at which the batch machine becomes available for the  $(k+1)$ st time to receive product  $P_2$ ,
- $x_{13}(k)$ : time instant at which the batch machine starts processing a batch containing  $P_1$  and  $P_2$  for the  $(k+1)$ st time,
- $x_{14}(k)$ : time instant at which finished product  $P_1$  leaves the batch machine for the  $(k)$ th time  $(= y_{13}(k-1))$ ,
- $x_{15}(k)$ : time instant at which finished product  $P_2$  leaves the batch machine for the  $(k)$ th time  $(= y_{14}(k-1))$ .

The machine can start processing a new batch for the  $(k+1)$ st time,  $x_{13}(k+1)$ , if the products  $P_1$  and  $P_2$  are fed for the  $(k+1)$ st time,  $u_{13}(k)$  and  $u_{14}(k)$ , if the previous batch has been processed,  $x_{13}(k) + d_3$  and if both processed products have left the batch machine for the  $k$ th time,  $y_{13}(k-1)$  and  $y_{14}(k-1)$ . This results in the following equation:

$$x_{13}(k+1) = \max(u_{13}(k), u_{14}(k), x_{13}(k) + d_3, y_{13}(k-1), y_{14}(k-1)). \quad (4.14)$$

As mentioned in the beginning of this section, the batch machine sends the products to the output in the standard product mix. Product  $P_1$  leaves the machine for the  $(k+1)$ st time,  $y_{13}(k)$ , if the machine processed the batch for the  $(k+1)$ st time,  $x_{13}(k+1) + d_3$ , if the output can receive  $P_1$  for the  $(k+1)$ st time,  $u_{15}(k)$  and if, according to the product mix,  $P_2$  has left the batch machine for the  $(k)$ th time,  $y_{14}(k-1)$ . Now,  $y_{13}(k)$  becomes:

$$\begin{aligned}
y_{13}(k) = & \max(u_{13}(k) + d_3, u_{14}(k) + d_3, u_{15}(k), x_{13}(k) + 2d_3, \\
& y_{13}(k-1) + d_3, y_{14}(k-1) + d_3).
\end{aligned} \tag{4.15}$$

A similar approach is used to determine  $y_{14}(k)$ . Product  $P_2$  leaves the machine for the  $(k+1)$ st time,  $y_{14}(k)$ , if the machine processed the batch for the  $(k+1)$ st time,  $x_{13}(k+1) + d_3$ , if the output can receive  $P_2$  for the  $(k+1)$ st time,  $u_{16}(k)$  and if, according to the product mix,  $P_1$  has left the batch machine for the  $(k+1)$ st time,  $y_{13}(k)$ . This all, including the substitution of (4.14), results in the following equation for  $y_{14}(k)$ :

$$\begin{aligned}
y_{14}(k) = & \max(u_{13}(k) + d_3, u_{14}(k) + d_3, u_{15}(k), u_{16}(k), x_{13}(k) + 2d_3, \\
& y_{13}(k-1) + d_3, y_{14}(k-1) + d_3).
\end{aligned} \tag{4.16}$$

Due to the items  $y_{13}(k-1)$  and  $y_{14}(k-1)$  in (4.14) to (4.16), two extra states are necessary:

$$x_{13}(k+1) = y_{13}(k) \tag{4.17}$$

$$x_{14}(k+1) = y_{14}(k). \tag{4.18}$$

Substitution of (4.17) and (4.18) in (4.14) to (4.16) gives:

$$x_{13}(k+1) = \max(u_{13}(k), u_{14}(k), x_{13}(k) + d_3, x_{14}(k), x_{15}(k)). \tag{4.19}$$

and:

$$\begin{aligned}
y_{13}(k) = & \max(u_{13}(k) + d_3, u_{14}(k) + d_3, u_{15}(k), x_{13}(k) + 2d_3, \\
& x_{14}(k) + d_3, x_{15}(k) + d_3) \\
y_{14}(k) = & \max(u_{13}(k) + d_3, u_{14}(k) + d_3, u_{15}(k), u_{16}(k), x_{13}(k) + 2d_3, \\
& x_{14}(k) + d_3, x_{15}(k) + d_3).
\end{aligned}$$

The availability information stream can be calculated as follows. The batch machine can receive a product  $P_1$  for the  $(k+1)$ st time,  $y_{15}(k)$ , if product  $P_2$  (and automatically, according to the policy,  $P_1$ ) has left the machine for the  $k$ th time,  $y_{14}(k-1)$ , or  $x_{15}(k)$ .

For  $P_2$ , the same approach is used. The machine can receive a product of type 2 for the  $(k+1)$ st time,  $y_{16}(k)$ , if a product  $P_2$  has left the machine for the  $k$ th time,  $y_{14}(k-1)$ , or  $x_{15}(k)$ . This results in the following equations:

$$\begin{aligned} y_{15}(k) &= y_{14}(k-1) = x_{15}(k) \\ y_{16}(k) &= y_{14}(k-1) = x_{15}(k). \end{aligned}$$

Now, the state-space description of the entire batch machine of the form (3.1) becomes:

$$\begin{aligned} \bar{x}(k+1) &= \begin{pmatrix} d_3 & 0 & 0 \\ 2d_3 & d_3 & d_3 \\ 2d_3 & d_3 & d_3 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} 0 & 0 & . & . \\ d_3 & d_3 & 0 & . \\ d_3 & d_3 & 0 & 0 \end{pmatrix} \otimes \bar{u}(k) \\ \bar{y}(k) &= \begin{pmatrix} 2d_3 & d_3 & d_3 \\ 2d_3 & d_3 & d_3 \\ . & . & 0 \\ . & . & 0 \end{pmatrix} \otimes \bar{x}(k) \oplus \begin{pmatrix} d_3 & d_3 & 0 & . \\ d_3 & d_3 & 0 & 0 \\ . & . & . & . \\ . & . & . & . \end{pmatrix} \otimes \bar{u}(k) \end{aligned}$$

where:

$$\bar{x}(k) = \begin{pmatrix} x_{13}(k) \\ x_{14}(k) \\ x_{15}(k) \end{pmatrix} \quad \bar{u}_k = \begin{pmatrix} u_{13}(k) \\ u_{14}(k) \\ u_{15}(k) \\ u_{16}(k) \end{pmatrix} \quad \bar{y}(k) = \begin{pmatrix} y_{13}(k) \\ y_{14}(k) \\ y_{15}(k) \\ y_{16}(k) \end{pmatrix}.$$

## 4.6 Coupling

The individual modules (structures) still should be coupled to obtain the overall system dynamics, which is done by connecting the outputs to the corresponding inputs. The structures have to be coupled using the following coupling equations:

$$\begin{aligned} u_3(k) &= y_6(k) & u_9(k) &= y_5(k) \\ u_4(k) &= y_8(k) & u_{10}(k) &= y_7(k) \\ u_5(k) &= y_1(k) & u_{11}(k) &= y_{15}(k) \\ u_6(k) &= y_{11}(k) & u_{12}(k) &= y_{16}(k) \\ u_7(k) &= y_2(k) & u_{13}(k) &= y_9(k) \\ u_8(k) &= y_{12}(k) & u_{14}(k) &= y_{10}(k). \end{aligned}$$

This substitution, which is done by hand, is not presented explicitly. The state-space description of the entire manufacturing system can be written down. This model has a

dimension of fifteen, has four inputs,  $u_1(k)$ ,  $u_2(k)$ ,  $u_{15}(k)$ , and  $u_{16}(k)$ , and four outputs,  $y_1(k)$ ,  $y_2(k)$ ,  $y_{15}(k)$ , and  $y_{16}(k)$ . The system matrices  $A$ ,  $B$ ,  $C$ , and  $D$  can be seen in Appendix C.

Now the state-space description is known, simulations can be done using Matlab. This program uses a max-plus toolbox with max-plus functions to calculate max-plus addition and max-plus multiplication [Sta03]. To simulate and analyse this case, initial values are needed. Nine iterations are done to keep the notation of the products simple (see Figures 4.8 and 4.9). The first product of type 1 is presented as  $P_{11}$  and the ninth product is presented as  $P_{19}$ . Here, the input time instants  $u(k)$  are given. Using this input vector, the time instants at which the products leave the system,  $y(k)$  and the time instants at which machines (buffers) start processing (buffering) can be determined. In the next section the simulation results are presented and the max-plus model is validated using the formalism  $\chi$ .

## 4.7 Output explanation and validation

Before the model of this manufacturing system can be analyzed, first validation has to be done. According to Kleijnen [Kle92] validation is concerned with determining that a simulation model (as opposed to the computer program) is an accurate representation of the system under study. This validation is done using a simple test, such as graphical analysis, and the relationship between the simulations and other models (for example the results obtained by calculation by hand and a  $\chi$  model). Using these validation methods, the model should be valid under 'extreme' conditions [Kle92]. Now the initial conditions, the input time instants and the availability information are known, the time instants at which the machines and buffers start processing respectively buffering products, can be calculated by hand using the standard state-space description (3.1). The results of these calculations have to be compared with the results of the simulation. If these results are equal, the model behaves as expected and desired, and is validated successfully.

Aside from the calculations by hand, the max-plus model is validated with the formalism  $\chi$ , which is a specification language designed for describing real-world concurrent systems [Hof02]. In this report, two  $\chi$  models are made to validate the max-plus model. In the first  $\chi$  model, which can be seen in Appendix E.1, the manufacturing system of the theoretical case is modelled using the standard modelling techniques according to [Hof02]. In the second  $\chi$  model, the manufacturing system is modelled according to the actions and events that take place in the max-plus model (see Appendix E.2). The validation of the max-plus models is done using two different sessions of in- and output sequences. In the first session, the input sequence is non-decreasing and the output of the system can always receive products, see Table 4.1. The second session contains all kind of special situations in the input sequence and the output cannot always receive products, see Table 4.2. First the calculations by hand are presented, followed

by simulation results of the  $\chi$  model. Then, the results are compared to the results of the max-plus model. If the results of the calculation by hand, the  $\chi$  models and the max-plus model are equal, the max-plus model is validated successfully.

To simulate this manufacturing system, system matrices  $A$ ,  $B$ ,  $C$ , and  $D$  of the state-space description are necessarily. These can be seen in Appendix C. To calculate the time instants of step  $k + 1$ , information of the previous step,  $k$ , is needed. To calculate values for  $k = 1$ , the initial state values (for  $k = 0$ ),  $\bar{x}(0)$ , are necessary. These initial states, input time instants of raw material ( $u_1(k)$  and  $u_2(k)$ ), and output time instants of availability information, ( $u_{15}(k)$  and  $u_{16}(k)$ ) enables simulation of this model. Their values for  $k = 1$  to  $k = 9$  can be seen in Table 4.1. The free output can always receive products, which results in  $u_{15}(k) = u_{16}(k) = \varepsilon \forall k$ . The process time of machine 1, machine 2 and the batch machine are respectively 1, 3, and 10 time units. The initial state vector is:  $\bar{x}(0) = x_0 = [\varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ \varepsilon]^T$ . This initial state vector indicates an empty manufacturing system. If the manufacturing system is not empty another initial state vector has to be chosen. As mentioned before, the validation of the max-plus model is done using two different sessions of in- and output sequences. These sessions can be seen in Tables 4.1 and 4.2.

The first session of in- and output sequences do not need an extensive explanation. The input sequence is non-decreasing and the output is a free output, which means that it can always receive products.

systemfeed nr.	1	2	3	4	5	6	7	8	9
$u_1(k)$	0	5	10	15	20	25	30	35	40
$u_2(k)$	1	6	11	16	21	26	31	36	41
$u_{15}(k)$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$
$u_{16}(k)$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$

Table 4.1: Input and output values of session 1

The second session contains all kind of input sequences and at one system feed, the output can only receive products at a certain time instant. A short description of the values in Table 4.2 is given here, where  $u(k + 1)$  is assumed to takes place later than  $u(k)$ . Three situations in which this is not true, are simulated, to check the correctness of the model, for instance:  $u_1(k) < u_1(k - 1)$ . This situation takes place in feed nr. 3;  $u_1(3) = 0$  and  $u_1(2) = 5$ . The second situation is  $u_2(k) < u_2(k - 1)$ . This situation takes place in feed nr. 5;  $u_2(5) = 11$  and  $u_2(4) = 16$ . Third,  $u_1(k) < u_2(k - 1)$ , this situation takes place in feed nr. 7;  $u_1(7) = 21$  and  $u_2(6) = 26$ . The last situation represents that the inputs  $u_{15}(k)$  and  $u_{16}(k)$  are not equal to  $\varepsilon$  and  $u_{15}(k) \neq \varepsilon$ ,  $u_{16}(k) \neq \varepsilon$ . In this case, the batch machine can not send its products immediately to the output. This situation takes place in feed nr. 9;  $u_{15}(9) = u_{16}(9) = 100$ . These input values can be seen in Table 4.2.

Now the results of the calculations by hand, the  $\chi$  model and the max-plus model are discussed.

systemfeed nr.	1	2	3	4	5	6	7	8	9
$u_1(k)$	0	5	0	15	20	25	21	35	40
$u_2(k)$	0	6	11	16	11	26	31	36	41
$u_{15}(k)$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	100
$u_{16}(k)$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	100

Table 4.2: Input and output values session 2

### Results of the calculations by hand

Here, the results of the calculations by hand are determined. This is done using the values of both Tables 4.1 and 4.2 and a lot-time diagram. A short description of the computation of the events at the resources is illustrated subsequently. The results can be seen in Figure 4.8 and 4.9 and Tables 4.3 and 4.5 for respectively session 1 and session 2.

system feed nr.	1	2	3	4	5	6	7	8	9
$x_1(k)$	0	5	10	15	20	25	30	35	40
$x_2(k)$	1	6	11	16	21	26	31	36	41
$x_4(k)$	0	5	10	15	20	25	34	44	54
$x_6(k)$	1	6	11	16	24	34	44	54	64
$x_8(k)$	1	6	11	16	24	34	44	54	64
$x_9(k)$	4	9	14	24	34	44	54	64	74
$x_{13}(k)$	4	14	24	34	44	54	64	74	84

Table 4.3: Time instants of start processing/buffering, session 1

system feed nr.	1	2	3	4	5	6	7	8	9
$y_{13}(k)$	14	24	34	44	54	64	74	84	94
$y_{14}(k)$	14	24	34	44	54	64	74	84	94

Table 4.4: Time instants at which products leave the system, session 1

The first raw material is fed to the system at time instant 0 (for  $P_1$ ) and 1 (for  $P_2$ ), see Table 4.1. Consequently, the infinite buffer starts buffering  $P_1$  and  $P_2$  at time instant 0 and 1 respectively. Machines 1 and 2 are idle at time instant 0. Therefore, they start processing the products at time instant 0 and 1 respectively. Machine 1 has a process time of 1 time unit. This means that  $P_1$  is sent to the finite buffer at time instant 1. The buffer starts buffering  $P_1$  at time instant 1. The process time of machine 2 is 3 time units. This means that  $P_2$  has been finished at time instant 4 and is buffered by the finite buffer immediately. Now two products of different type are in the buffer. As long as the batch machine is idle, these products are sent to the batch machine. Therefore, this machine starts processing the batch at time instant 4. The process time of the batch machine is 10 time units, which means that the products can leave the system at



time instant 14. The new products are fed to the system at time instant 5 (for  $P_1$ ) and 6 (for  $P_2$ ). Machine 1 and 2 receive the products immediately because they are idle at that time. Machine 1 finishes  $P_1$  at time instant 6 and the finite buffer start buffering this product at the same time. Machine 2 finishes  $P_2$  at 9 according to its process time. The batch machine can start processing the new batch if the previous batch has been finished, at time instant 14. This all can be validated using Figure 4.8 and Tables 4.3 and 4.4.

These calculated results can be seen in Table 4.3. States  $x_1(k)$  and  $x_2(k)$  are the time instants at which the infinite buffer starts buffering respectively  $P_1$  and  $P_2$  for the  $k$ th time. States  $x_4(k)$  and  $x_6(k)$  are the time instants at which respectively machine 1 starts processing  $P_1$  and machine 2 starts working on  $P_2$  for the  $k$ th time. States  $x_8(k)$  and  $x_9(k)$  are the time instants at which the finite buffer starts buffering respectively  $P_1$  and  $P_2$  for the  $k$ th time. State  $x_{13}(k)$  is the time instant at which the batch machine starts processing a batch ( $P_1$  and  $P_2$ ) for the  $k$ th time.

The time instants at which the products leave the manufacturing system,  $y_{13}(k)$  for product  $P_1$  and  $y_{14}(k)$  for product  $P_2$  are presented in Table 4.4:

The values in Tables 4.1, 4.3, and 4.4 do not give a clear view of the events that occur during the production of the two products. A graphic representation is needed to make this output understandable. This can be done by using a lot-time diagram. In these diagrams, one can see when and on what machine/buffer products are processed/buffered to become a finished product. Lot-time diagrams are simple to understand and easy to construct. Therefore, the results calculated by hand are represented in a lot-time diagram chart, see Figure 4.8.

In the lot-time diagram, the horizontal axis represents time and the vertical axis represents the product type and number. For instance, 14 signifies the fourth product of type 1 ( $P_1$ ).

For the second session, the same approach as mentioned above, is used to determine the start and end time instants of all events. Now, the input sequence and availability information can be seen in Table 4.2. Tables 4.6 and 4.5 and the lot-time diagram (Figure 4.9) illustrate the results of these input values.

system feed nr.	1	2	3	4	5	6	7	8	9
$x_1(k)$	0	5	6	15	20	25	26	35	40
$x_2(k)$	0	6	11	16	20	26	31	36	41
$x_4(k)$	0	5	6	15	20	25	33	43	53
$x_6(k)$	0	6	11	16	23	33	43	53	63
$x_8(k)$	1	6	9	16	23	33	43	53	63
$x_9(k)$	3	9	14	23	33	43	53	63	73
$x_{13}(k)$	3	13	23	33	43	53	63	73	83

Table 4.5: Time instants of start processing/buffering, session 2

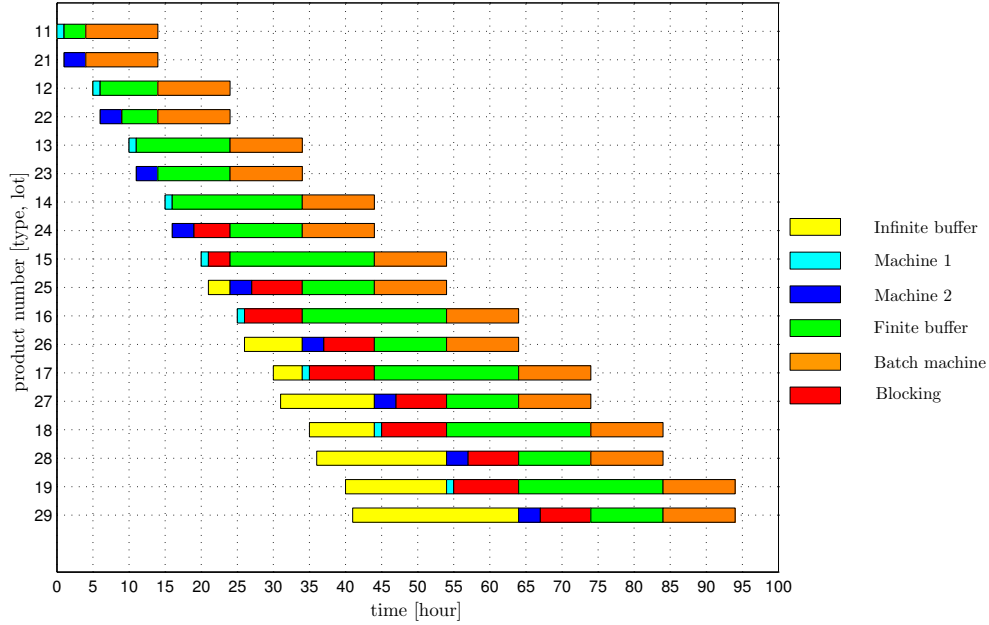


Figure 4.8: Lot-time diagram, session 1

system feed nr.	1	2	3	4	5	6	7	8	9
$y_{13}(k)$	13	23	33	43	53	63	73	83	100
$y_{14}(k)$	13	23	33	43	53	63	73	83	100

Table 4.6: Time instants at which products leave the system, session 2

In Figure 4.9 can be seen that in the first situation ( $u_1(k) < u_1(k-1)$ ) machine 1 starts working immediately on the third  $P_1$  when the second  $P_1$  has been finished. In the second situation ( $u_2(k) < u_2(k-1)$ ), the fifth product of type two is fed to the system when machine 2 is still processing. Therefore, this product is first buffered in the infinite buffer. A similar situation exist for  $u_1(k) < u_2(k-1)$ , here the seventh product of type one has to wait in the finite buffer until machine 1 becomes idle. The last situation can be seen in the last products (ninth feed). Here the products leave the batch machine at time instant 100, a result of the values of  $u_{15}(9)$  and  $u_{16}(9)$ .

### Results of the $\chi$ and max-plus model

In the previous part of this chapter, the results of both input sequences and availability information are presented. These results are shown using a lot-time diagram. Both the max-plus model, see Appendix C for the system matrices  $A$ ,  $B$ ,  $C$ , and  $D$ , and the  $\chi$  models, see Appendix E, return data that has to be treated first before lot-time diagrams can be made. The returned data are the time instants at which events occur.

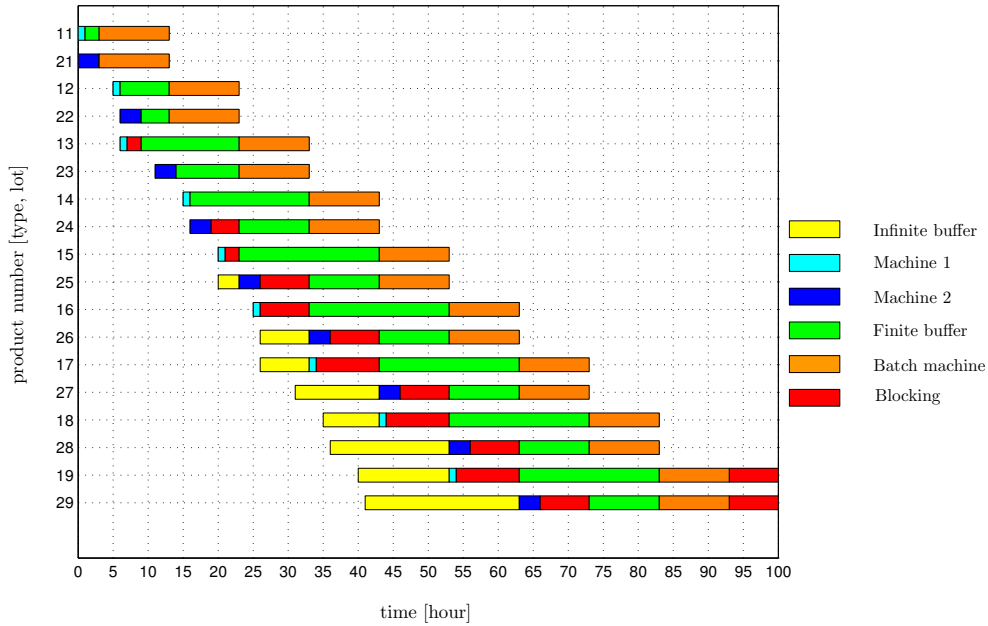


Figure 4.9: Lot-time diagram, session 2

Using this information, the lot-time diagrams can be made. The  $\chi$  models are made using the inputs as can be seen in Tables 4.1 and 4.2. The returned data of both the  $\chi$  files after simulations is used to make lot-time diagrams with Matlab. The diagrams of both these files (see Appendix 4.7) are equal to the diagrams as can be seen in Figure 4.8 and 4.9.

The max-plus model is implemented in Matlab. To simulate, first the max-plus toolbox of Stanczyk [Sta03], and the values as presented in Tables 4.1 and 4.2 are needed. Similar to the  $\chi$  model the returned data by the model is used to make the lot-time diagrams with Matlab. These lot-time diagrams are equal to the diagrams that are presented in Figure 4.8 and 4.9. All three methods, (1) calculations by hand, (2a)  $\chi$  specifications using a standard file and (2b) a specified file that corresponds to the max-plus dynamics and (3) the max-plus model return the same results after simulations. Therefore, the max-plus model is validated successfully. In Chapter 5 this model is used to control the output sequence with respect to a certain reference signal in combination with Model Predictive Control (MPC).

## 4.8 Discussion

In this chapter a manufacturing system with all sorts of structures and policies has been modelled using the max-plus-algebra and the formalism  $\chi$ . Both the max-plus-algebra

and  $\chi$  can serve as a tool to model and analyse DESs. As mentioned in Chapter 1, computer simulation (e.g. the formalism  $\chi$ ) is, up to now, the most widely used technique to study DESs. In this section, the max-plus-algebra is analyzed and compared to the  $\chi$  language.

Using both the max-plus-algebra and  $\chi$  to model a manufacturing system result in models that describe the behavior (both the transient and steady state) of this DES exactly. The best way to study a manufacturing system with its structures and policies using both the max-plus-algebra and the  $\chi$  language, is to cut the system into several small pieces. The smallest piece of the entire puzzle (or module) is an individual modules (e.g. a finite buffer) with a certain policy (e.g. FIFO). If all these individual pieces are modelled into both methods, coupling takes places. This coupling results in a max-plus model of the entire manufacturing system. Now, the entire manufacturing system is, for the max-plus-algebra written in two matrix formulations, called the standard max-algebraic state-space description, and in case of  $\chi$  in a formal language specification. One can gain, in case of a relative simple manufacturing system, good insight in its dynamics and structure by using the elegant max-plus state-space description, that contains four system matrices  $A$ ,  $B$ ,  $C$  and  $D$ . Unfortunately, this insight decreases if the size and/or complexity of the manufacturing system increases due to the increasing size of the system matrices. Here, complexity can depend on e.g. certain scheduling rules, the structure of the system, re-entrancy etc. If a manufacturing system is modelled using  $\chi$ , these (relative) complex situations can be modelled with functions outside the process environment. Large structures that contain similar processes can be modelled by defining the process only once and duplicate it in the *xper*-environment or use clusters. However, even the use of functions and the duplication of processes in the *xper*-environment in the  $\chi$  model might also lead to a decrease of insight of a manufacturing system.

If the input time instants of raw material are known, the time instants at which finished products leave the system and time instants at which machines or buffers start processing respectively buffering are easy to compute using the standard state-space description of a max-plus model. Of course, this advantage is only valid if deterministic process times are used. Using  $\chi$ , this simple, elegant calculation with a matrix formulation is not possible. These calculations can be done by hand if the system matrices are known. The formalism  $\chi$  requires simulations to determine start and end times of events. This formalism makes it possible to model machines with stochastic process times. Due to the limited duration of this project, these stochastic process times are not taken into account. The use of max-plus-algebra requires (in contrast to  $\chi$ ), up to now, that the route of the products through the manufacturing system has to be fixed. In Chapter 6 some recommendations about these topics are given.

In this report, several basic structures are modelled in the max-plus-algebra. In industry all kind of policies and structures are used. This means that these basic structures cannot always be used. Adapting these basic max-plus structures or modelling new structures from the starts is time consuming. A  $\chi$  model of a manufacturing system

can, in some cases, easily be adapted, if the policy and/or structure of a manufacturing system changes. If a max-plus model is used to analyse a manufacturing system, a small policy change means that this particular structure has to be modelled all over again. If the structure contains the new policy, all structures have to be coupled again before new analysis is possible. If, for example the number of buffer places of a finite buffer changes, in the max-plus-algebra, the buffer has to be modelled all over again. In case of using  $\chi$ , this only is a matter of changing a parameter. In Appendix A, an algorithm is presented to calculate the system matrices if a machine or buffer receive more than one product per feed. If the use of the max-plus-algebra to model manufacturing system can be explored, more algorithms can be found to determine system matrices in all sorts of situations. This automation might increase the suitability of the max-plus-algebra to model and analyze manufacturing systems.

Summarized, the max-plus-algebra is not useful if simulations are required to determine the influence on certain parameters or the performance of a manufacturing system with respect to for instance the number of machines or the number of buffer places. Manufacturing systems with a non-fixed product route and/or stochastic process times are not considered to model with the max-plus-algebra. Therefore, some recommendation with respect to these topics are given in Chapter 6.



## Chapter 5

# Control of a manufacturing system

In the previous chapters, the max-plus-algebra has been discussed in detail, applied to model different structures and an entire manufacturing system with all kind of structures and policies has been modelled and analyzed. One of the main purposes of this research project is to control a manufacturing system that is modelled using the max-plus-algebra. A control strategy may be described as the set of rules defining at which conditions, which controlled event should take place. An example of such an event may be the release of lots into the system. The control strategy that is used in this project is Model Predictive Control (MPC). MPC is a control strategy that is widely used in the process industry. Normally MPC uses linear discrete-time models for the process that has to be controlled. Van den Boom and De Schutter have extended MPC to max-plus-linear systems [Sch00a, Sch01]. In this chapter, an introduction to this max-plus extended MPC is given, followed by an implementation of MPC applied to the theoretical case of Chapter 4.

### 5.1 Model Predictive Control

The control strategy MPC is named *model-based* because it uses an internal dynamical model of the process that has to be controlled and it is *predictive* because this internal model is used to predict the future behavior of the process. Based on this prediction, an objective function is optimized with regard to the future control inputs of the process. MPC is optimal with respect to this chosen objective function. Only the first input of the generated optimal inputs over the future horizon is implemented for the next sample. After this implementation, the same procedure is repeated during this next sample. This mechanism is called the moving or receding horizon strategy.

Now the concept of MPC is discussed in more detail. MPC uses two horizons, the

prediction horizon and the control horizon, with respectively a length of  $N_p$  and  $N_c$  samples. The length of the control horizon is smaller than or equal to the prediction horizon due to a reduction of optimization variables. This results in less computation time, a smoother controller signal (because of the emphasis on the average behavior rather than on aggressive noise reduction) and a stabilizing effect (since the output signal is forced to its steady state value) [Sch01]. The use of a finite prediction horizon distinguishes MPC from standard feedback control. This prediction horizon allows MPC to take a control action at the current time step, in response to a possible future error between the reference and the actual output, even if the error is zero at that time [Ess02].

An important advantage of MPC is its constraint handling capability. All kinds of constraints (e.g. physical, safety and performance constraints) on the input(s), output(s) and state(s) can be taken into account. The use of MPC has some disadvantages. First, a model of the process that has to be controlled is needed. The performance of MPC depends on the accuracy of the available model. Secondly, the use of a suitable computer is necessary because MPC can be computationally demanding. Thirdly, the performance of MPC is also strongly dependent on the tuning parameters including weighting factors/parameters, and the length of both the control and prediction horizon. A schematic representation of the discussed MPC concept and its receding horizon is given in Figure 5.1. In contrast to the theoretical case, for reasons of clarity a Single-Input-Single-Output (SISO) system is considered here. The manufacturing system that has been worked out in Chapter 4 has four in- and outputs.

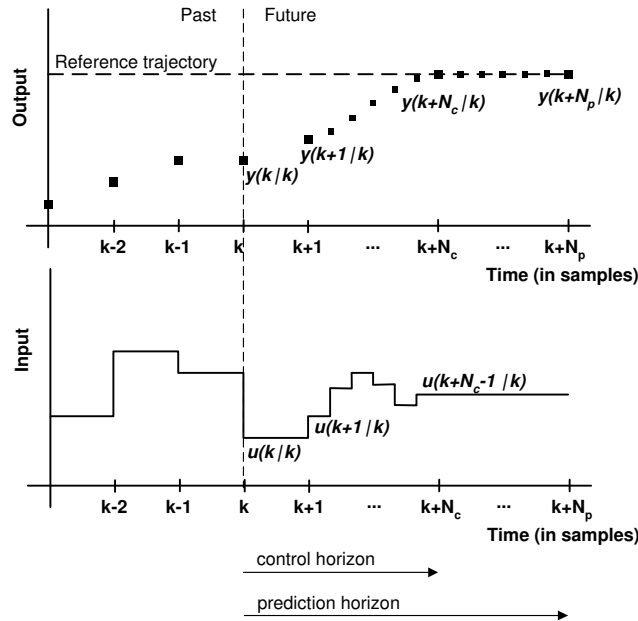


Figure 5.1: Concept of MPC



In the previous chapters, many structures of manufacturing systems have been modelled using the standard state-space description (3.1). At the present sample  $k$ , output  $y(k|k)$  is known, and the response of output  $y$  over the future prediction horizon is predicted:

$$\tilde{y}(k) = \begin{pmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+N_p|k) \end{pmatrix}$$

Here,  $\hat{y}(k+j|k)$  and  $\tilde{y}(k)$  stand for respectively the predicted value of the output  $y$  at sample  $k+j$  based on information that is available at sample  $k$  and the vector that contains these values. The prediction  $\hat{y}(k+j|k)$  is based on:

- the past input  $u(k|k)$ ,
- the current internal model state  $x(k|k)$ ,
- the proposed future inputs over the control horizon:

$$\bar{u}(k) = \begin{pmatrix} u(k+1|k) \\ \vdots \\ u(k+N_c|k) \end{pmatrix},$$

- the future reference signal  $\bar{r}(k)$  and if possible, the prediction is also based on predicted or estimated future disturbances. These disturbances are not considered in this thesis.

At sample  $k$  the future input  $\bar{u}(k)$  is determined such that a certain objective function  $J$  is minimized subject to certain constraints. Consecutive in- and outputs of max-plus models of manufacturing systems are time instants that should be non-decreasing. Therefore, is, compared to MPC of linear discrete-time systems, instead of the input  $u$ , the input rate  $\Delta u$  taken to be constant between the end of the control horizon and the end of the prediction horizon:  $\Delta u(k+j) = \Delta u(k+N_c)$  for  $j = N_c, \dots, N_p$ . Here  $\Delta u$  is defined as:  $\Delta u(k) = u(k) - u(k-1)$ .

## 5.2 The standard MPC problem

In the previous section, the concept of MPC has been considered. Main MPC items such as the objective function  $J$  and constraint handling have been briefly introduced. In this section, the objective function and the constraint handling to be used in this project are treated. Combining the objective function, the constraints and the dynamics of the manufacturing system as described in Chapter 4, a standard MPC control problem is composed that is used in this project.

## Objective function

As mentioned in the previous section, the input vector determined by MPC, is optimal with respect to the chosen objective function or cost criterion. This objective function reflects the reference tracking error or output cost criterion ( $J_{\text{out}}$ ) and the control effort or input cost criterion ( $J_{\text{in}}$ ). Several objective functions can be chosen. In this report, the difference between the due dates (reference signal) and the actual output time instants is minimized. This objective function should, on the one hand, prevent finished products to leave the system too early (which would result in stock), on the other hand this function should prevent finished products to leave the system too late (to prevent back log). This stock and tardiness result in certain penalties. The tracking error ( $J_{\text{out}}$ ) becomes:

$$J_{\text{out}} = \sum_{i=1}^l \sum_{j=1}^{N_p} |\hat{y}_i(k+j|k) - r_i(k+j)|. \quad (5.1)$$

In (5.1),  $l$  denotes the number of in- and outputs. For the manufacturing system of the theoretical case, see Chapter 4, that is modelled using the max-plus-algebra, the number of inputs is equal to the number of outputs.

The tracking error (5.1) can be written using the maximization operator:

$$J_{\text{out}} = \sum_{i=1}^l \sum_{j=1}^{N_p} \max(\hat{y}_i(k+j|k) - r_i(k+j), r_i(k+j) - \hat{y}_i(k+j|k)). \quad (5.2)$$

If, according to the reference signal  $\bar{r}(k)$ , finished products have to leave the manufacturing system in a certain (too) high production rate this results in an instable situation. Stability in conventional system theory is concerned with boundedness of the states. In max-plus linear systems the sequence of consecutive states is always nondecreasing. This means that for  $k \rightarrow \infty$ , the time instants  $x_i(k)$  will be unbounded. Therefore, De Schutter and Van den Boom define stability for DES as follows: a discrete event system is called stable if all its buffer levels remain bounded [Sch00b]. To prevent this instable situation, another part of the objective function,  $J_{\text{out}}$  has to result in a maximization of the control input. This maximization prevents the overflow of the input buffer. The necessary raw material is fed to the system as late as possible and the internal buffer levels are kept as low as possible. The input cost criterion  $J_{\text{in}}$  then becomes:

$$J_{\text{in}} = - \sum_{i=1}^l \sum_{j=1}^{N_p} u_i(k+j). \quad (5.3)$$

The entire objective function becomes the sum of the output (5.2) and input (5.3) criterion with a certain (weighting) parameter  $\lambda$ . This parameter  $\lambda$  makes a trade-off between minimization of the tracking error and the needed control effort.

$$J = J_{\text{out}} + \lambda J_{\text{in}}.$$

### Constraint handling

As mentioned before, a great benefit of MPC is the possibility to use constraints on the input(s), output(s) and/or state(s). For manufacturing systems, typical constraints are a lower bound (lb) or an upper bound (ub) on the input or output rates:

$$\text{lb}_1 \leq \Delta u(k+j) \leq \text{ub}_1 \quad \text{for } j = 1, \dots, N_p, \quad (5.4)$$

$$\text{lb}_2 \leq \Delta y(k+j|k) \leq \text{ub}_2 \quad \text{for } j = 1, \dots, N_p, \quad (5.5)$$

where,  $\Delta u(k) = u(k) - u(k-1)$ .

As mentioned before, consecutive inputs and outputs of max-plus models of manufacturing systems are time instants that should be non-decreasing. Therefore, the lower bounds of (5.4) and (5.5) should always be greater than zero.

### Output prediction

Using the standard max-algebraic state-space description (3.1a) and (3.1b), the future values of the output can be computed. An important assumption is that the states at event step  $k$  can be measured or estimated using (3.1). In order to keep the number of variables in the resulting optimization problem as low as possible, the predictions of the output values are not determined by (3.1b). A different, more efficient way of determining the predicted output, which leads to faster computation of the optimal input and is explained as follows. Recall that (3.1) gives the standard state-space description:

$$x(k+1) = A \otimes x(k) \oplus B \otimes u(k) \quad (5.6a)$$

$$y(k) = C \otimes x(k) \oplus D \otimes u(k). \quad (5.6b)$$

for  $k = 0$ , (5.6a) and (5.6b) become:

$$x(1) = A \otimes x(0) \oplus B \otimes u(0) \quad (5.7a)$$

$$y(0) = C \otimes x(0) \oplus D \otimes u(0). \quad (5.7b)$$

for the next sample ( $k = 1$ ), (5.6a) and (5.6b) become:

$$x(2) = A \otimes x(1) \oplus B \otimes u(1) \quad (5.8a)$$

$$y(1) = C \otimes x(1) \oplus D \otimes u(1). \quad (5.8b)$$

Substitution of (5.7a) in (5.8b) gives:

$$y(1) = C \otimes A \otimes x(0) \oplus C \otimes B \otimes u(0) \oplus D \otimes u(1).$$

Note that now, only the initial state vector (here  $x(0) = x_0$ ) is needed to determine future output values. The number of optimization variables is reduced. Using this way of predicting the output values, the following matrix notation can be used:

$$\tilde{y}(k) = H \otimes \bar{u}(k) \oplus g \otimes x_0 \quad (5.9)$$

with:

$$H = \begin{pmatrix} D & \varepsilon & \cdots & \cdots & \varepsilon \\ C \otimes B & D & \ddots & & \varepsilon \\ C \otimes A \otimes B & C \otimes B & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \varepsilon \\ C \otimes A^{\otimes N_p-1} \otimes B & C \otimes A^{\otimes N_p-2} \otimes B & \cdots & C \otimes B & D \end{pmatrix} \quad (5.10)$$

$$g = \begin{pmatrix} C \\ C \otimes A \\ C \otimes A^{\otimes 2} \\ \vdots \\ C \otimes A^{\otimes N_p} \end{pmatrix}. \quad (5.11)$$

### Standard MPC problem

So far, the objective function, the constraint handling, and the output prediction have been discussed. By combining these items, an optimization problem can be composed. This problem is called the standard MPC problem, as presented in (5.12). In this problem, only the input rates are limited by a lower- and upper bound. The original problem can be denoted as follows:

$$\min_{\bar{u}(k)} J = \min_{\bar{u}(k)} J_{\text{out}} + \lambda J_{\text{in}} \quad (5.12a)$$

with:

$$J_{\text{out}} = \sum_{i=1}^l \sum_{j=1}^{N_p} \max(y_i(k+j|k) - r_i(k+j), r_i(k+j) - y_i(k+j|k))$$

$$J_{\text{in}} = \sum_{i=1}^l \sum_{j=1}^{N_p} u_i(k+j)$$

subject to:

$$\tilde{y}(k) = H \otimes \bar{u}(k) \oplus g \otimes x_0 \quad (5.12b)$$

$$\text{lb} \leq \Delta u(k+j) \leq \text{ub} \quad \text{for } j = 1, \dots, N_p \quad (5.12c)$$

$$\Delta^2 u(k+j) = 0 \quad \text{for } j = N_c, \dots, N_p. \quad (5.12d)$$

$$\begin{aligned} \text{with: } \Delta^2 u(k) &= \Delta u(k) - \Delta u(k-1) \\ &= u(k) - 2u(k-1) + u(k-2). \end{aligned}$$

Solving this problem, the input rates in  $\bar{u}(k)$  are the only optimization variables. The output predictions  $\tilde{y}(k)$  are determined using (5.12b). In general this standard MPC problem (5.12) is a nonlinear nonconvex optimization problem due to (5.12b). The equality constraint (5.12d) and inequality constraint (5.12c) are convex in  $u$ . Equality constraint (5.12b) is in general not convex due to the max-plus operators  $\oplus$  and  $\otimes$ .

Many methods exist for solving the standard optimization problem (5.12). In this research project three optimization methods are implemented in Matlab. The first method is a constrained nonlinear optimization solver from Matlab, called **fmincon**. This solver finds the constrained minimum of a function of several variables using Sequential Quadratic Programming (SQP) and can deal with linear and nonlinear constraints and objective functions [Pap00]. **Fmincon** does not need information about the

properties of the optimization problem (such as gradients and information on convexity and linearity). The standard MPC problem is a nonlinear nonconvex optimization problem due to (5.12b). De Schutter and Van den Boom describe in [Sch00a] a method to reformulate this problem as a convex optimization problem. This so called relaxed problem can be solved very efficiently by replacing the  $=$  – sign in (5.12b) with  $\geq$ . This relaxed problem is called the relaxed MPC problem. Compared to the standard MPC problem (5.12), now both  $u(k)$  and  $y(k)$  are optimization variables. Due to the replacement of the  $=$  – sign to the  $\geq$  – sign the feasible area becomes larger. In [Sch00a] is proven that if the objective function  $J$  is a monotonically nondecreasing function of  $\tilde{y}$  and  $(\tilde{u}_{\text{nlcon}}^*, \tilde{y}_{\text{nlcon}}^*)$  is an optimal solution of the relaxed MPC problem, then  $(\tilde{u}_{\text{nlcon}}^*, \tilde{y}_{\text{nlcon}}^\#)$  is an optimal solution of the standard original MPC problem. Here,  $\tilde{y}_{\text{nlcon}}^\# = H \otimes \tilde{u}_{\text{nlcon}}^* \oplus g \otimes x_0$ . In this research project, the objective function  $J$  is not monotonically nondecreasing. Therefore, no proof exists that if  $(\tilde{u}^*, \tilde{y}^*)$  is an optimal solution of the relaxed problem,  $(\tilde{u}^*, \tilde{y}^\#)$  is an optimal solution of the original problem. Due to the lack of this proof, the difference between the output of the relaxed problem,  $\tilde{y}^*$ , and the original problem,  $\tilde{y}^\#$  has to be zero to know if the solution is optimal. The feasible solution of the original problem is a subset of the set of feasible solutions of the relaxed problem. Unfortunately the constrained nonlinear optimization method using `fmincon` does not converge to an optimal input sequence, but keeps oscillating. Detailed information, of the Matlab implementation of this optimization method, can be seen on the CD-rom that goes with this report.

Therefore, (5.12) is implemented as a linear constraint optimization problem where a penalty function is used for the nonlinear constraints. The nonlinear constraint (5.12b) is no longer described as a constraint. Problem (5.12) is relaxed by adding the difference of the output of the relaxed problem and the output of the original problem, multiplied by a weighting parameter  $\mu$ . Using a high value for  $\mu$ , compared to  $\lambda$ , results in an optimal solution whereas  $\sum_{i=1}^l \sum_{j=1}^{N_p} (y_{i, \text{pen}}^*(k+j) - y_{i, \text{pen}}^\#(k+j))$  is dominant with respect to the original terms in the original objective function. Unfortunately, this optimization method with a penalty function does not give satisfying results due to the significant differences between the output of the relaxed problem and the standard MPC problem. More information about this linear constraint optimization with a penalty function can be seen on the before mentioned CD-rom.

Due to the difficulties and the poor results of both the optimization methods mentioned above, the search for an efficient and suitable optimization method leads to Linear Programming (LP). The standard MPC problem has to be transformed into the following standard LP problem:

$$\min_{\bar{u}(k), \bar{y}(k), \bar{z}(k)} J = \min_{\bar{u}(k), \bar{y}(k), \bar{z}(k)} c^T \bar{x} \quad (5.13a)$$

subject to

$$A\bar{x} \leq b \quad (5.13b)$$

$$A_{\text{eq}}\bar{x} \leq b_{\text{eq}} \quad (5.13c)$$

$$\text{lb} \leq \bar{x} \leq \text{ub}. \quad (5.13d)$$

Variables  $\bar{u}(k)$ ,  $\bar{y}(k)$  and  $\bar{z}(k)$  in (5.13a) are elements of the vector  $x$  that contains all decision variables, as presented in Appendix F. Transforming the standard MPC problem (5.12) into an LP problem means that the objective function and all constraints have to be written in a linear formulation. Therefore, the objective function will be of the form:

$$J = c_1x_1 + c_2x_2 + \dots + c_nx_n. \quad (5.14)$$

In addition to the objective function, also the equality and/or equality constraints have to be written in a linear combinations of the decision variables:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \left\{ \begin{array}{l} \leq \\ = \end{array} \right\} b. \quad (5.15)$$

In Appendix F, the transformation of (5.12) into an LP problem of the form (5.13) is worked out. This transformation results in an adapted objective function due to the addition of dummy variables  $z$ :

$$J = \lambda \sum_{i=1}^2 \sum_{j=1}^{N_p} u_i(k+j) + \sum_{i=1}^2 \sum_{j=1}^{N_p} y_i(k+j) + \sum_{i=1}^2 \sum_{j=1}^{N_p} z_i(k+j). \quad (5.16)$$

In Appendix F the transformation is explained using an example. In this example, the length of the prediction horizon equals 2 and the length of the control horizon equals 1. Transformation of (5.16) into an objective function of the form (5.14) results in:

$$J = c^T \bar{x} \quad (5.17)$$

with (in case of  $N_p = 2$  and  $N_c = 1$ ):

$$c^T = ( \lambda \quad \lambda \quad \lambda \quad \lambda \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 )$$

$$\bar{x} = \begin{pmatrix} u_1(1) & u_2(1) & u_1(2) & u_2(2) & y_1(1) & y_2(1) & y_1(2) & y_2(2) \\ \dots & z_1(1) & z_2(1) & z_1(2) & z_2(2) \end{pmatrix}^T.$$

The (in)equality constraints remain, in contrast to the objective function, similar to the original problem (5.12). Constraints (5.12c) and (5.12d) are reformulated as (5.13c) and (5.13b). Before the manufacturing system that is worked out in Chapter 4 is controlled using MPC and (5.13), certain parameters have to be tuned to gain satisfactory results. This parameter tuning is discussed in Section 5.3.

### 5.3 Tuning the MPC parameters

A proper MPC controller contains certain parameters that need to be tuned to gain satisfactory results. In this section, the parameters that have most influence on the performance of the controller are presented and their way of tuning is discussed. Here, only the necessary aspects or guidelines with relation to parameter tuning that leads to a good working controller are given. The following parameters are considered:

- prediction horizon  $N_p$ ,
- control horizon  $N_c$ ,
- weighting parameter  $\lambda$ .

In this report, time-invariant state-space models are used to describe manufacturing systems. As mentioned in Chapter 1, time-invariant systems respond to a certain input sequence, which means that these systems are not dependent on absolute time. Therefore, the term sample time can not be used here. Sample time is a parameter of MPC that has to be tuned only if models are used that are dependent on absolute time.

#### Prediction horizon $N_p$

The first parameter that is tuned is the prediction horizon  $N_p$ . The event interval  $(1, N_p)$  has to contain the crucial dynamics of the process. Or, in other words, the prediction horizon has to be taken at least long enough for the complete effect of an input can be seen within  $N_p$  samples. Determination of  $N_p$  can be done using the max-plus-algebraic impulse response, or short impuls response [Sch00b, Sch96]. The impuls response can



be defined as the output sequence that results from a max-plus-algebraic unit impulse applied to a max-plus system. This unit impulse  $e(k)$  can be interpreted as follows:

$$\begin{cases} e(k) = 0 & \text{if } k = 0 \\ e(k) = \varepsilon & \text{if } k \neq 0. \end{cases}$$

Feeding a max-algebraic unit impuls to the  $i$ th input of the system with  $x(0) = \varepsilon_{n \times 1}$ , results in  $y(k) = C \otimes A^{\otimes k-1} \otimes B_i$  for  $k = 1, 2, \dots$  as the output of the system, where  $B_i$  is the  $i$ th column of  $B$ . Using this unit impuls for all the inputs  $i = 1, 2, \dots, m$  of the system, this can be written using matrix  $G_{k-1} = C \otimes A^{\otimes k-1} \otimes B$  for  $k = 1, 2, \dots$ . The  $\{G_k\}_{k=0}^{\infty}$  is called the impuls response of the system. Note, that there is an analogy between the impuls response and the  $H$  matrix (5.10). The  $G_k$ 's are called response matrices or Markov parameters [Sch96]. Considering a manufacturing system, the following physical interpretation of the impulse response can be given. At event counter  $k = 0$ , all the internal buffers of the system are empty. Then, raw material is fed to the system and this is done at such a rate that the internal buffers never become empty. The time instants at which the finished products leave the system correspond to the terms of the impulse response [Sch00b]. Using this theory, a lower bound for  $N_p$  can be determined. Let  $\{G_k\}_{k=0}^{\infty}$  be the impuls response of a max-plus-linear manufacturing system, then there exist constants  $c$ ,  $k_0$  and  $\rho$ , such that:

$$G(k) = c\rho + G(k - c) \quad \forall k \geq k_0. \quad (5.18)$$

A specific impuls response, such as (5.18), is called ultimately periodic with cycle period  $c$ . Variable  $\rho$  gives the average duration of a cycle. The length of the impuls response is now defined as the minimal value of  $k_0$  for which (5.18) holds [Sch00b]. The average production rate of the manufacturing system, then becomes  $1/\rho$  because every  $\rho$  time units (a) finished part(s) leave(s) the manufacturing system. Next, an upper bound has to be found. The upper bound of the prediction horizon length is determined by the available computation time, since a larger horizon requires more computation time. By using this approach to determine the lower bound of  $N_p$ , the manufacturing system is assumed to behave according to (5.18) and that no noise or peaks appear using a unit impulse.

### Control horizon $N_c$

The second parameter that has to be tuned is the control horizon  $N_c$ . If the reference trajectory that has to be followed by the manufacturing system varies relatively fast in time, a longer control horizon is more suitable. A large interval of  $(1, N_c)$  results therefore, in more 'aggressive' control. On the other hand, and similar to the prediction horizon, a larger control horizon leads to larger computation time. A smaller control horizon leads to less computational effort, but results in a slower system response. Since the reference trajectory varies slowly,  $N_p$  can be chosen small. A rule of thumb is that the length of the control horizon is chosen to be between 1/6 and 1/3 of the prediction horizon [Ess02].

### Weighting parameter $\lambda$

The last parameter to tune is the weighting parameter  $\lambda$ . Parameter  $\lambda$  makes, in the original objective function (5.12a), a trade-off, between minimization of the tracking error and the needed control effort. In case of  $\lambda = 0$ , the control effort is no part of the objective function to be minimized. This means that the input time instants of the products are not longer maximized. Using this setting, no unique solution of this optimization problem exists, any input  $u(k)$  that results in the minimization of the output and the reference signal can be a solution. One of these solutions is  $u(k) = u(k-1) \forall k$  which causes stability problems due to the overflow of the input buffer. In this thesis, the input rate is bounded between a lower- and a upper bound. Therefore, for  $\lambda = 0$ ,  $u(k) - u(k-1)$  is equal to the lower bound. A situation where  $\lambda < 0$ , results in minimization of the control input and is, as mentioned earlier, in practice not suitable. Therefore, the parameter  $\lambda$ , has to be chosen larger than zero. For values of  $\lambda$  larger than a certain value  $\lambda_0$ , the cost criterion  $J_{in}$  in the objective function (5.12a) can be dominant compared to the tracking error  $J_{out}$ . This setting results in the maximization of the control input. As mentioned before (5.12c) prevents a strongly varying input rate. Therefore, in cases of  $\lambda > \lambda_0$ ,  $u(k) - u(k-1)$  becomes equal to the upper bound. Preventing that, if the reference signal does not require a minimum or maximum input rate,  $u(k) - u(k-1)$  becomes equal to the lower- or the upper bound, parameter  $\lambda$  should be in the following interval :

$$0 < \lambda < \lambda_0. \quad (5.19)$$

and is usually chosen as small as possible [Sch00b].

## 5.4 MPC implementation and simulation results

Now, the standard MPC problem with its objective function, constraint handling, output prediction and its parameter tuning has been discussed, the theoretical case of Chapter 4 has to be controlled. The objective here is controlling the output sequence of the manufacturing system with respect to a given desired output sequence of finished products. In this section, the implementation of the MPC approach and the theoretical case and its results are given.

### Explanation of MPC implementation

A graphical representation and the Matlab file of this implementation can be seen in respectively Figure 5.2 and Appendix G.

In Figure 5.2 four gray blocks can be distinguished. The first block increases counter  $k$  by 1. This block receives as input the 'old' value of  $k$ , and the 'old' state vector  $x_0$ . The second gray block, represents the optimization method. In this thesis this

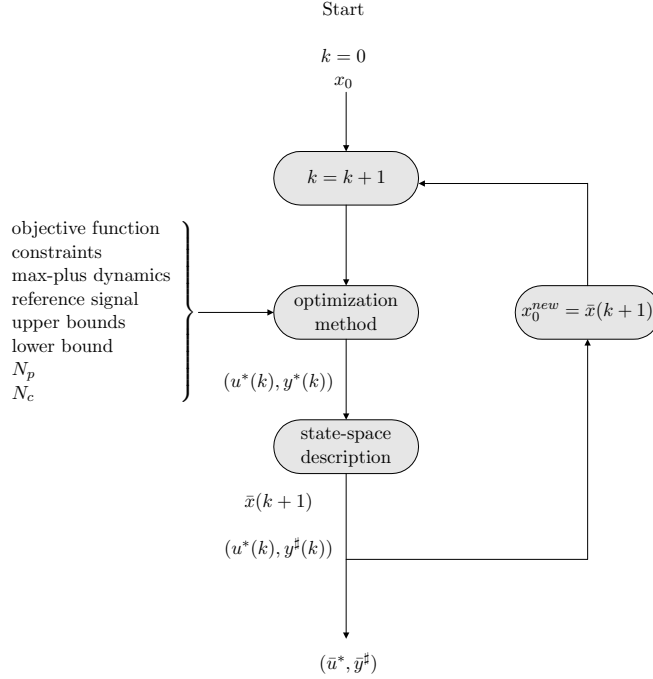


Figure 5.2: MPC implementation

optimization method is LP, which needs a certain input to determine an optimum. The parameters such as the objective function and the constraints have been discussed already in Section 5.2. Other important inputs of the optimization method are the reference signal  $\bar{r}(k)$  and the max-plus-dynamics of the manufacturing system using the standard state-space description (3.1). The second part of the state-space description (3.1b) can be written as (5.9) to keep the number of variables in the optimization problem as low as possible. Using (5.9) to obtain the output predictions, an initial state vector  $x_0$  is needed. This state vector is updated after each sample. The upper and lower bounds of the input rate are chosen to be equal to 5, respectively 12 time units. The upper bound is chosen to be smaller than the minimum cycle time of the manufacturing system, which is 13 time units due to the process time of machine 2 and the batch machine. The lower bound is chosen to be not too small to prevent buffers overflow. Last, the length of both  $N_c$  and  $N_p$  are needed. These are parameters to tune and are determined later in this section. Due to the LP transformation (see Appendix G), the original problem (5.12) is relaxed. This means that the optimal input and output sequence  $(u_{LP}^*(k), y_{LP}^*(k))$  does not accompany the original problem, but corresponds to the relaxed problem (5.13). As mentioned in Section 5.2, if  $(\tilde{u}_{LP}^*, \tilde{y}_{LP}^*)$  is an optimal solution of the relaxed MPC problem, then  $(\tilde{u}_{LP}^*, \tilde{y}_{LP}^\#)$  is an optimal solution of the standard original MPC problem, only if  $\tilde{y}_{LP}^* = \tilde{y}_{LP}^\#$ . Here,  $\tilde{y}_{LP}^\# = H \otimes \tilde{u}_{LP}^* \oplus g \otimes x_0$ . In Figure 5.2 can be seen that the found optimal input sequence  $\bar{u}_{LP}^*(k)$  is substituted in the state-space description of the manufacturing system. Using this approach, the

state vector  $\bar{x}(k+1)$  and the optimal output  $\hat{y}_{LP}^\#$  can be determined. Due to the use of a theoretical case, no error exists between the model and the physical manufacturing system. At each step, the state is assumed to be measurable or reconstructible from previous measurements. Since a state  $x(k)$  correspond to event times, they are in general easy to measure [Sch00b]. Now the optimal in- and output sequences zijn known, only the first sequences are implemented. The initial state vector  $x_0$  is updated and becomes equal to  $\bar{x}(k+1)$ . Then, the entire procedure is repeated until the number of maximum steps,  $k_{\max}$ , is reached, see Appendix G.

### Application of tuning guidelines

In Section 5.3, the tuning guidelines of MPC parameter  $N_p$ ,  $N_c$ , and  $\lambda$  have been discussed to obtain a proper MPC controller for the manufacturing system described in Chapter 4. First, the length of the prediction horizon,  $N_p$  is tuned. In Section 5.3, an  $N_p$  tuning approach has been discussed. The max-plus dynamics of the manufacturing system can be seen in Section 3.6 and Appendix C. To find the minimum length of  $N_p$ , the impuls response of the manufacturing system has to be determined. The length of this impuls response is defined as the minimum value of  $k_0$  for which (5.18) holds [Sch00b]. Experiments show that the average duration of a cycle equals 10 time units. According to the output sequence the value of  $k_0$  equals 0. In practice, this means that the manufacturing system immediately reaches a certain constant state. This constant state can not be compared to a steady state situation, due to the overflow of the internal buffers. A value of  $k_0 = 0$  results in a length of the prediction horizon  $N_p$  that equals one. A prediction horizon equal to one sample should give satisfactorily results, within its constraints with respect to a strongly changing reference signal. This remarkable result can be explained by analyzing the structure of the manufacturing system. The bottleneck of the system is without doubt the batch machine due to its long process time ( $d_3 = 10$ ). The utilization level of a machine is in [Sch96] defined as:

$$u_i = \frac{d_i}{\lambda} \quad (5.20)$$

where:  $i$  is the machine number.

Using (5.20), the utilization of the batch machine is computed as 100%. A utilization level of 100% refers to an instable situation if process time is stochastic. The 100% utilization level does not necessarily correspond to an instable situation if the process times are deterministic. The use of the max-algebraic unit impulse response on a manufacturing system results in an instable situation due to the overflow of the buffers. Therefore, the utilization level of 100%, determined using (5.20) confirms the instable situation. The behavior of this machine dominates the dynamics of the entire system. Another possible reason for the minimum length of the prediction horizon that equals one are the two parallel machines. The difference between the process times and the fact that the batch machine can only start if both machines finished their products, make the machine with the longest process time dominant compared to the other machine. This

dominance and the deterministic behavior of the entire manufacturing system might explain the short length of the prediction horizon. If, in practice,  $N_p$  is required to be equal to 1, this means that the manufacturing system does not need much future information to deliver products on time with respect to its constraints, which is a huge economical advantage. The length of the control horizon is equal to or smaller than the length of the prediction horizon. In case of  $N_p = 1$ , the length of  $N_c$  becomes 1 as well.

Now the length of both the prediction horizon and the control horizon are known, weighting parameter  $\lambda$  can be tuned, where:  $0 < \lambda < \lambda_0$ . To determine the value of  $\lambda_0$ , two reference signals are fed to the system. The first reference,  $r_1(k)$ , see Table 5.2, should be achievable by the manufacturing system. In contrast with  $r_1(k)$ , the second reference signal,  $r_2(k)$ , see Table 5.3, is not achievable by the manufacturing system. Weighting parameter  $\lambda$  varies from 0 to 10. Two criteria are used to find a good value for  $\lambda$ . The first criterium is the absolute value of the difference between the reference signal and the actual output sequence of the original MPC problem:

$$\sum_{i=1}^2 \sum_{j=1}^{N_p} |(\hat{y}_i^\#(k+j|k) - r_i(k+j))|. \quad (5.21)$$

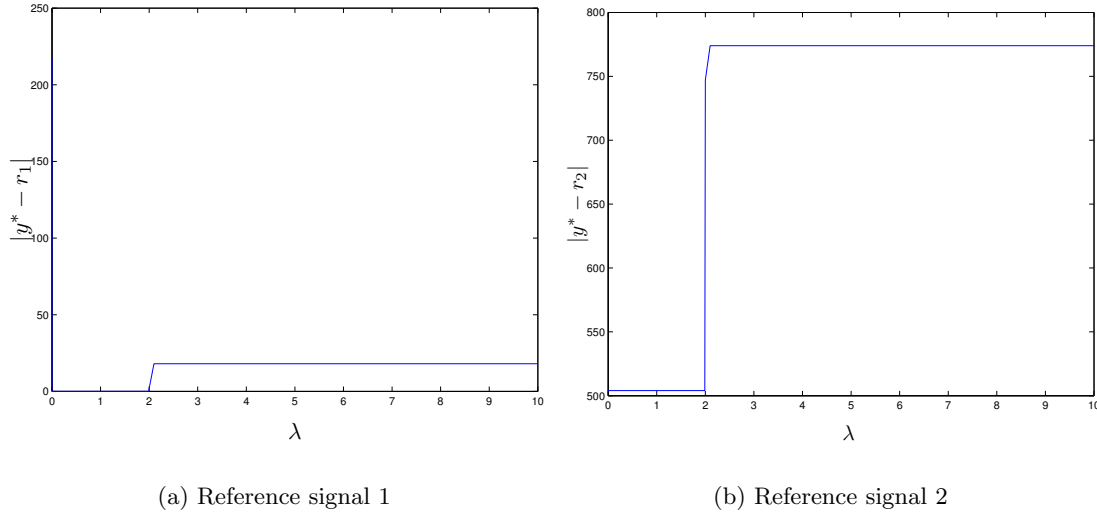
The second criterium is the absolute value of the difference between the optimal output sequence of the relaxed and the original optimization problem:

$$\sum_{i=1}^2 \sum_{j=1}^{N_p} |(\hat{y}_i^*(k+j|k) - \hat{y}_i^\#(k+j|k))|. \quad (5.22)$$

The optimal solution of the relaxed MPC problem is only an optimal solution of the original problem if  $\hat{y}^* = \hat{y}^\#$ . This means that both criteria should be, in best case, equal to zero.

The results of the two reference signals to tune  $\lambda$  can be seen in Figure 5.3. In Figures 5.3(a) and 5.3(b) the results of the value of the first criterium, (5.21), with respect to respectively  $r_1(k)$  and  $r_2(k)$  can be seen. The second criterium has to be equal to zero in all cases. Only then, the optimal solution of the relaxed MPC problem is an optimal solution of the original MPC problem. In case of almost all values of  $\lambda$  and both reference signals, this criterium equals zero. However, if  $\lambda = 0$  and the use of the achievable reference signal  $r_1(k)$ , criterium 2 does not equal zero. In the objective function (5.12a),  $\lambda = 0$  implies that the maximization of the control input is ignored and that only the reference signal has to be followed. Since the reference signal  $r_1(k)$  is achievable by the manufacturing system, this result is remarkable. Therefore, a recommendation for future research is to compose another LP formulation to prevent this error. This difference between  $\hat{y}^*$  and  $\hat{y}^\#$  for  $\lambda = 0$  and the results presented in Figures 5.3(a) and 5.3(b) imply that the value of  $\lambda$  should be between 0 and 2, thus ( $\lambda_0 = 2$ ). A value of  $\lambda > 2$  results in the dominance of the maximization of the objective function. Therefore,  $\lambda$  should be in the following interval:

$$0 < \lambda < 2. \quad (5.23)$$

Figure 5.3: Graphical representation of results due to tuning of  $\lambda$ 

and is usually chosen as small as possible [Sch00b]. In this research is chosen to set  $\lambda$  at 0.01.

Now, all three parameters have been tuned. The fixed values of  $N_p$ ,  $N_c$  and  $\lambda$  that are used in the rest of this report can be seen in Table 5.1.

MPC parameters	fixed value
$N_p$	1
$N_c$	1
$\lambda$	0.01

Table 5.1: Fixed values of  $N_p$ ,  $N_c$ , and  $\lambda$ 

## Control simulations

Now, the tuning parameters are set at fixed values (see Table 5.1), the MPC controller should give satisfactorily results. To test the MPC controller, several different reference signals are fed to the MPC controller. The performance of the controller is determined with respect to the criteria as mentioned earlier in this section, (5.21) and (5.22). The first criterium, (5.21) can be greater than zero, if the reference signal is not achieved. The second criterium, (5.22), should always be equal to zero, due to the dynamics of the manufacturing system. Hereafter, the expectations of the simulations are given. Here, only expectations are done with respect to (5.21), because (5.22) has to be equal to zero in all cases. The process times of machine 1, 2 and the batch machine are, as mentioned

in Chapter 4, respectively 1, 3 and 10 time units. The upper- and lower bounds of the input rates are chosen to be respectively 12 and 5 time units.

The five different reference signals can be seen in Tables 5.2 to 5.6. The first reference (see Table 5.2),  $\bar{r}_1(k)$ , that is fed to the MPC controller should be achievable by the manufacturing system, due to the minimum cycle time of a product. This minimum cycle time through the manufacturing system is 13 time units due to the process time of machine 2 which is 3 time units and the process time of the batch machine of 10 time units. Expected is that the input rate becomes equal to the upper bound, which is 12 time units. One exception is the time instant at which the first two products are fed to the system. These products have to be fed to the system at time instant 11 to leave the system at time instant 24, see Table 5.2. Criterium (5.21) is expected to be equal to zero. Simulation shows that the above mentioned expectations are correct. In

system feed nr.	1	2	3	4	5	6	7	8	9
$\bar{r}_1(k)$ of $P_1$	24	36	48	60	72	84	96	108	120
$\bar{r}_1(k)$ of $P_2$	24	36	48	60	72	84	96	108	120

Table 5.2: Reference signal 1

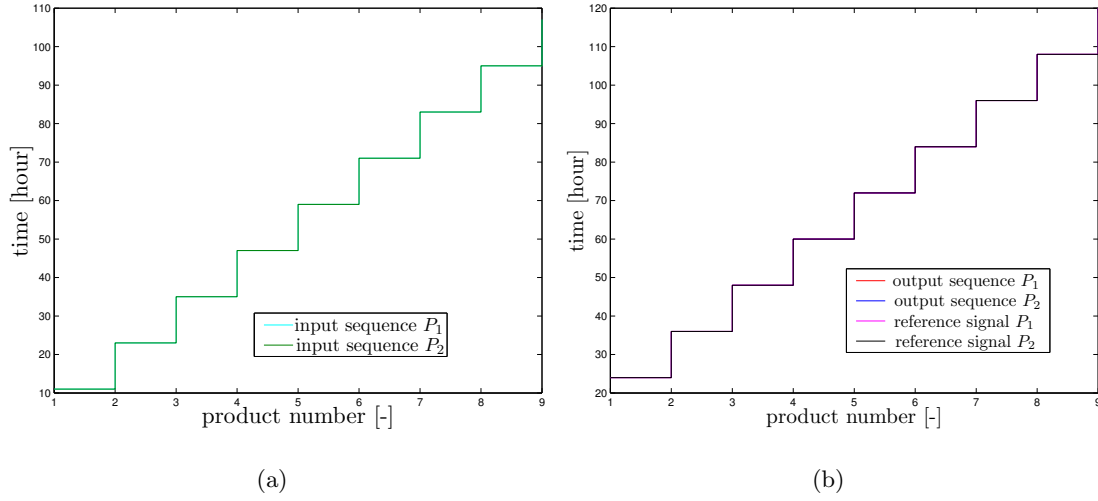
Figures 5.4(a) and 5.4(b) the results of the simulations can be seen. In Figure 5.4(a) the input sequence can be seen. Note that since the input sequence of both product  $P_1$  and  $P_2$  only one line is plotted. In Figure 5.4(b) the reference signal and the actual output sequence can be seen. The output sequence of both products  $P_1$  and  $P_2$  is always equal due to the batch machine policy and the free output that can always receive products. In Figure 5.4(b) one can see that the reference signal is equal to the actual output sequence of the products. This results in a value of criterium 1, (5.21) that equals zero.

The second reference signal,  $\bar{r}_2(k)$  is the opposite of the first reference, see Table 5.3. This reference signal is not achievable for the manufacturing system due to its minimum flow time. Feeding all products at an input rate that equals the lower average duration of a cycle of the manufacturing system (see Section 5.4). Due to the maximization of the input in the objective function, one expects the input to equal 10. Criterium 1, (5.21), does not equal zero due to the reference signal which is not achievable.

system feed nr.	1	2	3	4	5	6	7	8	9
$\bar{r}_2(k)$ of $P_1$	10	15	20	25	30	35	40	45	50
$\bar{r}_2(k)$ of $P_2$	10	15	20	25	30	35	40	45	50

Table 5.3: Reference signal 2

Figures 5.5(a) and 5.5(b) show a similar behavior of the manufacturing system as expected. As can be seen in Figure 5.5(a), the first products are fed to the system with an input rate of 5. All next products are fed to the system with an input rate of 10. Figure 5.5(b) represents the difference between the actual output sequence of the products and the reference signal.

Figure 5.4: Graphical representation of results due to reference signal  $r_1$ 

The third reference signal, see Table 5.4, is almost similar to  $\bar{r}_1(k)$ , with the difference that in system feed number 4 the desired output sequence is not achievable for the system. The input rate is expected to equal its upper bound, except for the first products (due to the reference signal, see  $\bar{r}_1$ ) and system feed number 4. Here, the difference between the third and the fourth feed becomes equal to 10, due to its average duration of a cycle and the input cost criterion of the objective function. The difference between the desired output sequence and the actual output (5.21), does not equal zero, due to system feed number 4.

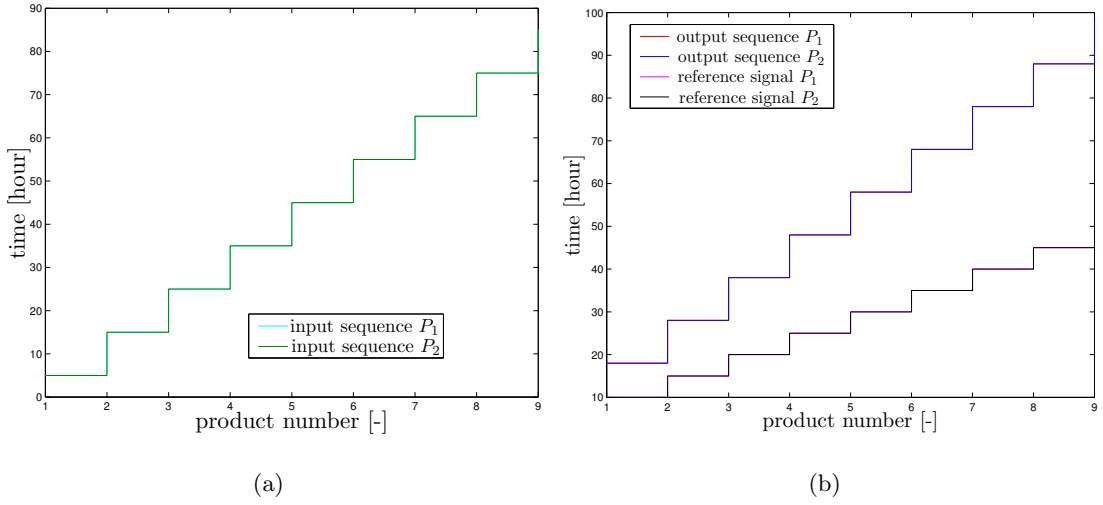
system feed nr.	1	2	3	4	5	6	7	8	9
$\bar{r}_3(k)$ of $P_1$	24	36	48	12	72	84	96	108	120
$\bar{r}_3(k)$ of $P_2$	24	36	48	12	72	84	96	108	120

Table 5.4: Reference signal 3

The third reference signal that is fed to the system results in the expected in- and output sequence. Due to the low reference signal in feed 4, the manufacturing system is, due to its constraints on the input rate, not capable of following the desired reference. The input sequence can be seen in Figure 5.6(a), the reference signal and the output sequence can be seen in Figure 5.6(b).

The fourth reference signal, see Table 5.5, is the opposite of  $\bar{r}_3(k)$ . Here, a signal is fed to the system that is not achievable, with exception of feed number 4, by the manufacturing system. In this feed, the desired output is achievable. The input rate of the first feed is expected to become 5 and that it equals 10 for the rest of the system feeds, with



Figure 5.5: Graphical representation of results due to reference signal  $r_2$ 

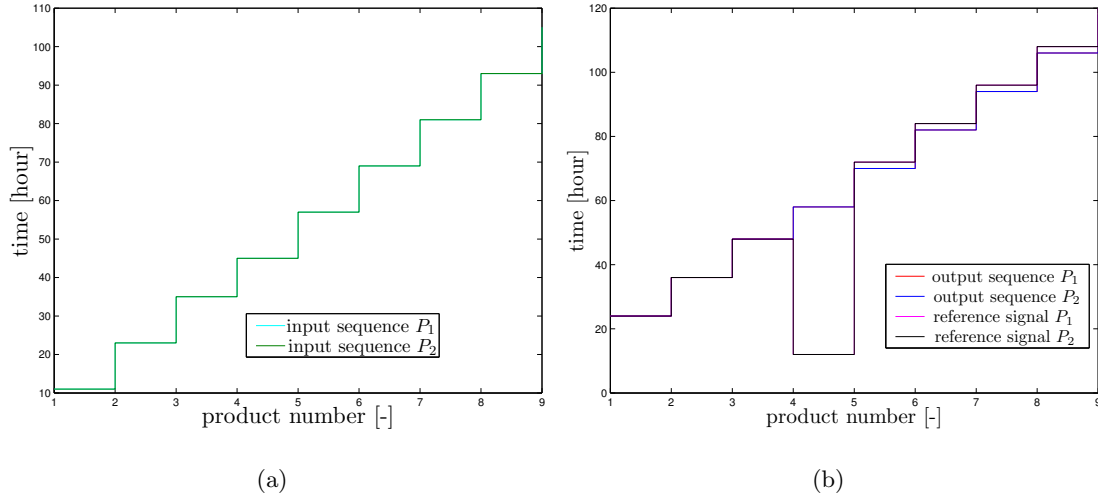
system feed number 4 as exception. Here, the difference between the input time instant of system feed number 3 is equal to the upper bound, which is 12. Criterion (5.21) does not equal zero due to the reference signal which is not achievable.

system feed nr.	1	2	3	4	5	6	7	8	9
$\bar{r}_4(k)$ of $P_1$	10	15	20	100	30	35	40	45	50
$\bar{r}_4(k)$ of $P_2$	10	15	20	100	30	35	40	45	50

Table 5.5: Reference signal 4

Figures 5.7(a) and 5.7(b) show that the expectations about the simulations are correct. The first product is fed to the system at time instant 5. Due to the minimum cycle time of 10, feeding the manufacturing system with an input that equals the lower bound is not useful. As mentioned before, due to the maximization input sequence in the objective function, an input rate of 10 is optimal. In feed number 4, the desired reference signal requires an input rate that equals the upper bound. In all remaining feeds, the input rate again becomes 10.

The last desired signal,  $\bar{r}_5(k)$  (see Table 5.6) that is fed to the MPC controller demands the manufacturing system to finish product  $P_2$  earlier than product  $P_1$ . This is, with respect to the dynamics of the system, not possible due to the batch machine policy. The MPC controller is expected to feed both products only with respect to the product that has to leave the system earliest, in this case  $P_2$ , in order to keep the difference between  $P_2$  and its reference as low as possible. The difference between the reference signal of  $P_1$  and the actual output of  $P_1$  remain the same. Therefore, the reference of

Figure 5.6: Graphical representation of results due to reference signal  $r_3$ 

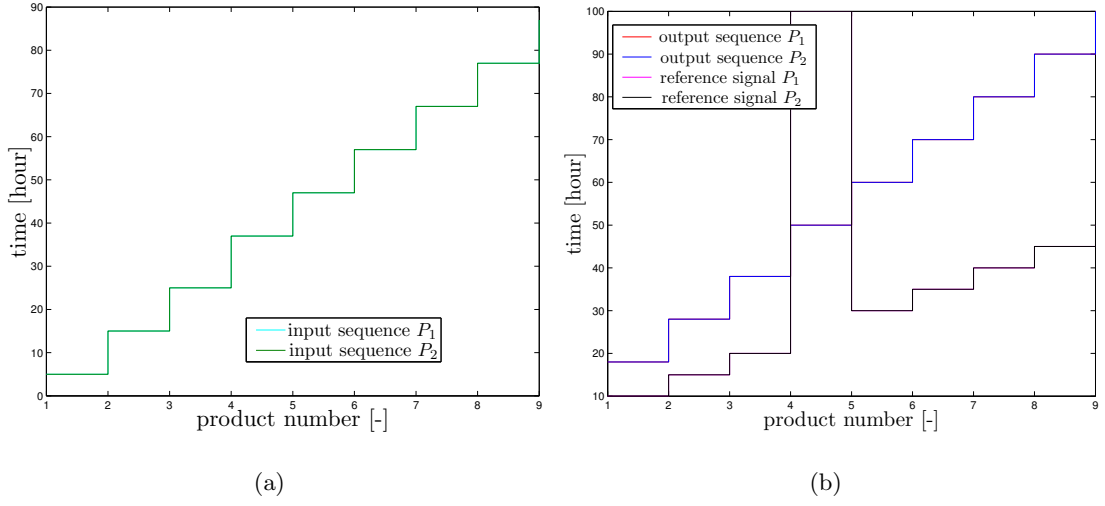
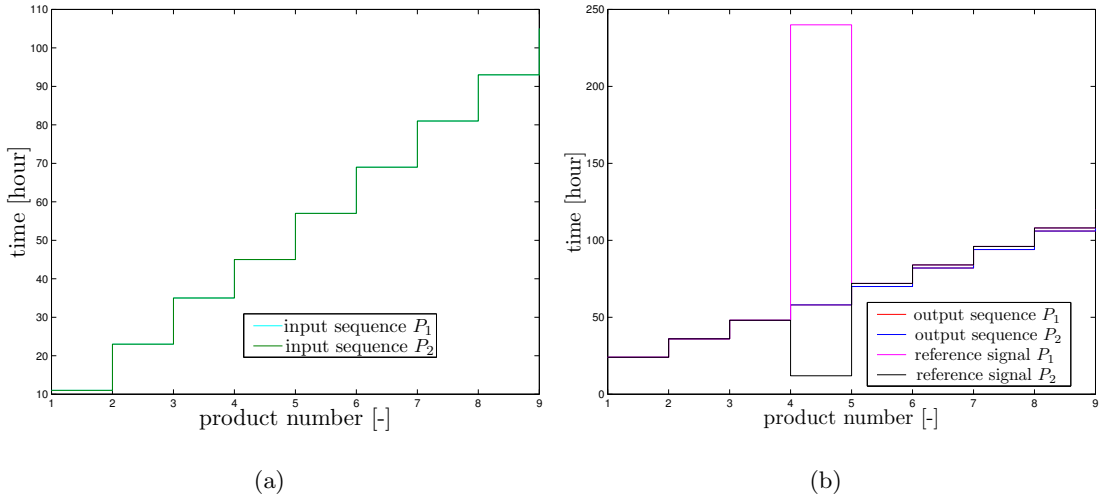
$P_2$  is expected to become dominant. Criterium (5.21) does not equal zero due to the non achievable reference signal in feed number 4.

system feed nr.	1	2	3	4	5	6	7	8	9
$\bar{r}_5(k)$ of $P_1$	24	36	48	60	72	84	96	108	120
$\bar{r}_5(k)$ of $P_2$	24	36	48	12	72	84	96	108	120

Table 5.6: Reference signal 5

The expectations of the behavior with respect to the fifth and last reference signal, which can be seen in Table 5.6, are correct, see Figures 5.8(a) and 5.8(b). The reference signal of the second product,  $P_2$  becomes dominant in feed number 4, with respect the the first product,  $P_1$ . Therefore, the input rate of both products becomes equal to the minimum cycle time, which is 10 time units. Due to feed number 4 and its constraints, the manufacturing system is not capable to follow the reference signal in feeds number 5 to 9.

Out of the simulation results can be concluded that an MPC controller has been designed that is able to let a max-plus model of a manufacturing system follow an output reference signal. If the reference signal not achievable by the manufacturing system, e.g. due to its upper- and lower bound with respect to the input rate, the input rate equals the upper bound, the lower bound or the average duration of a cycle.

Figure 5.7: Graphical representation of results due to reference signal  $r_4$ Figure 5.8: Graphical representation of results due to reference signal  $r_5$

## 5.5 Discussion

Chapter 5 presents an approach to control a manufacturing system that is modelled using the max-plus-algebra with MPC. This discussion consists of a summary of this chapter and a short revision of the most important observations. Before MPC can be used to control a manufacturing system, first a model has to be made. In this report, the max-plus-algebra is used as a tool to model and analyse DESs. Besides the max-plus-algebra, many tools exist to describe the behavior of a manufacturing system. Instead of the max-plus-algebra, any another suitable tool can be used in combination with MPC to control a manufacturing system. As mentioned in Section 4.8, the use of max-plus algebra has, as most tools, both advantages and disadvantages. An important remark that goes with the implementation of the max-plus model in combination with MPC is that the use of both the conventional and the max-plus algebra can cause some difficulties. Especially the use of  $\varepsilon$  in combination with the conventional algebra due to MPC results in a sometimes untidy and time-consuming approach as a whole. In the max-plus-algebra,  $\varepsilon$  can be seen as the 'zero'-element. Using both the algebra's, this causes some difficulties. The elegant state-space description of the manufacturing system gains an orderly effect, but this advantage decreases if this algebra is used in combination with another algebra. Before the max-plus-algebra can be used to control manufacturing systems, a recommendation is to introduce a certain level of automation. Using this automation, the entire approach of MPC control in combination with the max-plus-algebra becomes less time-consuming and more suitable to analyse, model and control manufacturing systems.

The manufacturing system that is used in combination with MPC contains all sorts of structures and a certain product mix. This system proved to be a good case study for modelling and analysis using the max-plus-algebra. Unfortunately the discrete behavior of both the product route and the process time turn out not to give the diverse manufacturing system as desired. Time instants at which products leave the system if the input time instants are known can easily be computed by hand. For the opposite situation, if desired output time instants are known, the same is valid. This, in combination with the dominance of the single lot machine with the largest process time and the batch machine resulted in a system with predictable dynamics. Therefore, both the prediction and the control horizon do not have to exceed a length of one sample. The simulations show, that MPC can be used to control this manufacturing system. An item for future research is, to replace this theoretical case by a more complex manufacturing system if necessary with stochastic process times. Using this alternate manufacturing system in combination with MPC might result in less predictable results and more insight about the strengths and weaknesses of the combination between max-plus-algebra and MPC.

The remainder of this thesis consists of a conclusion of the total thesis followed by recommendations for future research.

## Chapter 6

# Conclusions and recommendations

Many frameworks exist to analyze and model DESs such as manufacturing systems. The most widely used method is computer simulation, for example the formalism  $\chi$ . A model that is modelled using  $\chi$  does not always give a real understanding and/or explanation of the effects of parameter changes on properties of modelled systems. Therefore, mathematical models are preferred for analysis. One of these mathematical tools is the max-plus-algebra. In this thesis, the max-plus-algebra has been used as a tool to build and analyse these models. Afterwards a max-plus model has been used to control the output sequence of the manufacturing system with respect to a given desired output sequence of finished products. This has been done using Model Predictive Control (MPC). In this chapter the conclusions and recommendations for further research are given.

### 6.1 Conclusions

The max-plus-algebra has been used to model, analyse and control manufacturing systems. In this section, a revision of the most important observations and conclusions are presented. The limits and (dis)advantages of the max-plus-algebra are discussed and are compared to the  $\chi$  language.

#### Max-plus-algebra and manufacturing systems

The max-plus-algebra is a tool to model and analyse manufacturing systems. This max-plus-algebra has been studied and analyzed in detail. At the beginning of this report, the basic operations, maximization or max-plus addition,  $\oplus$ , and addition or max-plus multiplication,  $\otimes$ , have been explained and the algebra's main elements and standard

matrices have been discussed. The max-plus-algebra has been defined as  $\mathbb{R} \cup \{\varepsilon\}$ ,  $\oplus$ , and  $\otimes$ . In some cases, the max-plus-algebra has been compared to the conventional algebra to show similarities and differences.

In industry, the main elements of manufacturing systems are machines and (in)finite buffers. These elements and some other basic structures that are used in industry, for instance merging and batching, have been modelled using the max-plus-algebra. Machines and (in)finite buffers are in the max-plus-algebra modelled with great similarity. Main elements such as machines and buffers, have two incoming streams and two outgoing streams. One input receives availability information of the coupled structure in downstream direction, while the other input receives products from the coupled structure in upstream direction. The same is valid for the outputs, one output sends finished products to the coupled structure in downstream direction, the other output sends its availability information to the structure upstream. In contrast with the conventional algebra, where models of DESs lead to non-linearity, in the max-plus-algebra, these DESs can be written in a max-linear time invariant state-space description of the form:

$$\begin{aligned} x(k+1) &= A \otimes x(k) \oplus B \otimes u(k) \\ y(k) &= C \otimes x(k) \oplus D \otimes u(k) \end{aligned}$$

In this standard state-space description, matrices  $A$ ,  $B$ ,  $C$  and  $D$  are the system matrices. The states are in general time instants at which a buffer or machine starts respectively processing or buffering, the inputs are time instants at which a structure receives its raw material and the outputs are time instants at which a structure sends its finished products away. In this max-plus model, synchronization is addressed whereas concurrency is neglected. Synchronization requires the availability of several resources (e.g. machines) at the same time. Concurrency appears when a choice has to be made between the use of several resources, or, in other words, the product flow through the systems is variable [Sch96]. Both the process times and the product route through the manufacturing system are deterministic. The process times include setup times, while machines are assumed to be reliable and transportation times are neglected. Machines and (in)finite buffers are presented in a standard form because they are often used in every manufacturing system.

Since machines and buffers can be modelled separately, entire manufacturing systems can be modelled using a general coupling approach. This approach makes it possible to couple ( $u(k) = y(k)$ ) structures that are modelled separately to obtain entire manufacturing systems with all kinds of structures, for example splitting, merging, batching, and re-entrancy. A model of a large manufacturing system obtains, using the use of a standard state-space description, large system matrices. Therefore, model reduction has been discussed shortly in this report.

## Modelling and analysis

To test the observations discussed above and the standard forms of machines and buffers, a test case has been introduced. This case contains a manufacturing system with all sorts of structures and a certain product mix. The objective of this theoretical case is to study the limits and (dis)advantages of the max-plus algebra with respect to modelling and analyzing manufacturing systems. The manufacturing system of the theoretical case has been built using the standard coupling approach. Many observations have been done by modelling the entire manufacturing system of the theoretical case using the max-plus-algebra. These observations can be used to compare the max-plus-algebra with  $\chi$ . Using both tools, each process or structure is modelled separately and an entire manufacturing system is obtained by coupling inputs and outputs. The max-plus-algebra, as well as the  $\chi$  language describe the behavior (both the transient and the steady state) of a manufacturing system exactly. The max-plus-algebra describes the behavior of a manufacturing system using an elegant state-space description that provides good insight in the dynamics of a manufacturing system. If input time instances are known, the output time instants of the finished products and the time instants at which machines start processing can be determined easily, due to the deterministic behavior. Unfortunately, this insight decreases if the size and/or complexity of the manufacturing system increases due to the increasing size of the system matrices. Here, complexity can depend on e.g. certain scheduling rules, the structure of the system, re-entrancy etc. If a manufacturing system is modelled using  $\chi$ , these (relative) complex situations can be modelled with functions outside the process environment. Large structures that contain similar processes can be modelled by defining the process only once and duplicate it in the *xper*-environment or using a cluster. However, even the use of functions and the duplication of processes in the *xper*-environment in the  $\chi$  model might also lead to a decrease of insight of a manufacturing system.

Unfortunately, due to the limited duration of this project, up to now, only manufacturing systems with both deterministic process times and product route are considered. Using the  $\chi$  language to model and analyse manufacturing systems, stochastic process times and a non-fixed route are items that can already be implemented. The standard blocks of machines and buffers cannot be used if a certain policy or a special structure is required. Modelling a manufacturing system using both  $\chi$  and the max-plus-algebra is time-consuming. Once both models are finished and a certain policy and/or a process changes or an addition of one or more process(es) takes place, using the max-plus-algebra this alternation takes time to change. A change in the  $\chi$  specification might also be time consuming, but in general, this model adaptation can be done in less time. Therefore, the max-plus-algebra is, up to now, not useful if simulations are required to determine the influence on certain parameters or the performance of a manufacturing system with respect to for instance the number of machines or the number of buffer places. Manufacturing systems in which non-determinism takes place are not considered in this thesis. Recommendations with respect to the use of the max-plus-algebra in combination with a non-fixed product route and stochastic process times are done in

## Section 6.2.

**Max-plus-algebra and MPC**

Due to the standard max-algebraic state-space description of a manufacturing system, conventional control techniques can be used. In this report, MPC is used to control the output sequence of a manufacturing system with respect to a desired reference signal. MPC is a relatively recently developed control strategy that can deal with constraints on the states, inputs and outputs. This constraint handling and the use of a moving or receding horizon, makes MPC a suitable discrete event time controller to control a manufacturing system. MPC finds an optimal input sequence with respect to an objective function. In this thesis, the optimization is done by means of Linear Programming (LP). LP is chosen due to difficulties with other optimization methods and the LP suitability to solve the standard optimization problem. A standard MPC optimization problem is composed. This standard problem contains the dynamics of the manufacturing system of the theoretical case through a slightly adapted state-space description. Due to the use of LP, the non-linearities have to be eliminated. The operator that causes the non-linearity is  $\oplus$ , which is equal to the maximization operator in the conventional algebra. The maximization operator can be reformulated such that it becomes suitable with respect to LP. Due to the transformation of the maximization that is required to obtain an LP problem, the non-linear, non-convex max-plus dynamics becomes convex, implies that the standard MPC problem is replaced by a relaxed MPC problem. Implementation of the optimal input sequence of the relaxed problem in the state-space description of the manufacturing system, gains the optimal output sequence with respect to the original MPC problem. To gain satisfactory results using MPC, first certain parameters have to be tuned. Both the control and the prediction horizon are parameters to tune. Determination of the length of these parameters can be done using a max-algebraic unit impulse. The use of this unit impulse results in a length of one sample for both. This means that the manufacturing system in the case study does not need future information to follow the desired reference signal with respect to its constraints. Simulations implementing several different reference signals gave expected and satisfactory results.

An observation that goes with the implementation of the max-plus model in combination with MPC, is that the use of both the max-plus-algebra and the conventional algebra causes some difficulties. Especially the use of  $\varepsilon \stackrel{\text{def}}{=} -\infty$  and the max-plus state-space description in combination with the conventional algebra results in a sometimes untidy and time-consuming approach as a whole. The elegant state-space description of the manufacturing system is surveyable, but this advantage decreases if this algebra is used in combination with the conventional algebra.

The manufacturing system that is used as a theoretical case proved to be a good study for modelling and analysis. Unfortunately the dominant behavior of the batch machine with a large process time, the single lot machine with the largest process time compared



to the other single lot machine and the fixed product route and deterministic process times did not turn out to have the diverse dynamic behavior as desired. Therefore, controlling a manufacturing system in combination with MPC seems to be messy as a whole. However, a more dynamic case study might give some more information about the strength of the combination of the max-plus-algebra and MPC.

## 6.2 Recommendations for future research

During this research project, several questions about the max-plus-algebra have come up that could not be solved due to the limited duration of the project. In this section, the most important questions are treated here. Based on these questions, recommendations, proposals and basic ideas that might be useful for further research are formulated.

### Proposal to implement stochastic process times

An item that has not been discussed in this report, is the use of stochastic process times. In this thesis only deterministic process times are used to model manufacturing systems. Using for example the  $\chi$  language to model industrial systems, stochastic process times are modelled using a sample of a certain distribution. The same can be done using the max-plus-algebra. This results in a difference between process times of different samples, or in other words it might be possible that  $d_i(k)$  does not equal  $d_i(k+1)$ . This results in system matrices that depend on  $k$ . To make this more clear, the standard machine is discussed in Section 3.3. A schematic representation of this machine with process time  $d(k)$  can be seen in Figure 6.1.

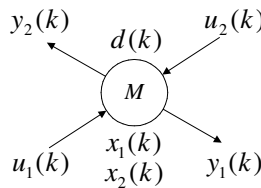


Figure 6.1: Machine with stochastic process time

The state-space description of this machine with stochastic process times becomes different. The time instant at which the machine starts processing for the  $(k+1)$ st time becomes:

$$x_1(k+1) = \max(u_1(k), x_1(k) + d(k), x_2(k)) \quad (6.1)$$

$$x_2(k+1) = u_2(k). \quad (6.2)$$

The time instant at which a product leave the system for the  $(k + 1)$ st becomes

$$y_1(k) = \max(x_1(k + 1) + d(k + 1), u_2(k)). \quad (6.3)$$

Substitution of (6.1) in (6.3) results in:

$$y_1(k) = \max(u_1(k) + d(k + 1), x_1(k) + d(k) + d(k + 1), x_2(k) + d(k + 1), u_2(k) + d(k + 1)). \quad (6.4)$$

It can be seen that (6.4) contains two different process times of two different samples. This results in variable system matrices that are dependent on  $k$ :  $A(k)$ ,  $B(k)$ ,  $C(k)$  and  $D(k)$ . This way of implementing stochastic process times is a recommendation for future research.

### Proposal to implement non-fixed product routes

One of the main shortcomings of the max-plus-algebra up to now, is that the product route through a manufacturing system has to be fixed. Flexible manufacturing systems cannot be modelled using the max-plus-algebra due to the flexible product route. Due to the two basic operations of the max-plus-algebra,  $\oplus$  and  $\otimes$ , the modelling of manufacturing system is limited with respect to this product route. Therefore, a proposal is to introduce a third operator that is needed to model non-fixed product route of products through a manufacturing system. This third operator is minimization. Due to the introduction of this third operator an entire new algebra is created, the max-min-plus-algebra, see [Bac92]. Now, a simple example, see Figure 6.2, including minimization is given to explain this basic idea. For reasons of clarity, the availability information is, in this example, not taken into account.

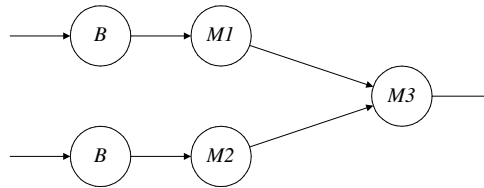


Figure 6.2: Schematic representation of proposal

In Figure 6.2, a simple production system can be seen. The two parallel single lot machines have stochastic process times with mean process times  $d_1 = d_2$ . The machine that finishes its product first, sends its finished product to the third single lot machine with a certain process time  $d_3$ . Due to the stochastic process times, the product route through the manufacturing system is non-fixed. In case of both deterministic process times and the introduction of machine break downs, the above mentioned way to model

a non-fixed product route might be suitable. Using the max-plus-algebra as discussed and worked out in this report, this simple manufacturing system cannot be modelled. Using the so called max-min-plus systems the standard state-space description as used in this report has to be adapted. Machine 3 can receive a finished product if it finished its previous product and if the first incoming product of machine 1 or machine 2 has arrived. This can be written down as follows:

$$x_3(k+1) = \max(\min(x_1(k+1) + d_1, x_2(k+1) + d_2), x_3(k) + d_3). \quad (6.5)$$

Using only (6.5), the last product that arrives at machine 3 is not processed at all, but leaves the system in some mysterious way. Therefore, a fourth equation has to be introduced that takes care of last arriving product at machine 3. Using this fourth equation results in a change of (6.5):

$$x_3(k+1) = \max(\min(x_1(k+1) + d_1, x_2(k+1) + d_2), x_4(k) + d_3) \quad (6.6)$$

$$x_4(k+1) = \max(\max(x_1(k+1) + d_1, x_2(k+1) + d_2), x_3(k+1) + d_3). \quad (6.7)$$

Substitution of (6.6) in (6.7) results in:

$$x_3(k+1) = \max(\min(x_1(k+1) + d_1, x_2(k+1) + d_2), x_4(k) + d_3) \quad (6.8)$$

$$x_4(k+1) = \max(x_1(k+1) + d_1, x_2(k+1) + d_2, \min(x_1(k+1) + d_1 + d_3, x_2(k+1) + d_2 + d_3), x_4(k) + 2d_3). \quad (6.9)$$

The use of the minimization operator might be a possible approach to model non-fixed product routes.

## General proposals

Now, a certain class of manufacturing systems can be modelled using the max-plus-algebra, still a lot of work has to be done. In this section some proposals are done for future research, which might increase the strength and suitability of the max-plus-algebra.

In this report some statements have been done about the sometimes time-consuming and untidy approach as a whole if the max-plus-algebra is used in combination with the conventional algebra. This is caused especially by  $\varepsilon$ . A recommendation to solve this problem, is to introduce a certain automation level. The use of this automation level might result in a more user-friendly tool to model, analyse and control manufacturing systems.

In this thesis, disturbances and noise have not been considered. A common disturbance with respect to manufacturing systems are machine break downs. A proposal to model these machine break downs using the max-plus-algebra is to abruptly enlarge the length of the process time of a machine to simulate a break down. By enlarging the process time of a machine, it cannot receive any products. Therefore, this state can be considered as down. Using this approach, machine break downs might be modelled using the max-plus-algebra.

The manufacturing system that has been used as a theoretical case does not have the dynamical behavior as desired. Therefore, a recommendation for future research is to investigate a manufacturing system with more structures (eg. re-entrancy) and policies that increase the diversity of the dynamical behavior of the system. This in combination with the use of MPC might give more information about the limits and strengths of the max-plus-algebra.

The use of LP to determine an optimal solution gives satisfactory results, except when the weighting parameter that makes a trade off between minimization of the tracking error and the needed control effort equals zero. Using this value for  $\lambda$  the optimal solution of the relaxed MPC problem is not equal to the original MPC problem. This might be caused by the used LP formulation. A recommendation for future research is to implement another LP formulation that looks after an optimal solution of the relaxed MPC problem that is an optimal solution of the original MPC problem, since the set of feasible solutions of the original problem is a subset of the set of feasible solutions of the relaxed problem.

A last recommendation with respect to future research is the study of model reduction. In this research, this model reduction has been done by manually search of max-plus linear combinations. This way of searching is not efficient. A recommendation is to increase the efficiency of this model reduction.

# Bibliography

- [Bab03] Babylon information @ a click. Babylon online dictionary. [www.babylon.com](http://www.babylon.com), 2003.
- [Bac92] F. Baccelli, G. Cohen, G.J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: an algebra for discrete event systems*. Wiley, 1992.
- [Cas95] C.G. Cassandras, S. Lafortune, and G.J. Olsder. *Introduction to the modelling, control and optimization of Discrete Event Systems*. Springer-Verlag, 1995.
- [Ess02] H.A. van Essen and M. Steinbuch. *Lecture notes on model predictive control 'Capita Selecta in Control'*. Eindhoven University of Technology, 2002.
- [Gau92] S. Gaubert. *Théorie des Systèmes Linéaires dans les Dioïdes*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, France, 1992.
- [Gau94] S. Gaubert. On rational series in one variable over certain dioids. Technical Report 2162, INRIA, Le Chesnay, France, 1994.
- [Hof02] A.T. Hofkamp and J.E. Rooda.  *$\chi$  reference manual*. Systems Engineering Group, Eindhoven University of Technology, <http://se.wtb.tue.nl>, 2002.
- [Kle92] J.P.C. Kleijnen. Verification and validation of models. Technical report, Tilburg University, department of economics, 1992.
- [Pap00] P.Y. Papalambros and D.J. Wilde. *Principles of optimal design*. Cambridge University Press, 2000.
- [Sch94] B. de Schutter and B. de Moor. The characteristics equation and minimal state-space realization of siso systems in the max algebra. In G. Cohen and J.P. Quadrat, editors, *11th International Conference on Analysis and Optimization of Systems*, volume 199, pages 273–282. Springer-Verlag, 1994.
- [Sch95] B. de Schutter and B. de Moor. The extended linear complementarity problem. *Mathematical Programming*, 71(3):289–325, December 1995.
- [Sch96] B. de Schutter. *Max-algebraic system theory for discrete event systems*. PhD thesis, Katholieke Universiteit Leuven, 1996.

- [Sch97] B. de Schutter, R. de Vries, and G.J. Olsder. The minimal realization problem in the max-plus algebra: An overview. Technical report, Katholieke Universiteit Leuven, 1997.
- [Sch00a] B. de Schutter and T. van den Boom. Model predictive control for max-plus-linear systems. *Proceedings of the 2000 American Control Conference*, pages 4046–4050, 2000.
- [Sch00b] B. de Schutter and T. van den Boom. Model predictive control for max-plus-linear systems: closed-loop behavior and tuning. *Proceedings of the Workshop on Systems with Time-Domain Constraints*, pages 4046–4050, 2000.
- [Sch01] B. de Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37:1049–1056, 2001.
- [Sch02] B. de Schutter and T. van den Boom. Model predictive control for perturbed max-plus-linear systems. *Systems and Control Letters*, 37(1):21–33, January 2002.
- [Sta03] J. Stanczyk. *Max-plus algebra toolbox for Matlab*. University of Magdeburg, Germany, 0.1 edition, August 2003.
- [Van01] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 2001.
- [Wei03] S. Weiland. Model approximations of dynamic systems. Hand-out Stochastic Systems Theory, June 2003.

## Appendix A

# Algorithms to calculate system matrices

This appendix represents the algorithm to calculate the system matrices  $A$ ,  $B$ ,  $C$  and  $D$  of a machine or finite buffer who receives  $n$  products per system feed. Note that for the algorithms presented in this appendix,  $n > 1$ . The process times of the machines are indexed as follows:  $d_i$ . Index,  $i$ , represents the number of the product that is fed to the machine, for instance, the process time of product 2 of a certain feed is  $d_2$ . In these matrices, some of the  $\varepsilon$ 's are indexed. These single indices are added to indicate the number of columns only filled with  $\varepsilon$ 's of this heuristic. For instance,  $\varepsilon^1 \dots \varepsilon^{n-1}$  represents a block of  $(n - 1)$  columns filled  $\varepsilon$ 's

$$A_{n \times n} = \begin{pmatrix} \varepsilon^1 & \dots & \varepsilon^{n-1} & d_n & \varepsilon^1 & \dots & \varepsilon^{n-1} & 0 \\ \varepsilon & \dots & \varepsilon & \sum_{i=1}^1 d_i + d_n & \varepsilon & \dots & \varepsilon & d_1 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \varepsilon & \dots & \varepsilon & \sum_{i=1}^{n-1} d_i + d_n & \varepsilon & \dots & \varepsilon & \sum_{i=1}^{n-1} d_i \\ \varepsilon & \dots & \varepsilon & \varepsilon & \varepsilon & \dots & \varepsilon & \varepsilon \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \varepsilon & \dots & \varepsilon & \varepsilon & \varepsilon & \dots & \varepsilon & \varepsilon \end{pmatrix}$$

$$B_{n \times n} = \begin{pmatrix} 0 & \varepsilon & \cdots & \cdots & \varepsilon & \varepsilon & \cdots & \cdots & \cdots & \varepsilon \\ d_1 & 0 & \ddots & & \vdots & 0 & \ddots & & & \vdots \\ \sum_{i=1}^2 d_i & d_2 & \ddots & \ddots & \vdots & d_2 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \varepsilon & \vdots & \ddots & \ddots & \ddots & \varepsilon \\ \sum_{i=1}^{n-1} d_i & \sum_{i=2}^{n-1} d_i & \cdots & d_{n-1} & 0 & \sum_{i=2}^{n-1} d_i & \cdots & d_{n-1} & 0 & \varepsilon \\ \varepsilon & \cdots & \cdots & \cdots & \varepsilon & 0 & \varepsilon & \cdots & \cdots & \varepsilon \\ \vdots & & & & \vdots & \varepsilon & 0 & \ddots & & \vdots \\ \vdots & & & & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & \vdots & \vdots & & \ddots & \ddots & \varepsilon \\ \varepsilon & \cdots & \cdots & \cdots & \varepsilon & \varepsilon & \cdots & \cdots & \varepsilon & 0 \end{pmatrix}$$

$$C_{n \times n} = \begin{pmatrix} \varepsilon & \cdots & \cdots & \varepsilon & \sum_{i=1}^1 d_i + d_n & \varepsilon & \cdots & \cdots & \varepsilon & d_1 \\ \vdots & & & \vdots & \vdots & \vdots & & & \vdots & \vdots \\ \varepsilon & \cdots & \cdots & \varepsilon & \sum_{i=1}^{n-1} d_i + d_n & \varepsilon & \cdots & \cdots & \varepsilon & \sum_{i=1}^n d_i \\ d_1 & \varepsilon & \cdots & \cdots & \varepsilon & 0 & \varepsilon & \cdots & \cdots & \varepsilon \\ \varepsilon & \ddots & \ddots & & \vdots & \varepsilon & 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \varepsilon & \vdots & & \ddots & \ddots & \varepsilon \\ \varepsilon & \cdots & \cdots & \varepsilon & d_n & \varepsilon & \cdots & \cdots & \varepsilon & 0 \end{pmatrix}$$

$$D_{n \times n} = \begin{pmatrix} d_1 & \varepsilon & \cdots & \varepsilon & 0 & \varepsilon & \cdots & \varepsilon \\ \sum_{i=1}^2 d_i & d_2 & \ddots & \vdots & d_2 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \varepsilon & \vdots & \ddots & \ddots & \varepsilon \\ \sum_{i=1}^n d_i & \sum_{i=2}^n d_i & \cdots & d_n & \sum_{i=2}^n d_i & \cdots & d_n & 0 \\ \varepsilon & \cdots & \cdots & \varepsilon & \varepsilon & \cdots & \cdots & \varepsilon \\ \vdots & & & \vdots & \vdots & & & \vdots \\ \varepsilon & \cdots & \cdots & \varepsilon & \varepsilon & \cdots & \cdots & \varepsilon \end{pmatrix}$$



## Appendix B

# Details model reduction

In this Appendix, ways for reducing the number of states of max-plus models are discussed. Unfortunately, all these ideas have major disadvantages which are presented at the end of the subsections.

Before definitions as state-space realization can be discussed, first the definition and the determination of the impuls responses have to be explained. The standard state-space description (see Chapter 3) is:

$$\begin{aligned}x(k+1) &= A \otimes x(k) \oplus B \otimes u(k) \\ y(k) &= C \otimes x(k) \oplus D \oplus u(k)\end{aligned}$$

Using the following unit impuls response:

$$\begin{cases} e(k) = 0 & \text{if } k = 0 \\ e(k) = \varepsilon & \text{if } k \neq 0 \end{cases}$$

to the  $i$ th input of the system and if  $x(0) = \varepsilon_{n \times 1}$ , then  $y(k) = C \otimes A^{\otimes k-1} \otimes B_i$  for  $k = 1, 2, \dots$  as the output of the system, where  $B_i$  is the  $i$ th column of  $B$ . If this is done for all the inputs  $i = 1, 2, \dots, m$  of the system, and this can be written using matrix  $G_{k-1} = C \otimes A^{\otimes k-1} \otimes B$  for  $k = 0, 1, \dots$ . The  $G_k$ 's are called *impuls response matrices* or *Markov parameters* [Sch96].

Consider a common max-linear time-invariant DES with  $m$  inputs and  $l$  outputs that can be described by an  $n$ th order state-space model of the form (3.1a) and (3.1b). Suppose that the matrices  $A$ ,  $B$  and  $C$  are unknown, and that only the impuls response  $g_k = \{G_k\}_{k=0}^{\infty}$  are known. Constructing the matrices  $A$ ,  $B$  and  $C$  from the impuls response is called *state-space realization*. If the dimension of  $A$  is chosen to be minimal, then the dimension of  $A$ , is equal to the minimal system order and matrices triple  $A$ ,  $B$  and  $C$  is called a minimal state-space realization of  $g_k = \{G_k\}_{k=0}^{\infty}$ . The problem described above has, at present, not been solved entirely [Sch94]. This minimal state-space realization problem for max-linear time-invariant DESs has been studied by many

authors and for some specific cases the problems have been solved. In the rest of this chapter, the minimal system order and the minimal state-space realization are discussed. This is given in an overview of the obtained results so far.

## B.1 The minimal system order

In conventional system theory, the minimal system order is given by the rank of the Hankel matrix  $H(\infty, \infty)$ . However, in contrast to linear algebra, the different notions of rank (like column rank, row rank, minor rank etc.) are in general not equivalent in the max-plus-algebra. In the literature upper and lower bounds of the minimal system order can be found. Lower bounds can be found computing the minor rank and the Schein rank of the Hankel matrix [Gau92, Gau94]. At present, there are no efficient (i.e., polynomial time) algorithms to compute the max-plus-algebraic minor rank or the Schein rank of a matrix. The upper bound of the system can be found by computing the weak column rank. Many efficient methods exist, to compute the max-algebraic weak column rank of a matrix [Sch97].

## B.2 Minimal state-space realization

Until now, there are three different ways to compute a minimal state-space realization can be distinguished: transformation to conventional algebra, partial state-space realization and special sequences of Markov parameters. These three different ways of computing the minimal state-space realization are discussed in the following subsections.

### Transformation to conventional algebra

There exists a transformation from the max-plus-algebra to linear algebra that is based on the following equivalences:

$$\begin{aligned} x \oplus y &= z \Leftrightarrow e^{xs} + e^{ys} \sim ce^{zs}, \quad s \rightarrow \infty \\ x \otimes y &= z \Leftrightarrow e^{xs} \cdot e^{ys} = e^{zs} \quad \text{for all } s > 0 \end{aligned}$$

with  $x, y, z \in \mathbb{R}_\varepsilon$ , and  $c = 2$  if  $x = y$  and  $c = 1$  otherwise. The symbol  $\sim$  stands for asymptotic equivalence.

Using this transformation, a minimal realization problem in the max-plus-algebra can be mapped to a minimal realization problem for matrices with exponentials as entries and with conventional addition and multiplication as basic operations [Sch97]. Now the techniques used in the conventional algebra to calculate the minimal state-space realization can give a result. The only step that is left now, is the transformation back to the max-plus-algebra. However, only realizations with positive coefficients for the

leading exponentials can be mapped back to the max-plus-algebra, and it is not always obvious how and whether such a realization can be constructed. In general the minimal system order obtained using the procedure above is a lower bound for the minimal system order.

### Partial state-space realization

This is the second way of computing the minimal state-space realization. Given is a finite sequence  $g_1, g_2, \dots, g_N$ , find matrices  $A, B$  and  $C$ , such that  $G_k = C \otimes A^{\otimes k} \otimes B$  holds for  $k = 1, 2, \dots, N$ . It can be shown that this leads to a system of so-called max-plus-algebraic polynomial equations and that such a system can be recast as a mathematical programming problem that is called the Extended Linear Complementarity Problem (ELCP) [Sch96]. This procedure can also be used for MIMO systems. This enables that the partial minimal realization problem can be solved and by applying some limit arguments this results in a realization of the entire impuls response. However, it can be shown that the general ELCP is NP-hard [Sch95]. A problem is NP-hard if an algorithm for solving it can be translated into one for solving any other NP-problem (nondeterministic polynomial time) problem. Due to the limitation of this project, NP-problems that might be difficult to solve, have not been considered.

### Special sequences of Markov parameters

The last manner to calculate the minimal state-space realization is a way that only can be used in special cases. Only then methods exist to compute on a efficient way the minimal state-space realization. These two cases are [Sch97]:

- if the sequence  $g_k = \{G_k\}_{k=0}^{\infty}$  exhibits uniformly up-terrace behavior, i.e., if it consists of a concatenation of, say,  $m$ , subsequences with rates  $c_1, c_2, \dots, c_M$ , where in the  $k$ th subsequence the following is valid:  $g_{i+1} = g_i + c_k$  and  $c_1 < c_2 < \dots < c_M$ ,
- if the sequence  $g_k = \{G_k\}_{k=0}^{\infty}$  exhibits a convex transient behavior and an ultimately geometric behavior with period 1:

$$\begin{aligned} g_{k+1} - g_k &\geq g_k - g_{k-1} && \text{for } k = 2, \dots, k_0 \\ g_{k+1} &= \lambda \otimes g_k && \text{for } k \geq k_0. \end{aligned} \tag{B.1}$$

However, this manner of minimizing the state-space realization can only be used in case of SISO models [Sch96]. Some models described at the end in Chapter 3 are MIMO systems.

### B.3 Summary

Some methods exist to compute the minimal system order and the minimal state-space realization. However, for example the lower bound necessary to calculate the minimal system order is a complex process (see Section B.1). The minimal state-space realization can be computed with three methods described in Section B.2. Unfortunately, all three methods have major disadvantages, which are represented at the end of the subsections. The objectives as discussed in Chapter 1 limits the number of items that can be investigated in the time that stands for this Master's thesis. This explains why is not chosen to use the techniques mentioned above, but to search manually for max-linear combinations to reduce the models.

## Appendix C

### System matrices theoretical case

In this appendix the system matrices  $A$ ,  $B$ ,  $C$ , and  $D$  of the theoretical case as worked out in Chapter 4 are presented.

$$A = \begin{pmatrix} \cdot & 0 & \cdot & \cdot & \cdot \\ \cdot & 0 & \cdot & \cdot & \cdot \\ \cdot & 0 & 0 & d_1 & 0 \\ \cdot & 0 & 0 & d_1 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 0 & d_1 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & d_1 & d_1 & 2d_1 & d_1 \\ \cdot & d_1 \oplus d_2 & d_1 \oplus d_2 & 2d_1 \otimes (d_1 \oplus d_2) & d_1 \oplus d_2 \\ \cdot & d_1 & d_1 & 2d_1 & d_1 \\ \cdot & d_1 \oplus d_2 & d_1 \oplus d_2 & 2d_1 \otimes (d_1 \oplus d_2) & d_1 \oplus d_2 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & d_1 \oplus d_2 & d_1 \oplus d_2 & 2d_1 \otimes (d_1 \oplus d_2) & d_1 \oplus d_2 \\ \cdot & d_1 \oplus d_2 \otimes d_3 & d_1 \oplus d_2 \otimes d_3 & (2d_1 \oplus d_1 \otimes d_2) \otimes d_3 & d_1 \oplus d_2 \otimes d_3 \\ \cdot & d_1 \oplus d_2 \otimes d_3 & d_1 \oplus d_2 \otimes d_3 & (2d_1 \oplus d_1 \otimes d_2) \otimes d_3 & d_1 \oplus d_2 \otimes d_3 \end{pmatrix}$$

$$\begin{pmatrix}
 \dots & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \dots & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \dots & d_2 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \dots & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \dots & \cdot & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot \\
 \dots & d_2 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \dots & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \dots & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \dots & 2d_2 & d_2 & \cdot & 0 & 0 & 0 & \cdot & \cdot & \cdot \\
 \dots & \cdot & \cdot & \cdot & 0 & \cdot & 0 & 0 & \cdot & \cdot \\
 \dots & 2d_2 & d_2 & \cdot & 0 & 0 & 0 & 0 & \cdot & \cdot \\
 \dots & \cdot & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot \\
 \dots & 2d_2 & d_2 & \cdot & 0 & 0 & 0 & 0 & d_3 & 0 \\
 \dots & 2d_3 \otimes d_3 & d_2 \otimes d_3 & \cdot & d_3 & d_3 & d_3 & d_3 & 2d_3 & d_3 \\
 \dots & 2d_2 \otimes d_3 & d_2 \otimes d_3 & \cdot & d_3 & d_3 & d_3 & d_3 & 2d_3 & d_3
 \end{pmatrix} \quad (C.1)$$

$$B = \begin{pmatrix} 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ d_1 & \cdot & \cdot & \cdot \\ d_1 \oplus d_2 & d_2 & \cdot & \cdot \\ d_1 & \cdot & \cdot & \cdot \\ d_1 \oplus d_2 & d_2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ d_1 \oplus d_2 & d_2 & \cdot & \cdot \\ d_1 \oplus d_2 \otimes d_3 & d_2 \otimes d_3 & 0 & \cdot \\ d_1 \oplus d_2 \otimes d_3 & d_2 \otimes d_3 & 0 & 0 \end{pmatrix} \quad (\text{C.2})$$

$$C = \begin{pmatrix} \cdot & d_1 \oplus d_2 \otimes d_3 & d_1 \oplus d_2 \otimes d_3 & (2d_1 \oplus d_1 \otimes d_2) \otimes d_3 & d_1 \oplus d_2 \otimes d_3 \\ \cdot & d_1 \oplus d_2 \otimes d_3 & d_1 \oplus d_2 \otimes d_3 & (2d_1 \oplus d_1 \otimes d_2) \otimes d_3 & d_1 \oplus d_2 \otimes d_3 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} \dots & 2d_2 \otimes d_3 & d_2 \otimes d_3 & \cdot & d_3 & d_3 & d_3 & d_3 & 2d_3 & d_3 & d_3 \\ \dots & 2d_2 \otimes d_3 & d_2 \otimes d_3 & \cdot & d_3 & d_3 & d_3 & d_3 & 2d_3 & d_3 & d_3 \\ \dots & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \dots & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \quad (\text{C.3})$$

$$D = \begin{pmatrix} d_1 \oplus d_2 \otimes d_3 & d_2 \otimes d_3 & 0 & \cdot \\ d_1 \oplus d_2 \otimes d_3 & d_2 \otimes d_3 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \quad (\text{C.4})$$





## Appendix D

# Matlab model of the theoretical case

In this chapter the mathematical max-plus model of the theoretical case as described in Chapter 4 is presented. A max-plus-algebra toolbox for Matlab, has been used to calculate the output [Sta03]. An important note here is that in the loop, the vector  $u(k)$  has been labelled to  $(k + 1)$ .

```
clear all;
e=-inf;

% process times machines
d1=1; % process time machine 1
d2=3; % process time machine 2
d3=10; % process time batch machine

% system matrices A, B, C, and D

A=[ e 0 e e e e e e e e e e e;
    e 0 e e e e e e e e e e e;
    e 0 0 d1 0 d2 0 e e e e e e e;
    e 0 0 d1 0 e e e e e e e e e;
    e e e e e e e e e 0 e e e;
    e 0 0 d1 0 d2 0 e e e e e e e;
    e e e e e e e e e 0 e e e e;
    e d1 d1 2*d1 d1 e e e 0 e e 0 e e e;
    e mp_sum(d1,d2) mp_sum(d1,d2) mp_sum(2*d1,mp_multi(d1,d2)) mp_sum(d1,d2)...
    2*d2 d2 e 0 0 e 0 e e e;
    e d1 d1 2*d1 d1 e e e 0 e 0 0 e e 0;
    e mp_sum(d1,d2) mp_sum(d1,d2) mp_sum(2*d1,mp_multi(d1,d2)) mp_sum(d1,d2)...
    2*d2 d2 e 0 0 0 0 e e 0;
    e e e e e e e e e 0 e e e e;
    e mp_sum(d1,d2) mp_sum(d1,d2) mp_sum(2*d1,mp_multi(d1,d2)) mp_sum(d1,d2)...
    2*d2 d2 e 0 0 0 0 d3 0 0;
    e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
```

```

    mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
    mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3;
    e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
    mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
    mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3]

B=[ 0 e e e;
    0 0 e e;
    0 0 e e;
    0 e e e;
    e e e e;
    0 0 e e;
    e e e e;
    d1 e e e;
    mp_sum(d1,d2) d2 e e;
    d1 e e e;
    mp_sum(d1,d2) d2 e e;
    e e e e;
    mp_sum(d1,d2) d2 e e;
    mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3) 0 e;
    mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3) 0 0];

C=[e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
    mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
    mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3;
    e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
    mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
    mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3;
    e e e e e e e e e e e e e;
    e e e e e e e e e e e e e];

D=[mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3) 0 e;
    mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3) 0 e;
    e e e e;
    e e e e];

% initial values of x, u1, u2, u15, u16
x=[e; e; e; e; e; e; e; e; e; e; e; e; e; e; e];
u1=[e; 0; 5; 0; 15; 20; 25; 21; 35; 40; 39];
u2=[e; 0; 6; 11; 16; 11; 26; 31; 36; 41; 46];
u15=[e; e; e; e; e; e; e; e; e; e; 100; e; e; e];
u16=[e; e; e; e; e; e; e; e; e; e; 100; e; e; e];

% number of feeds
N=10;

% loop
for k=1:N
    u(:,k)=[u1(k+1);
            u2(k+1);
            u15(k+1);
            u16(k+1)];
    x(:,k+1)=mp_sum(mp_multi(A,x(:,k)),mp_multi(B,u(:,k)));
    y(:,k+1)=mp_sum(mp_multi(C,x(:,k)),mp_multi(D,u(:,k)));

```

end

```
%values on screen
% input values
u;
% x1: time instant at which the infinite buffer starts buffering product P1 for the kth time
% x2: time instant at which the infinite buffer starts buffering product P2 for the kth time
x1=[x(1,:)];
x2=[x(2,:)];
% x4: time instant at which machine 1 starts starts working on product P1 for the kth time
x4=[x(4,:)];
% x6: time instant at which machine 2 starts starts working on product P2 for the kth time
x6=[x(6,:)];
% x8: time instant at which the finite buffer starts buffering product P1 for the kth time
x8=[x(8,:)];
% x9: time instant at which the finite buffer starts buffering product P2 for the kth time
x9=[x(9,:)];
% x13: time instant at which the batch machine starts working on the batch (P1 and P2)
% for the kth time
x13=[x(13,:)];

% output vectors
u=[u(1,:); u(2,:)] % input
x=[x(1,:); x(2,:); x(4,:); x(6,:); x(8,:); x(9,:); x(13,:)] % states
y=[y(:,2) y(:,3) y(:,4) y(:,5) y(:,6) y(:,7) y(:,8) y(:,9) y(:,10) y(:,11)] % output
```



# Appendix E

## $\chi$ validation files

In this appendix, two  $\chi$ 0.8-files of the manufacturing line, discussed in Chapter 4 that are used to validate the max-plus model are presented. In Section E.1 a  $\chi$  file can be seen that is modelled using the standard modelling techniques according to [Hof02]. In Section E.2 a  $\chi$  file is presented that is modelled such that its actions and events corresponds to the actions and events of the max-plus model. In Section E.3 the results of both  $\chi$  models are discussed.

### E.1 Standard $\chi$ model

In this section the  $\chi$  model that is modelled using the standard modelling techniques in case of manufacturing systems according to [Hof02]. Due to the used product mix and the models determinism, the structure of the model might not look familiar immediately. The model that is built using the standard modelling technique should behave similar to the max-plus model. For reasons of clarity, a graphical representation of the standard model and its channels can be seen in Figure E.1. In this figure  $G$  represents the

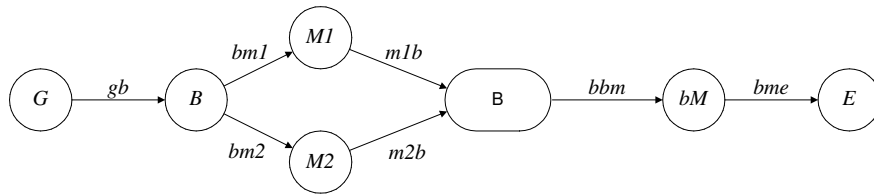


Figure E.1: graphical representation of the standard  $\chi$ -file

generator,  $B_{\text{inf}}$  the infinite buffer,  $M_i$  machine  $i$ ,  $B_{\text{fin}}$  the finite buffer,  $bM$  the batch machine and  $E$  the exit process. Using this file, the situations with inputs as in Table 4.1 (session 1) and Table 4.2 (session 2) can be validated. The time instants at which  $G$  sends products to the  $B_{\text{inf}}$  and the time instants at which the  $E$  can receive products

from  $bM$  are given in the  $xper$ -environment of the model. The vector  $u_1$  and  $u_2$  contain the time instants at which  $P_1$  and  $P_2$  enter the manufacturing system. Vectors  $u_{15}$  and  $u_{16}$  contain the time instant at which  $P_1$  and  $P_2$  can leave the system. The booleans `receiveP1` and `sendP1` etc. are added due to the product mix.

```

from std import *

type prodtypenum = nat          // product type number, eg 11 means first P1
  , Binfstart    = real        // time instant at which infinite buffer starts buffering,
                                // analogously to the case, this is equal to states x1 and x2
  , Mstart       = real        // time instant at which the machines (M1 or M2) start processing,
                                // analogously to the case, this is equal to states x4 and x6
  , Bfinstart    = real        // time instant at which the finite buffer starts buffering,
                                // analogously to the case, this is equal to states x8 and x9
  , BMstart      = real        // time instant at which the batch machine starts working,
                                // analogously to the case, this is equal to state x13
  , Eend         = real        // time instant at which the finished products leave the
                                // manufacturing system, analogously to the case this is equal
                                // to y13 and y14

// in this file a manufacturing system is modelled using the standard way of modelling to
// compare chi with the max-plus-algebra

// the manufacturing system that is modelled in this file behaves deterministic for both the
// process time and product route and contains a certain product mix: P1, P2, P1, etc.

// created by: D.Wetjens
// creation date: 16-05-04

type lot= prodtypenum # Binfstart # Mstart # Bfinstart # BMstart # Eend

proc G(a: !lot, u1: real*, u2: real*) =
| [ sendp1, sendp2: bool, n1, n2: nat
  | n1:= 11; n2:= 21
  ; sendp1:= true; sendp2:= false
  ; *[ sendp1 and len(u1) > 0 and hd(u1) >= time; delta (hd(u1) - time)
    -> a!< n1, 0.0, 0.0, 0.0, 0.0, 0.0 >
    ; u1:= tl(u1)
    ; n1:= n1 + 1
    ; sendp1:= false
    ; sendp2:= true
  ; *[ sendp1 and len(u1) > 0 and hd(u1) < time
    -> a!< n1, 0.0, 0.0, 0.0, 0.0, 0.0 >
    ; u1:= tl(u1)
    ; n1:= n1 + 1
    ; sendp1:= false
    ; sendp2:= true
  | sendp2 and len(u2) > 0 and hd(u2) < time

```

```

        -> a!< n2, 0.0, 0.0, 0.0, 0.0, 0.0 >
        ; u2:= tl(u2)
        ; n2:= n2 + 1
        ; sendp2:= false
        ; sendp1:= true
    ]
| sendp2 and len(u2) > 0 and hd(u2) >= time; delta (hd(u2) - time)
-> a!< n2, 0.0, 0.0, 0.0, 0.0, 0.0 >
; u2:= tl(u2)
; n2:= n2 + 1
; sendp2:= false
; sendp1:= true
; *[ sendp1 and len(u1) > 0 and hd(u1) < time
    -> a!< n1, 0.0, 0.0, 0.0, 0.0, 0.0 >
    ; u1:= tl(u1)
    ; n1:= n1 + 1
    ; sendp1:= false
    ; sendp2:= true
| sendp2 and len(u2) > 0 and hd(u2) < time
-> a!< n2, 0.0, 0.0, 0.0, 0.0, 0.0 >
; u2:= tl(u2)
; n2:= n2 + 1
; sendp2:= false
; sendp1:= true
]
]
]]

proc Binf(a: ?lot, b: !lot, c: !lot) =
| [ x: lot, xs: lot*, sendp1, sendp2, receivep1, receivep2: bool
| xs:= []; sendp1:= true; sendp2:= false; receivep1:= true; receivep2:= false
; *[ receivep1; a?x
    -> x.1:= time
    ; xs:= xs ++ [x]
    ; receivep1:= false
    ; receivep2:= true
| receivep2; a?x
    -> x.1:= time
    ; xs:= xs ++ [x]
    ; receivep2:= false
    ; receivep1:= true
| sendp1 and len(xs) > 0; b!hd(xs)
    -> xs:= tl(xs)
    ; sendp1:= false
    ; sendp2:= true
| sendp2 and len(xs) > 0; c!hd(xs)
    -> xs:= tl(xs)
    ; sendp2:= false
    ; sendp1:= true
]
]]

proc M (a: ?lot, b: !lot, pt:real) =
| [ x: lot

```

```

| * [ true
    -> a?x
      ; x.2:= time
      ; delta pt
      ; b!x
  ]
]|

proc Bfin( a: ?lot, b: ?lot, c: !lot) =
| [ x: lot, xs: lot*, sendp1, sendp2, receivep1, receivep2: bool
  | xs:= []; sendp1:= true; sendp2:= false; receivep1:= true; receivep2:= false
  ; * [ receivep1 and len(xs) < 3; a?x
    -> x.3:= time
      ; xs:= xs ++ [x]
      ; receivep1:= false
      ; receivep2:= true
    | receivep2 and len(xs) < 3; b?x
    -> x.3:= time
      ; xs:= xs ++ [x]
      ; receivep2:= false
      ; receivep1:= true
    | sendp1 and len(xs) > 0; c!hd(xs)
    -> xs:= tl(xs)
      ; sendp1:= false
      ; sendp2:= true
    | sendp2 and len(xs) > 0; c!hd(xs)
    -> xs:= tl(xs)
      ; sendp2:= false
      ; sendp1:= true
  ]
]|

proc bM(a: ?lot, b: !lot, pt: real) =
| [ x, y: lot, xs: lot*, sendp1, sendp2, receivep1, receivep2: bool, tready: real
  | xs:= []; sendp1:= false; sendp2:= false; receivep1:= true; receivep2:= false
  ; * [ len(xs) = 0 and receivep1; a?x
    -> xs:= xs ++ [x]
      ; receivep1:= false
      ; receivep2:= true
    | len(xs) = 1 and receivep2; a?x
    -> xs:= xs ++ [x]
      ; receivep1:= true
      ; receivep2:= false
      ; y:= hd(xs)
      ; y.4:= time
      ; xs:= tl(xs) ++ [y]
      ; y:= hd(xs)
      ; y.4:= time
      ; xs:= tl(xs) ++ [y]
      ; tready:= time + pt
    | len(xs) = 2; delta pt
    -> sendp1:= true
    | sendp1; b!hd(xs)
    -> xs:= tl(xs)
  ]
]|

```



```

        ; sendp1:= false
        ; sendp2:= true
    | sendp2; b!hd(xs)
    -> xs:= tl(xs)
        ; sendp2:= false
    ]
]]

proc E (a:? lot, u1: real*, u2: real*) =
| [x: lot, receivep1, receivep2: bool
  | receivep1:= true; receivep2:= false
  ;*[ receivep1 and len(u1) > 0 and hd(u1) >= time; delta (hd(u1) - time)
    -> a?x
        ; u1:= tl(u1)
        ; x.5:= time
        ; !x, nl(), nl()
        ; receivep1:= false
        ; receivep2:= true
        ; *[ receivep1 and len(u1) > 0 and hd(u2) < time
          -> a?x
              ; u1:= tl(u1)
              ; x.5:= time
              ; !x, nl(), nl()
              ; receivep1:= false
              ; receivep2:= true
          | receivep2 and len(u2) > 0 and hd(u2) < time
            -> a?x
                ; u2:= tl(u2)
                ; x.5:= time
                ; !x, nl(), nl()
                ; receivep2:= false
                ; receivep1:= true
            ]
  | receivep2 and len(u2) > 0 and hd(u2) >= time; delta (hd(u2) - time)
    -> a?x
        ; u2:= tl(u2)
        ; x.5:= time
        ; !x, nl(), nl()
        ; receivep2:= false
        ; receivep1:= true
        ; *[ receivep1 and len(u1) > 0 and hd(u2) < time
          -> a?x
              ; u1:= tl(u1)
              ; x.5:= time
              ; !x, nl(), nl()
              ; receivep1:= false
              ; receivep2:= true
          | receivep2 and len(u2) > 0 and hd(u2) < time
            -> a?x
                ; u2:= tl(u2)
                ; x.5:= time
                ; !x, nl(), nl()
                ; receivep2:= false
                ; receivep1:= true
            ]
  ]
]

```

```

    ]
  ]
]

clus S() =
| [ gb, bm1, bm2, m1b, m2b, bbm, bme: -lot
| G(gb, [0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0],
|      [1.0, 6.0, 11.0, 16.0, 21.0, 26.0, 31.0, 36.0, 41.0])
|| Binf(gb, bm1, bm2)
|| M(bm1, m1b, 1.0)
|| M(bm2, m2b, 3.0)
|| Bfin(m1b, m2b, bbm)
|| bM(bbm, bme, 10.0)
|| E(bme, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
|      [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
]

xper = [ S() ]

```

## E.2 Specified $\chi$ model

In this section a  $\chi$  model is presented that is used to validate the max-plus model in a different way as the model that is presented in Section E.1. The  $\chi$  model in this section is modelled such that its actions and events correspond to the actions of the max-plus model. Compared with the standard  $\chi$  model, the specified model contains synchronization channels that are needed to model the availability information. A graphical representation can be seen in Figure E.2. All processes have to be able to

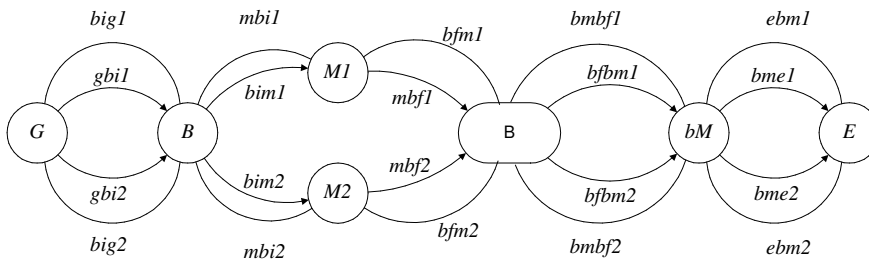


Figure E.2: graphical representation of the specified  $\chi$ -file

receive this availability information at all times. Even if, for example, a machine is processing a product, the machine must be able to receive availability information from the process downstream. Therefore, the systematic way of modelling of processes, such as machines and buffers as presented in [Hof02], can not be used here. The specification of the most simple form of a machine is given here:

```

proc M(a: ?lot, b: !lot) =
  |[ x: lot
    | *[ true -> a?x; delta 3.0; b!x]
  ]|

```

A machine is modelled here, using a repetitive selection statement. Due to the boolean that is always true, the machine can always receive a products if its state is idle. In case of the max-plus model, a process can always receive availability information from the connected process in downstream direction. This means that, for instance, a machine should receive availability information, even if the machine is processing a product. Using the most simple specification, the machine cannot communicate with other processes while processing. Therefore, the most simple specification of a machine is not useful if this model is used as a validation file with respect to the max-plus model. In the  $\chi$  file that is presented in Section E.1, instead of repetitive selection statement, a repetitive selective waiting statement is used to model all processes. Using this statement, a process can always receive availability information. If this information has been received, a certain counter is increased by one. If this counter is not equal to zero, the process can send a finished product to its connected process in downstream direction. As mentioned earlier in this appendix, the availability information as used in the max-plus model is modelled using synchronization channels. The input sequence and the availability information of the output is, similar to the max-plus model given in vectors. These vectors can be seen in the *xper*-environment. Due to the product mix  $P_1, P_2, P_1$  etc. the booleans such as receiveP1 and sendP1 are added.

```

from std import *

type prodtypenum = nat          // product type number, eg 11 means first P1
  , Binfstart    = real          // time instant at which infinite buffer starts buffering,
  // analogously to the case, this is equal to states x1 and x2
  , Mstart       = real          // time instant at which the machines (M1 or M2) start processing,
  // analogously to the case, this is equal to states x4 and x6
  , Bfstart      = real          // time instant at which the finite buffer starts buffering,
  // analogously to the case, this is equal to states x8 and x9
  , BMstart      = real          // time instant at which the batch machine starts working,
  // analogously to the case, this is equal to state x13
  , Eend         = real          // time instant at which the finished products leave the
  // manufacturing system, analogously to the case this is equal to
  // y13 and y14

// in this file a manufacturing system is modelled conform the actions that go with the
// max-plus model

// the manufacturing system that is modelled in this file behaves deterministic for both the
// process time and product route and contains a certain product mix: P1, P2, P1, etc.

// if counter v is higher than zero, products can be sent downstream by the process
// if counter s is higher than zero, products can be received by the process

```

```

// if counter p is higher than zero, due to the way of modelling, a product can be
// sent by the generator
// or received by the exit process

// created by: D.Wetjens
// creation date: 16-05-04

type lot= prodtypenum # Binfstart # Mstart # Bfinstart # BMstart # Eend

proc G(a: !lot, b:!lot, c: ?void, d: ?void, u1: real*, u2: real*) =
| [ sendp1, sendp2: bool, n1, n2, v1, v2, p1, p2: nat
| n1:= 11; n2:= 21; v1:= 0; v2:= 0; p1:= 0; p2:= 0
; sendp1:= true; sendp2:= false
; * [ sendp1 and len(u1) > 0 and hd(u1) >= time and p1 = 0; delta (hd(u1) - time)
-> p1:= p1 + 1
; u1:= tl(u1)
| sendp1 and len(u1) > 0 and hd(u1) < time and p1 = 0 and v1 > 0
; a!< n1, 0.0, 0.0, 0.0, 0.0, 0.0 >
-> u1:= tl(u1)
; n1:= n1 + 1
; v1:= v1 - 1
; sendp1:= false
; sendp2:= true
| sendp2 and len(u2) > 0 and hd(u2) >= time and p2 = 0; delta (hd(u2) - time)
-> p2:= p2 + 1
; u2:= tl(u2)
| sendp2 and len(u2) > 0 and hd(u2) < time and p2 = 0 and v2 > 0
; b!< n2, 0.0, 0.0, 0.0, 0.0, 0.0 >
-> u2:= tl(u2)
; n2:= n2 + 1
; v2:= v2 - 1
; sendp2:= false
; sendp1:= true
| sendp1 and p1 > 0 and v1 > 0; a!< n1, 0.0, 0.0, 0.0, 0.0, 0.0 >
-> n1:= n1 + 1
; p1:= p1 - 1
; v1:= v1 - 1
; sendp1:= false
; sendp2:= true
| sendp2 and p2 > 0 and v2 > 0; b!< n2, 0.0, 0.0, 0.0, 0.0, 0.0 >
-> n2:= n2 + 1
; p2:= p2 - 1
; v2:= v2 - 1
; sendp2:= false
; sendp1:= true
| true; c?
-> v1:= v1 + 1
| true; d?
-> v2:= v2 + 1
]
]

```

```

proc Binf (a: ?lot, b: ?lot, c: !lot, d: !lot
           ,e: ?void, f: ?void, g: !void, h: !void, pnmb1: nat, pnmb2: nat) =
| [x: lot, sendp1, sendp2, receivep1, receivep2: bool, xs, ys: lot*, p1, p2, v1, v2: nat
  | sendp1:= true; sendp2:= false; receivep1:= true; receivep2:= false
  ; xs:= []; ys:= []; p1:= 0; p2:= 0; v1:= 0; v2:= 0
  ; *[ pnmb1 > 0; g!
    -> pnmb1:= pnmb1 - 1
    | pnmb2 > 0; h!
    -> pnmb2:= pnmb2 - 1
    | receivep1; a?x
    -> x.1:= time
      ; xs:= xs ++ [x]
      ; receivep1:= false
      ; receivep2:= true
      ; p1:= p1 + 1
    | receivep2; b?x
    -> x.1:= time
      ; ys:= ys ++ [x]
      ; receivep2:= false
      ; receivep1:= true
      ; p2:= p2 + 1
    | sendp1 and p1 > 0 and v1 > 0; c!hd(xs)
    -> xs:= tl(xs)
      ; p1:= p1 - 1
      ; v1:= v1 - 1
      ; sendp1:= false
      ; sendp2:= true
    | sendp2 and p2 > 0 and v2 > 0; d!hd(ys)
    -> ys:= tl(ys)
      ; p2:= p2 - 1
      ; v2:= v2 - 1
      ; sendp2:= false
      ; sendp1:= true
    | true; e?
    -> v1:= v1 + 1
    | true; f?
    -> v2:= v2 + 1
  ]
]

```

```

proc M(a: ?lot, b: !lot, c: ?void, d: !void, pt: real) =
| [ x: lot, xs, ys: lot*, v, s: nat, tready: real
  | xs:= []; ys:= []; v:= 0; s:= 0; d!
  ; *[ len(xs) = 0 and len(ys) = 0 and s = 0; a?x
    -> x.2:= time
      ; xs:= xs ++ [x]
      ; tready:= time + pt
    | len(xs) > 0 and len(ys) = 0; delta (tready - time)
    -> ys:= ys ++ [hd(xs)]
      ; xs:= tl(xs)
    | len(xs) = 0 and len(ys) > 0 and v > 0; b!hd(ys)
    -> ys:= tl(ys)
      ; s:= s + 1
      ; v:= v - 1
  ]

```

```

| true; c?
-> v:= v + 1
| s > 0; d!
-> s:= s - 1
]
]

```

```

proc Bfin(a: ?lot, b: ?lot, c: !lot, d: !lot, e: ?void, f: ?void, g: !void, h: !void) =
| [ xs, ys: lot*, v1, v2, s1, s2: nat, x: lot, receivep1, receivep2, sendp1, sendp2: bool
| xs:= []; ys:= []; v1:= 0; v2:=0; s1:= 1; s2:= 0; g!; h!
; receivep1:= true; receivep2:= false; sendp1:= true; sendp2:= false
; *[ receivep1 and len(xs) < 3 and s1 = 0; a?x
-> x.3:= time
; xs:= xs ++ [x]
; receivep1:= false
; receivep2:= true
| receivep2 and len(xs) < 3 and s2 = 0; b?x
-> x.3:= time
; xs:= xs ++ [x]
; receivep2:= false
; receivep1:= true
| sendp1 and len(xs) > 0 and v1 > 0; c!hd(xs)
-> xs:= tl(xs)
; s1:= s1 + 1
; v1:= v1 - 1
; sendp1:= false
; sendp2:= true
| sendp2 and len(xs) > 0 and v2 > 0; d!hd(xs)
-> xs:= tl(xs)
; s2:= s2 + 1
; v2:= v2 - 1
; sendp2:= false
; sendp1:= true
| true; e?
-> v1:= v1 + 1
| true; f?
-> v2:= v2 + 1
| s1 > 0; g!
-> s1:= s1 -1
| s2 > 0; h!
-> s2:= s2 -1
]
]

```

```

proc bM(a: ?lot, b: ?lot, c: !lot, d: !lot, e: ?void, f: ?void, g: !void, h: !void, pt: real) =
| [ xs, ys: lot*, tready: real, v1, v2, s1, s2: nat, x, y: lot, sendp1, sendp2, receivep1, receivep2: bool
| xs:= []; ys:= []; v1:= 0; v2:= 0; s1:= 0; s2:= 0
; receivep1:= true; receivep2:= false; sendp1:= true; sendp2:= false; g!; h!
; *[ receivep1 and len(xs) = 0 and len(ys) = 0 and s1 = 0 and s2 = 0; a?x
-> xs:= xs ++ [x]
; receivep1:= false
; receivep2:= true
| receivep2 and len(xs) = 1 and len(ys) = 0 and s1 = 0 and s2 = 0; b?x
-> xs:= xs ++ [x]

```

```

        ; receivep2:= false
        ; receivep1:= true
        ; tready:= time + pt
        ; y:= hd(xs)
        ; y.4:= time
        ; xs:=tl(xs) ++ [y]
        ; y:= hd(xs)
        ; y.4:= time
        ; xs:=tl(xs) ++ [y]
| len(xs) = 2 and len(ys) = 0; delta(tready - time)
-> ys:= ys ++ [hd(xs)]
    ; xs:= tl(xs)
    ; ys:= ys ++ [hd(xs)]
    ; xs:= tl(xs)
| true; e?
-> v1:= v1 + 1
| true; f?
-> v2:= v2 + 2
| sendp1 and len(ys) > 0 and v1 > 0 ; c!hd(ys)
-> ys:= tl(ys)
    ; s1:= s1 + 1
    ; v1:= v1 - 1
    ; sendp1:= false
    ; sendp2:= true
| sendp2 and len(ys) > 0 and v2 > 0 ; d!hd(ys)
-> ys:= tl(ys)
    ; s2:= s2 + 1
    ; v2:= v2 - 1
    ; sendp2:= false
    ; sendp1:= true
| s1 = 1 and s2 = 1; g!
-> s1:= s1 - 1
| s1 = 0 and s2 = 1; h!
-> s2:= s2 - 1
]
]]

proc E(a: ?lot, b: ?lot, c: !void, d: !void, u1: real*, u2: real*) =
| [ x: lot, receivep1, receivep2: bool, s1, s2, p1, p2: nat
| receivep1:= true; receivep2:= false; s1:= 0; s2:= 0; p1:= 0; p2:= 0
; *[ receivep1 and len(u1) > 0 and p1 = 0 and hd(u1) >= time; delta (hd(u1) - time)
-> p1:= p1 + 1
    ; u1:= tl(u1)
| receivep1 and p1 > 0 and s1 = 0; c!
-> s1:= s1 + 1
| receivep1 and s1 > 0; a?x
-> x.5:= time
    ; p1:= p1 - 1
    ; s1:= s1 - 1
    ; receivep1:= false
    ; receivep2:= true
    ; !x, nl(), nl()
    ; [ len(u2) > 0 and hd(u2) < time
-> p2:= p2 + 1

```

```

        ; u2:= tl(u2)
        | len(u2) > 0 and hd(u2) >= time
        -> skip
        | len(u2) = 0
        -> skip
    ]
| receivep2 and len(u2) > 0 and p2 = 0 and hd(u2) >= time; delta (hd(u2) - time)
-> p2:= p2 + 1
; u2:= tl(u2)
| receivep2 and p2 > 0 and s2 = 0; d!
-> s2:= s2 + 1
| receivep2 and s2 > 0; b?x
-> x.5:= time
; p2:= p2 - 1
; s2:= s2 - 1
; receivep2:= false
; receivep1:= true
; !x, nl(), nl()
; [ len(u1) > 0 and hd(u1) < time
-> p1:= p1 + 1
; u1:= tl(u1)
| len(u1) > 0 and hd(u1) >= time
-> skip
| len(u1) = 0
-> skip
]
]
]
]

clus S() =
|[ gbi1, gbi2, bim1, bim2, mbf1, mbf2, bfbm1, bfbm2, bme1, bme2: -lot
, big1, big2, mbi1, mbi2, bfm1, bfm2, bmbf1, bmbf2, ebm1, ebm2: -void
| G(gbi1, gbi2, big1, big2, [ 0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0 ]
, [ 1.0, 6.0, 11.0, 16.0, 21.0, 26.0, 31.0, 36.0, 41.0 ])
| Binf(gbi1, gbi2, bim1, bim2, mbi1, mbi2, big1, big2, 9, 9)
| M(bim1, mbf1, bfm1, mbi1, 1.0)
| M(bim2, mbf2, bfm2, mbi2, 3.0)
| Bfin(mbf1, mbf2, bfbm1, bfbm2, bmbf1, bmbf2, bfm1, bfm2)
| bM(bfbm1, bfbm2, bme1, bme2, ebm1, ebm2, bmbf1, bmbf2, 10.0)
| E(bme1, bme2, ebm1, ebm2, [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ]
, [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ])
]
]

xper = |[S()|

```



### E.3 Simulation results of both $\chi$ files

The  $\chi$  files presented and discussed in Sections E.1 and E.2 return data if simulations are done. This data corresponds to the start and end times of events that take place. Using this data, lot-time diagrams can be made. In Section 4.7 two sessions of input sequences and availability information are presented to validate the max-plus model, see Tables 4.1 and 4.2. The returned data of both the standard  $\chi$  file and the specified  $\chi$  file is equal. The data that is returned by both models using the specifications of session 1 is:

< 11 0 0 1 4 14 >	< 16 25 25 34 54 64 >
< 21 1 1 4 4 14 >	< 26 26 34 44 54 64 >
< 12 5 5 6 14 24 >	< 17 30 34 44 64 74 >
< 22 6 6 9 14 24 >	< 27 31 44 54 64 74 >
< 13 10 10 11 24 34 >	< 18 35 44 54 74 84 >
< 23 11 11 14 24 34 >	< 28 36 54 64 74 84 >
< 14 15 15 16 34 44 >	< 19 40 54 64 84 94 >
< 24 16 16 24 34 44 >	< 29 41 64 74 84 94 >
< 15 20 20 24 44 54 >	
< 25 21 24 34 44 54 >	

The data returns 18 tuples that represent the 18 lots that are processed by the manufacturing system. These products contain information about its type, product number and time instants at which it is fed to the system, processed and buffered. Using these time instants, a lot-time can be made. This can be seen in Figure E.3. This lot-time diagram is equal to the lot-time diagram that is obtained by calculations by hand. Therefore, based on session 1 the max-plus model is validated successfully.

The second session that is used to validate the max-plus model contains 'extreme' conditions. These conditions are already mentioned in Section 4.7. Implementation of these input sequence and availability information, result in the following similar returned data for both  $\chi$  files:

< 11 0 0 1 3 13 >	< 16 25 25 33 53 63 >
< 21 0 0 3 3 13 >	< 26 26 33 43 53 63 >

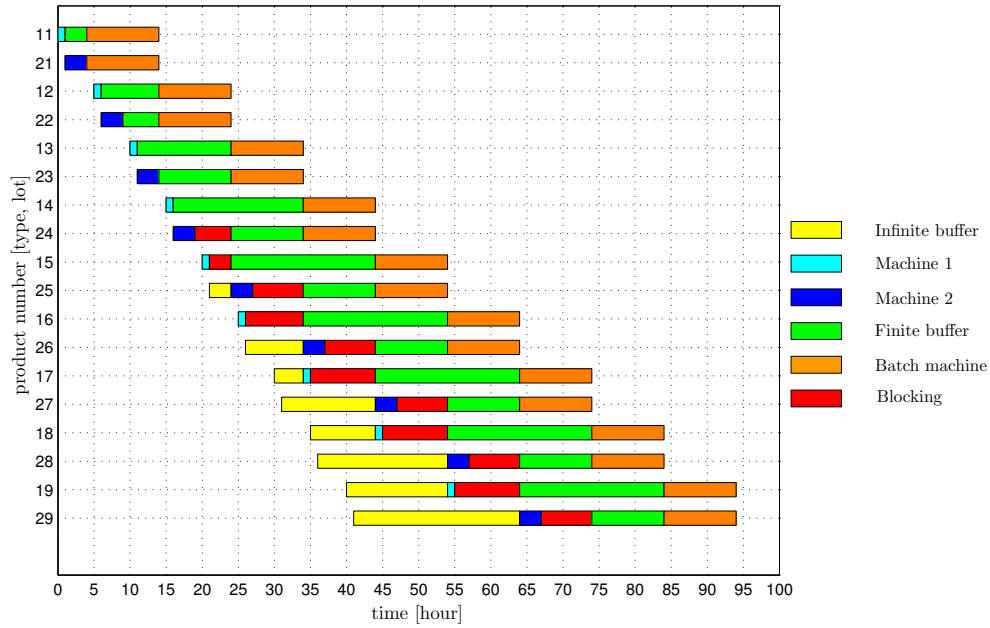


Figure E.3: Lot-time diagram, session 1

< 12 5 5 6 13 23 >	< 17 26 33 43 63 73 >
< 22 6 6 9 13 23 >	< 27 31 43 53 63 73 >
< 13 6 6 9 23 33 >	< 18 35 43 53 73 83 >
< 23 11 11 14 23 33 >	< 28 36 53 63 73 83 >
< 14 15 15 16 33 43 >	< 19 40 53 63 83 100 >
< 24 16 16 23 33 43 >	< 29 41 63 73 83 100 >
< 15 20 20 23 43 53 >	
< 25 20 23 33 43 53 >	

The returned data by the two  $\chi$  files is treated and used to illustrate the outcome in a lot-time diagram, see Figure E.4. Both lot-time diagrams as can be seen in Figures E.3 and E.4 are equal to the lot-time diagrams as obtained by calculations by hand and by simulating the max-plus model, see Figures 4.8 and 4.9. Therefore, the max-plus model is validated successfully.

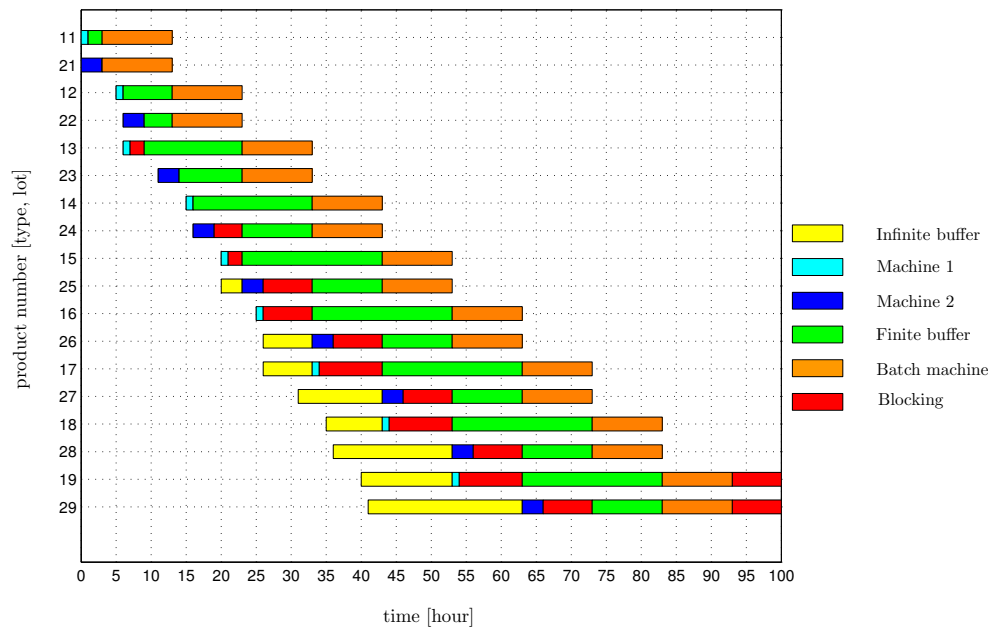


Figure E.4: Lot-time diagram, session 2



## Appendix F

# MPC implementation

Many methods exist to solve optimization problems. In this thesis, linear programming (LP) is used to find the optimal input sequence to feed the raw material to the manufacturing system with respect to a given reference signal at which products have to leave the system. Using LP, the objective function has to be written as a linear function of the decision variables (here called  $x_j$ ,  $j = 1, 2, \dots, n$ ). These decision variables (in problem (5.12) this is vector  $u(k)$ ) are the variables whose values are to be decided in some optimal fashion [Van01]. The objective function can be written in the following form:

$$J = c_1x_1 + c_2x_2 + \dots + c_nx_n \quad (\text{F.1})$$

with: coefficients  $c_j$ ,  $j = 1, 2, \dots, n$ .

In addition to the objective function, also the equality and/or in-equality constraints have to be written in linear combinations of the decision variables:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \left\{ \begin{array}{l} \leq \\ = \end{array} \right\} b \quad (\text{F.2})$$

with: coefficients  $a_j$ ,  $j = 1, 2, \dots, n$ .

The original optimization problem (5.12) contains all sort of non-linearities. The objective function (5.12a) contains the maximization operator that is non-linear. Constraint (5.12b) contains the operators  $\oplus$  and  $\otimes$ . These operations are in general non-linear due to the similarity with maximization. Using the LP method to solve this optimization problem, the mentioned non-linear operations in problem (5.12) have to be transformed to linear operations. This means that the structure of optimization problem will change, but the dynamics of the problem will remain the same. The transformation of the original non-linear problem into a linear problem is explained using an example. The size of

the problem depends on the number of in- and outputs  $l$  and the size of the prediction horizon  $N_p$ . The number of inputs of the manufacturing line as described in Chapter 4 is four. Two inputs are used for availability information ( $u_{15}(k)$  and  $u_{16}(k)$ ). Here, only the inputs that receive products are considered, which means that this number is two ( $u_1(k)$  and  $u_2(k)$ ). For the number of outputs of the system, a similar approach is used. This all can be seen in Figure F.1. The  $S$  stands for the entire manufacturing system. In the notation of the original problem, the outputs are number from  $i$  to  $l$ . Therefore, outputs  $y_{15}(k)$  and  $y_{16}(k)$  are in this LP formulation numbered from 1 to 2.

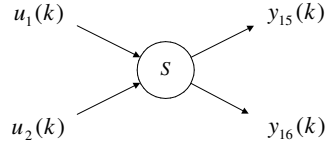


Figure F.1: Considered in- and outputs

In order to keep the example small and surveyable,  $N_p$  and  $N_c$  are chosen to be 2, respectively 1. First, the objective function (5.12a) and secondly the max-plus dynamics and the constraints (5.12b) to (5.12c) are transformed in some linear function of the decision variables.

The non-linear term in the objective function (5.12a) is the maximization operator in  $J_{\text{out},1}$ . If, for example, the maximum of two scalars  $x_1$  and  $x_2$  have to be determined ( $\max(x_1, x_2)$ ), a dummy variable  $x_3$  has to be introduced. Now, the maximization can be replaced by the following:

$$\begin{aligned} \min \quad & x_3 \\ & x_3 \geq x_1 \\ & x_3 \geq x_2. \end{aligned}$$

In words, this means that the value of  $x_3$  has to be equal to or larger than  $x_1$  and  $x_2$ . If  $x_1 \geq x_2$ , variable  $x_3$  will become equal to  $x_1$  (and larger than  $x_2$ ) due to its minimization. Using this principle, the non-linear max term can be linearized. Using MPC,  $N_p$  future inputs and its outputs are determined. As mentioned earlier,  $N_p$  is 2, which means that if  $k = 0$ , the  $u(k)$  and  $\tilde{y}(k)$  become:

$$\begin{aligned} \bar{u}(0) &= \begin{pmatrix} u_1(1) & u_2(1) & u_1(2) & u_2(2) \end{pmatrix}^T \\ \tilde{y}(0) &= \begin{pmatrix} \hat{y}_1(1) & \hat{y}_2(1) & \hat{y}_1(2) & \hat{y}_2(2) \end{pmatrix}^T. \end{aligned}$$

Here,  $u_1(1)$  stands for the time instant at which product  $P_1$  is fed to input 1 of the system for the first time. The same is valid for the predicted output  $\tilde{y}(k)$ . The reference signal

$\bar{r}(k)$  (in case  $k = 0$ ) becomes:

$$\bar{r}(1) = \begin{pmatrix} r_1(1) & r_2(1) & r_1(2) & r_2(2) \end{pmatrix}^T.$$

Here,  $r_1(1)$  stands for the time instant at which product  $P_1$  has to leave the system for the second time. Now, the maximization term in the objective function will be transformed in a linear notation using the above explained principle. For  $i = 1$  and  $j = 1$ ,  $J_{\text{out},1}$  becomes:

$$\max(\hat{y}_1(1) - r_1(1), r_1(1) - \hat{y}_1(1)).$$

This can be transformed in:

$$\begin{aligned} \min \quad & z_1(1) \\ z_1(1) \geq & \hat{y}_1(1) - r_1(1) \\ z_1(1) \geq & r_1(1) - \hat{y}_1(1). \end{aligned}$$

From mathematical point of view, there is a preferred presentation of constraints in general. This presentation is to pose (in)equalities as less-than or equal to ( $\leq$ ) constraints [Van01]. These constraints have to be written in the form (F.2). The decision variables are  $z$  and  $y$  and the parameters are  $r$ . Therefore, these constraints become:

$$\begin{aligned} \min \quad & z_1(1) \\ \hat{y}_1(1) - z_1(1) & \leq r_1(1) \\ -\hat{y}_1(1) - z_1(1) & \leq -r_1(1). \end{aligned}$$

This approach can be done for all combinations of  $i = 1, \dots, l$  and  $j = 1, \dots, N_p$ . In case of  $N_p = 2$  and  $N_c = 1$ , this results in eight inequality constraints (two equations per dummy variable  $z$ ). These equations can be written in the following matrix formulation (with  $A$  of the size  $4N_p \times 6N_p$  and  $b$  of the size  $4N_p \times 1$ ):

$$A\bar{x} \leq b \tag{F.3}$$

with:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} r_1(1) \\ r_2(1) \\ r_1(2) \\ r_2(2) \\ -r_1(1) \\ -r_2(1) \\ -r_1(2) \\ -r_2(2) \end{pmatrix}$$

where:

$$\bar{x} = ( u_1(1) \quad u_2(1) \quad u_1(2) \quad u_2(2) \quad \hat{y}_1(1) \quad \hat{y}_2(1) \quad \hat{y}_1(2) \quad \hat{y}_2(2) \quad z_1(1) \quad z_2(1) \quad z_1(2) \quad z_2(2) )^T.$$

Note that transforming the original non-convex objective function into a LP formulation results in an increase of the feasible area. Due to this relaxation the objective function becomes convex.

Part  $J_{in,1}$  of the objective function is the sum of all inputs over the prediction horizon. Summation of these inputs in a linear function of decision variables can be done by taking the relevant coefficients in (F.1) equal to one. Due to the weighting parameter  $\lambda$  these coefficients become equal to  $-\lambda$  (see (5.13)).

The next step is to transform the max-plus dynamics (5.12b) into a useful LP formulation. In case of  $N_p = 2$ , matrix  $H$  becomes of size  $4 \times 4$ . Due to its dynamics, the right upper elements of  $H$  are equal to  $\varepsilon$ . These right upper elements of  $H$  are  $H_{13}$ ,  $H_{14}$ ,  $H_{23}$ , and  $H_{24}$ . If the size of the prediction horizon differs, the size of matrix  $H$  differs as well and becomes  $2N_p \times 2N_p$ . The structure of matrix  $H$  can be seen in Figure F.2. In this figure, the structure with respect to the elements equal to  $\varepsilon$  (indicated in grey) can be seen.

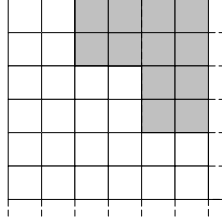


Figure F.2: Structure of matrix  $H$

Similar to the  $H$  matrix, the  $g$  matrix is dependent on  $N_p$ . The size of this  $g$  matrix is  $2N_p \times 15$ , due to the  $15 \times 1$  the state vector. In case  $N_p = 2$  and  $N_c = 1$ ,  $g$  becomes of the size  $4 \times 15$ . Now, the the first equation,  $\hat{y}_1(1)$ , using matrices  $H$  and  $g$ , input  $u$  and initial state vector  $x_0$  results in:

$$\begin{aligned} \hat{y}_1(1) = & H_{11} \otimes u_1(1) \oplus H_{12} \otimes u_2(1) \oplus H_{13} \otimes u_1(2) \oplus H_{14} \otimes u_2(2) \oplus \\ & g_{11} \otimes x_{0,11} \oplus g_{12} \otimes x_{0,21} \oplus \dots \oplus g_{115} \otimes x_{0,151}. \end{aligned} \quad (F.4)$$

As long as all states in the initial state vector  $x_0$  are equal to  $\varepsilon$ ,  $g \otimes x_0$  becomes a vector that contains only  $\varepsilon$ 's. Therefore, this element (which is equal to  $-\infty$ ) can be ignored in a max statement. The same is valid for the elements in the  $H$  matrix that equal  $\varepsilon$ . In Chapter 5 the receding horizon has been explained. Only the first elements



of the optimal input are implemented, whereas the procedure is repeated. This means that the  $x_0$  vector needs to be updated after each step and does only contain  $\varepsilon$ 's at the beginning of the procedure. Therefore, the equations to predict the outputs in which the  $H_{ij}$  and/or  $g_{ij} \otimes x_{0,i1}$  equals  $\varepsilon$ , are ignored. This is done by replacing all elements in row  $i$  in the  $A$  and  $b$  matrix by zero.

Using this approach and the replacements of  $\otimes$  by  $+$  and  $\oplus$  by  $\max$  (F.4) becomes:

$$\begin{aligned} \hat{y}_1(1) = \max(&H_{11} + u_1(1), H_{12} + u_2(1), H_{13} + u_1(2), H_{14} + u_2(2), \\ &g_{11} + x_{0,11}, g_{12} + x_{0,21}, \dots, g_{115} + x_{0,151}). \end{aligned} \quad (\text{F.5})$$

Transforming (F.5) into a LP formulation gives:

$$\begin{aligned} \min \quad & y_1(1) \\ \hat{y}_1(1) \quad & \geq H_{11} + u_1(1) \\ \hat{y}_1(1) \quad & \geq H_{12} + u_2(1) \\ \hat{y}_1(1) \quad & \geq H_{13} + u_1(2) \\ \hat{y}_1(1) \quad & \geq H_{14} + u_2(2) \\ \hat{y}_1(1) \quad & \geq g_{11} + x_{0,11} \\ \hat{y}_1(1) \quad & \geq g_{12} + x_{0,21} \\ & \vdots \\ \hat{y}_1(1) \quad & \geq g_{115} + x_{0,151}. \end{aligned}$$

Now, these equation have to be written in the preferred presentation:

$$\begin{aligned} \min \quad & y_1(1) \\ u_1(1) - \hat{y}_1(1) \quad & \leq -H_{11} \\ u_2(1) - \hat{y}_1(1) \quad & \leq -H_{12} \\ u_1(2) - \hat{y}_1(1) \quad & \leq -H_{13} \\ u_2(2) - \hat{y}_1(1) \quad & \leq -H_{14} \\ -\hat{y}_1(1) \quad & \leq -g_{11} - x_{0,11} \\ -\hat{y}_1(1) \quad & \leq -g_{12} - x_{0,21} \\ & \vdots \\ -\hat{y}_1(1) \quad & \leq -g_{115} - x_{0,151}. \end{aligned}$$

The same can be done for  $y_2(1)$ ,  $y_1(2)$  and  $y_2(2)$ . Each output results in 19 equations. Therefore, this LP transformation results in  $19 \times 2N_p = 38N_p$  equations. This can be subdivided in two inequality systems. The first takes care of the equations that contain elements of the  $H$  matrix. In case of  $N_p$  the  $A$  matrix becomes  $8N_p \times 6N_p$ . The other inequality system contains the constraints that contain elements of  $g$  and  $x_0$ . This  $A$  matrix becomes  $30N_p \times 6N_p$ . In case of the first inequality system of the form

$$A\bar{x} \leq b. \quad (\text{F.6})$$

the  $A$  and  $b$  matrices becomes:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} -H_{11} \\ -H_{12} \\ -H_{13} \\ -H_{14} \\ -H_{21} \\ -H_{22} \\ -H_{23} \\ -H_{24} \\ -H_{31} \\ -H_{32} \\ -H_{33} \\ -H_{34} \\ -H_{41} \\ -H_{42} \\ -H_{43} \\ -H_{44} \end{pmatrix}. \quad (\text{F.7})$$

The size of the  $A$  and  $b$  matrices of the second inequality system is too large to present here ( $60 \times 12$ ). Therefore, only the elements that correspond with  $y_1(0)$  are shown:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}, \quad b = \begin{pmatrix} -g_{11} - x_{0,11} \\ -g_{12} - x_{0,21} \\ -g_{13} - x_{0,31} \\ -g_{14} - x_{0,41} \\ -g_{15} - x_{0,51} \\ -g_{16} - x_{0,61} \\ -g_{17} - x_{0,71} \\ -g_{18} - x_{0,81} \\ -g_{19} - x_{0,91} \\ -g_{110} - x_{0,101} \\ -g_{111} - x_{0,111} \\ -g_{112} - x_{0,121} \\ -g_{113} - x_{0,131} \\ -g_{114} - x_{0,141} \\ -g_{115} - x_{0,151} \\ \vdots \end{pmatrix}. \quad (\text{F.8})$$

In both the inequality systems,  $x$  becomes:

$$\bar{x} = \begin{pmatrix} u_1(1) & u_2(1) & u_1(2) & u_2(2) & \hat{y}_1(1) & \hat{y}_2(1) & \hat{y}_1(2) & \hat{y}_2(2) & z_1(1) & z_2(1) & z_1(2) & z_2(2) \end{pmatrix}^T.$$

Merging both  $A$  matrices and  $b$  vectors, results in an  $A$  matrix of size  $38N_p \times 6N_p$  and a  $b$  matrix of the size  $38N_p \times 1$ .

Now, the objective function (5.12a) and the max-plus dynamics (5.12b) of the optimization problem has been converted in a LP problem. Constraints (5.12c) and (5.12d) are left. These constraints prevent strong variation in the input rate. Over the control and prediction horizon, the input rate is bounded with a lower- and upper bound. The input rate remains constant for the last  $N_p - N_c$  samples. Due to this condition, the input rate is automatically bounded between its boundaries. Therefore, in (5.12c) this boundary is only valid for  $j = 1, \dots, N_p - 1$ . A prediction horizon  $N_p = 2$  and a control horizon of  $N_c = 1$  results in the following:

$$u_1(1) - u_1(0) \geq \text{lb} \quad (\text{F.9})$$

$$u_1(1) - u_1(0) \leq \text{ub} \quad (\text{F.10})$$

$$u_1(2) - u_1(1) \geq \text{lb} \quad (\text{F.11})$$

$$u_1(2) - u_1(1) \leq \text{ub}. \quad (\text{F.12})$$

Rewriting (F.9) to (F.12) in the preferred form results in:

$$-u_1(1) \leq -u_1(0) - \text{lb} \quad (\text{F.13})$$

$$u_1(1) \leq u_1(0) + \text{ub} \quad (\text{F.14})$$

$$-u_1(2) + u_1(1) \leq -\text{lb} \quad (\text{F.15})$$

$$u_1(2) - u_1(1) \leq \text{ub}. \quad (\text{F.16})$$

Using the same approach for the second output ( $u_2$ ), the general matrix formulation becomes:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \bar{x} \leq \begin{pmatrix} \text{ub} + u_1(0) \\ \text{ub} + u_2(0) \\ \text{ub} \\ \text{ub} \\ -\text{lb} - u_1(0) \\ -\text{lb} - u_2(0) \\ -\text{lb} \\ -\text{lb} \end{pmatrix} \quad (\text{F.17})$$

where:

$$\bar{x} = ( u_1(1) \ u_2(1) \ u_1(2) \ u_2(2) \ \hat{y}_1(1) \ \hat{y}_2(1) \ \hat{y}_1(2) \ \hat{y}_2(2) \ z_1(1) \ z_2(1) \ z_1(2) \ z_2(2) )^T.$$

For the last  $N_p - N_c$  samples, the input rate remains constant. This can be formulated as follows:

$$u_1(2) - u_1(1) - u_1(1) + u_1(0) = 0 \quad (\text{F.18})$$

$$u_2(2) - u_2(1) - u_2(1) + u_2(0) = 0. \quad (\text{F.19})$$

Now, (F.18) and (F.19) can be written in a matrix formulation ( $A_{\text{eq}}\bar{x} = b_{\text{eq}}$ ) with:

$$A_{\text{eq}} = \begin{pmatrix} -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad b_{\text{eq}} = \begin{pmatrix} -u_1(0) \\ -u_2(0) \end{pmatrix}$$

where:

$$\bar{x} = ( u_1(1) \ u_2(1) \ u_1(2) \ u_2(2) \ \hat{y}_1(1) \ \hat{y}_2(1) \ \hat{y}_1(2) \ \hat{y}_2(2) \ z_1(1) \ z_2(1) \ z_1(2) \ z_2(2) )^T.$$

The size of the  $A_{\text{eq}}$  and  $b_{\text{eq}}$  become respectively  $2N_p(N_p - N_c) \times 6N_p$  and  $2N_p(N_p - N_c) \times 1$

The non-linear maximization terms in the objective function and the max-plus dynamics are replaced by in-equality constraints. These transformation result in an increase of the decision variables (due to dummy variables  $z$ ). In the original problem, the decision variables were the input variables. In the LP problem, not only the input variables are the decision variables. The output variables  $y$  and the dummy variables  $z$  are added to the original decision variables. In case of  $N_p = 2$  and  $N_c = 1$ , there are four input variables, four output variables, four dummy variables. The entire decision variable vector  $\bar{x}$  becomes  $6N_p \times 1 = 12 \times 1$ :

$$\bar{x} = ( u_1(1) \ u_2(1) \ u_1(2) \ u_2(2) \ \hat{y}_1(1) \ \hat{y}_2(1) \ \hat{y}_1(2) \ \hat{y}_2(2) \ z_1(1) \ z_2(1) \ z_1(2) \ z_2(2) )^T.$$

In this report the LP solver `linprog` is used to obtain the optimal input sequence with respect with a desired reference signal. This standard Matlab solver needs the (in)equality constraint matrices if the problem is bounded. These constraints are written in the standard matrix formulation:

$$\begin{aligned} A\bar{x} &\leq b \\ A_{\text{eq}}\bar{x} &= b_{\text{eq}}. \end{aligned}$$

Merging all  $A$  matrices into one large  $A$  matrix results in a matrix of the size  $46N_p \times 6N_p$ . The same can be done for  $b$ . This results in a  $46N_p \times 1$  vector.  $A_{\text{eq}}$  and  $b_{\text{eq}}$  become of the respectively size of  $2N_p(N_p - N_c) \times 6N_p$  and  $2N_p(N_p - N_c) \times 1$ .

The original objective function (5.12a) contains a weighting parameter  $\lambda$ . Weighting parameters make trade offs between different part of the objective function. A more detailed explanation of weighting parameters is given in Section 5.3. Therefore, the (LP) objective function becomes:

$$J = \lambda \sum_{i=1}^2 \sum_{j=1}^{N_p} u_i(k+j) + \sum_{i=1}^2 \sum_{j=1}^{N_p} y_i(k+j) + \sum_{i=1}^2 \sum_{j=1}^{N_p} z_i(k+j). \quad (\text{F.20})$$

This function is linear and can be written follows (F.1):

$$J = c^T \bar{x} \quad (\text{F.21})$$

with (in case of  $N_p = 2$  and  $N_c = 1$ ):

$$c = \begin{pmatrix} -\lambda & -\lambda & -\lambda & -\lambda & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$



## Appendix G

# Matlab file of MPC implementation

In this Appendix, the MPC implementation in Matlab is presented. In Section G.1 the main file is given. In this main file, all inputs and settings can be changed. The main file uses two other files to calculate the optimal input sequence with respect to the objective function. These can be seen in Section G.2.

### G.1 mainfile.m

```
clear all;

% process times machines
d1=1; % process time machine 1
d2=3; % process time machine 2
d3=10; % process time batch machine

% definition of epsilon (e)
e=-inf;

% system matrices A, B, C, and D
% the last inputs, u15 and u16, are for all k equal to epsilon. Therefore,
% the last 2 columns of the original matrices B and D can be deleted. The same is done
% for the last 2 outputs, y3 and y4. Therefore, the last two rows of the original C
% and D matrix are deleted.

Asys=[ e 0 e e e e e e e e e e e;
        e 0 e e e e e e e e e e e;
        e 0 0 d1 0 d2 0 e e e e e e e;
        e 0 0 d1 0 e e e e e e e e e;
        e e e e e e e e e 0 e e e;
        e 0 0 d1 0 d2 0 e e e e e e e;
```

```

e e e e e e e e 0 e e e e e;
e d1 d1 2*d1 d1 e e e 0 e e 0 e e e;
e mp_sum(d1,d2) mp_sum(d1,d2) mp_sum(2*d1,mp_multi(d1,d2)) mp_sum(d1,d2)...
  2*d2 d2 e 0 0 e 0 e e e;
e d1 d1 2*d1 d1 e e e 0 e 0 0 e e 0;
e mp_sum(d1,d2) mp_sum(d1,d2) mp_sum(2*d1,mp_multi(d1,d2)) mp_sum(d1,d2)...
  2*d2 d2 e 0 0 0 0 e e 0;
e e e e e e e e e 0 e e e e;
e mp_sum(d1,d2) mp_sum(d1,d2) mp_sum(2*d1,mp_multi(d1,d2)) mp_sum(d1,d2)...
  2*d2 d2 e 0 0 0 0 d3 0 0;
e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
  mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3;
e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
  mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3];

Bsys=[ 0 e ;
       0 0 ;
       0 0 ;
       0 e ;
       e e ;
       0 0 ;
       e e ;
       d1 e ;
       mp_sum(d1,d2) d2 ;
       d1 e ;
       mp_sum(d1,d2) d2 ;
       e e ;
       mp_sum(d1,d2) d2 ;
       mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3) ;
       mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3) ];

Csys=[e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
       mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
       mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3;
       e mp_multi(mp_sum(d1,d2),d3) mp_multi(mp_sum(d1,d2),d3)...
       mp_multi(mp_sum(2*d1,mp_multi(d1,d2)),d3) mp_multi(mp_sum(d1,d2),d3)...
       mp_multi(2*d2,d3) mp_multi(d2,d3) e d3 d3 d3 d3 2*d3 d3 d3];

Dsys=[mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3);
       mp_multi(mp_sum(d1,d2),d3) mp_multi(d2,d3)];

% the decision variables in this optimization problem are:
% 2*Np inputs u
% 2*Np output y
% 2*Np variables z
% total size of vector become 6*Np

Np = 1;
Nc = 1;

```



```

% determination of matrices H and g
[H,g]=calcHg(Asys,Bsys,Csys,Dsys,Np);

% x0 vector
x0=ones(15,1)*e;

% u0 vector
u0 = [0;0];

% upper and lower bound
ub = 12;
lb = 5;

% reference signal
r    = [2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 12 13 13
        14 14 15 15 16 16 17 17]*5;

% weighting parameters
lambda = 0.01;

% initial empty vectors for ustar, ystar and ybracket
vec_ystar = [];
vec_ustar = [];
vec_ybracket = [];

% kmax
kmax = 9;
for j = 1: kmax
    if j <= kmax
% \\\
% BLOCK 1

% this block goes with the objective function J = Jout + lambda*Jin

A1 = [ [zeros(2*Np,2*Np)] [eye(2*Np)] [-eye(2*Np)];
        [zeros(2*Np,2*Np)] [-eye(2*Np)] [-eye(2*Np)] ];
b1 = [ r(2*j-1:2*Np+2*(j-1))'; -r(2*j-1:2*Np+2*(j-1))' ];

A1eq = [];
b1eq = [];
% \\\

% \\\
% BLOCK 2
% this block goes with the max-plus dynamics: y = H*u + g*x0
% \\\

A2one = [];
b2one = [];
for i = 1:2*Np
A2oneS = [ [eye(2*Np)] [zeros(2*Np,i-1) -ones(2*Np,1) zeros(2*Np,2*Np-i)] ... ;
           [zeros(2*Np,2*Np)] ];
b2oneS = [-H(i,1:2*Np)'];
A2one = [A2one; A2oneS];

```

```

b2one = [b2one; b2oneS];
end
A2one;

A2two = [];
b2two = [];
for i = 1:2*Np
A2twoS = [ [zeros(15,2*Np)] [zeros(15,i-1) -ones(15,1) zeros(15,2*Np-i)] [zeros(15,2*Np)] ];
b2twoS = -[g(i,1) + x0(1); g(i,2) + x0(2); g(i,3) + x0(3); g(i,4) + x0(4); g(i,5) + x0(5); ...
           g(i,6) + x0(6); g(i,7) + x0(7); g(i,8) + x0(8); g(i,9) + x0(9); g(i,10) + x0(10); ...
           g(i,11) + x0(11); g(i,12) + x0(12); g(i,13) + x0(13); g(i,14) + x0(14); g(i,15) + x0(15); ];
A2two = [A2two; A2twoS];
b2two = [b2two; b2twoS];
end
A2two;

A2 = [A2one; A2two];
b2 = [b2one; b2two];

% replacement of all elements in A2 at row i into zero elements if the element of b(i) equals e
for i = 1:4*Np^2 + 30*Np
    if b2(i) == -e
        A2(i,1:6*Np) = zeros(1,6*Np);
        b2(i) = 0;
    else
        end
end

A2eq = [];
b2eq = [];

% \\\
% \\\
% BLOCK 3
% this block goes with the upper and the lower bounds of the input rates
% \\\

A3 = [ [eye(2*Np) + diag(-ones(2*Np-2,1),-2)] [zeros(2*Np,2*Np)] [zeros(2*Np,2*Np)];
       -[eye(2*Np) + diag(-ones(2*Np-2,1),-2)] [zeros(2*Np,2*Np)] [zeros(2*Np,2*Np)] ];
b3 = [ [ones(2,1)*ub + u0; ones(2*Np-2,1)*ub]; [-ones(2,1)*lb - u0; -ones(2*Np-2,1)*lb] ];

A3eq = [];
b3eq = [];

% \\\
% \\\
% BLOCK 4
% this block goes with the difference between Np and Nc
% \\\

```

```
A4eqS = [ [diag(-2*ones(2*Np,1)) + diag(ones((2*Np-2),1),-2) + diag(ones((2*Np-2),1),2)] ...
          [zeros(2*Np,2*Np)] [zeros(2*Np,2*Np)] ];
A4eq = [A4eqS(1:2*(Np - Nc),:)] ;
b4eqS = [ [-ones(2,1) + u0]; [zeros(2*(Np - Nc)-2,1)] ];
b4eq = [b4eqS(1:2*(Np - Nc),:)] ;

A4 = [];
b4 = [];

% =====

% merging of all equality and in-equality matrices

A = [A1; A2; A3; A4];
b = [b1; b2; b3; b4];

Aeq = [A1eq; A2eq; A3eq; A4eq];
beq = [b1eq; b2eq; b3eq; b4eq];

c = [-lambda*[ones(1,2*Np)] [ones(1,2*Np)] [ones(1,2*Np)] ];
lbx = [ [zeros(1,2*Np)] [zeros(1,2*Np)] [ones(1,2*Np)*e] ]';

[x,fval,exitflag,output] = linprog(c,A,b,Aeq,beq,lbx);

% optimal vector with decision variables
x;
ustar = x(1:2*Np);
ystar = x(2*Np+1:4*Np);

% update vector ustar
vec_ustar = [vec_ustar; ustar(1:2*Np)];

% update vector ystar
vec_ystar = [vec_ystar; ystar(1:2*Np)];

% determination of xkplus1 and ybracket
[xkplus1,ybracket] = process(Asys,Bsys,Csys,Dsys,ustar,x0);
%
% update vector ybracket
vec_ybracket = [vec_ybracket; ybracket];

% update new initial vector x0
x0 = xkplus1;

% update new initial input vector
u0 = ustar;

    else break
    end
end

% determination vec_dustar
vec_dustar(1)=vec_ustar(1);
vec_dustar(2)=vec_ustar(2);
```

```

for i=3:length(vec_ustar);
    vec_dustar(i)=vec_ustar(i) - vec_ustar(i-2);
end

% merging of all results in a matrix
ustar__dustar__ystar__ybracket__ref = [vec_ustar vec_dustar' vec_ystar vec_ybracket r(1:2*kmax*Np)']

%
%          2      Np
% determination of sum    sum |ystar - ybracket|
%          i=1    j=1
error_y_y = sum(abs(vec_ystar - vec_ybracket))

%
%          2      Np
% determination of sum    sum |ref - ybracket|
%          i=1    j=1
error_y_r = sum(abs(r(1:2*kmax*Np)' - vec_ybracket))

% \\\
% \\\
% PLOTTING
% \\\
% \\\

% product vector
prod = [ 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 9]
% input vector
x = ustar__dustar__ystar__ybracket__ref(:,1);

% output vector
y = ustar__dustar__ystar__ybracket__ref(:,4);

% reference vector
z = ustar__dustar__ystar__ybracket__ref(:,5);

for i = 1: 0.5*length(x)
    x1(i) = x(2*i-1);
    x2(i) = x(2*i);
    y1(i) = y(2*i-1);
    y2(i) = y(2*i);
    z1(i) = z(2*i-1);
    z2(i) = z(2*i);
end

X1= [];
X2 = [];
Y1 = [];
Y2 = [];
Z1 = [];
Z2 = [];

for i = 1:length(x1)
    X1S = [ x1(i) x1(i)];
    X2S = [ x2(i) x2(i)];

```

```

        Y1S = [ y1(i) y1(i)];
        Y2S = [ y2(i) y2(i)];
        Z1S = [ z1(i) z1(i)];
        Z2S = [ z2(i) z2(i)];
        X1 = [ X1 X1S ];
        X2 = [ X2 X2S ];
        Y1 = [ Y1 Y1S ];
        Y2 = [ Y2 Y2S ];
        Z1 = [ Z1 Z1S ];
        Z2 = [ Z2 Z2S ];
    end

% input signal plot

figure(1)
clf
plot(prod,X1,'c')
hold on
plot(prod,X2,'b')
legend('inputsequenceP1' , 'inputsequenceP2',0)
xlabel('p')
ylabel('t')

% output/reference signal plot

figure(2)
clf
plot(prod,Y1,'r')
hold on
plot(prod,Y2,'b')
plot(prod,Z1,'m')
plot(prod,Z2,'k')
legend('outputsequenceP1' , 'outputsequenceP2', 'referencesignalP1', 'referencesignalP2',0)
xlabel('p')
ylabel('t')

```

## G.2 *calcHG.m* and *process.m*

### *calcHG.m*

```

function [H,g]=calcHg(A,B,C,D,Np)
% m.file to calculate matrix H and g out of system matrices A, B, C and D
% which are used in the following state-space description:
%  $x(k+1) = A \otimes x(k) \oplus B \otimes u(k)$ 
%  $y(k) = C \otimes x(k) \oplus D \otimes u(k)$ 
% This m.file can be used in cases of one or more inputs!

```

```

% here Np is the length of the prediction horizon.

% date: 17 March 2004
% D. Wetjens

% input values are the systems matrices A, B, C, and D and Np

% example: x(0) = initial state
%           y(0) = C otimes x(0) oplus D otimes u(0)
%
%           x(1) = A otimes x(0) oplus B otimes u(0)
%           y(1) = C otimes x(1) oplus D otimes u(1)
%           = CA otimes x(0) oplus CB otimes u(0) oplus D otimes u(1)

% Using this algorithm, y(k) can be calculated as follows:
% then y(k) = H otimes u(k) oplus g otimes x(k)

% Matrices H and g are built up as follows (for reasons of clarity, the max-plus operations
% are replaced by conventional algebra operations):

% H = [D          .          .          .          .          .]
%      [CB         D          .          .          .          .]
%      [CAB        CB         D          .          .          .]
%      [CA^2B      CAB        CB         D          .          .]
%      [ :         :          :          CB         D          :]
%      [CA^(Np-1)B CA^(Np-1)B ..         ..         CB         D]

% g = [C      ]
%      [CA     ]
%      [CA^2   ]
%      [ :     ]
%      [CA^Np  ]

e=-inf;

d=length(D);
E=[ones(size(D))*e];

for i=1:Np;
    for j=1:Np;
        if i > j
            H((i-1)*d+1:i*d,(j-1)*d+1:j*d)= mp_multi(mp_multi(C,mp_power(A,i-j-1)),B);
        elseif i==j
            H((i-1)*d+1:i*d,(j-1)*d+1:j*d)= D;
        else
            H((i-1)*d+1:i*d,(j-1)*d+1:j*d) = E;
        end
        g((i-1)*d+1:i*d,:)=mp_multi(C,mp_power(A,i-1));
    end
end
end

```

**process.m**

```
function [xkplus1,yk] = process(Asys,Bsys,Csys,Dsys,u1,xstart)

xkplus1 = mp_sum(mp_multi(Asys,xstart),mp_multi(Bsys,u1));
yk      = mp_sum(mp_multi(Csys,xstart),mp_multi(Dsys,u1));
```