

Stochastic Switching Max-Plus-Linear Systems

Theory and Aspects

M.J. van Loenhout

Master of Science Thesis

Stochastic Switching Max-Plus-Linear Systems

Theory and Aspects

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Systems and Control at Delft
University of Technology

M.J. van Loenhout

August 22, 2011

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of
Technology

Abstract

Model Predictive Control (MPC) is a popular controller design method in the process industry. In the current literature, the principles of MPC has been applied to Max-Plus-Linear (MPL) systems. These classes have been extended to include either parametric uncertainty (such as noise and modeling errors) or mode switching uncertainty (where each mode is described by a different MPL system).

In literature, an approach has been found that incorporates both types of uncertainty in the same system. However, since this approach assumes bounded uncertainty and uncertain switching, the Model Predictive Controller can only assume the worst-case scenario. Therefore, a stochastic approach is proposed in this report, which allows us to use more information about the uncertainties. We derive a (general) model that includes both (stochastic) parametric uncertainty and (deterministic and/or stochastic) mode switching uncertainty.

Using MPC involves computing the minimum of a cost function. For this model, the cost function becomes very complex, since both uncertainties may be dependent. If we can assume that both uncertainties are independent, we have shown that we can split the problem in two subproblems, that are already analyzed in the current literature. These subproblems are already proved to be convex problems, and approximation methods have already been analyzed for these methods. If we cannot assume that uncertainties are independent, the cost function becomes very complex. We have proposed a method that reduces the computational complexity of this cost function.

Keywords: *Discrete Event Systems, Max-Plus-Linear Systems, Model Predictive Control, Stochastic Disturbance, Mode Switching.*

Table of Contents

Preface	xi
Acknowledgements	xiii
1 Introduction	1
1-1 Background	1
1-2 Literature Overview	2
1-3 Project Description	3
1-4 Practical Examples	3
2 Stochastic Switching Max-Plus-Linear Systems	7
2-1 Max-Plus Algebra	7
2-2 Deterministic Max-Plus-Linear Systems	8
2-3 Mode Switching Uncertainty	10
2-4 Parametric Uncertainty	11
2-5 Parametric and Mode Switching Uncertainty	13
2-6 Difference Between Uncertainties	16
2-7 Conclusions	16
3 Model Predictive Control	19
3-1 Why Model Predictive Control?	19
3-2 Basic Ingredients	19
3-3 Prediction Model	20
3-4 Cost Function	22
3-5 Constraint Handling	24
3-6 Optimization	25
3-7 Receding Horizon Principle – Timing Aspects	25
3-8 Interconnections and Implementation	26
3-9 Conclusions	26

4	Cost Function	29
4-1	Joint Density Function	29
4-2	Switching Probability	30
4-3	Classification I – Mode Switching Behavior	31
4-3-1	Deterministic Mode Switching	31
4-3-2	Stochastic Mode Switching	32
4-4	Classification II – Dependency between Random Variables	32
4-4-1	The Previous Mode $\ell(k-1)$	32
4-4-2	The Previous State $x(k-1)$	33
4-4-3	The Input Signal $\tilde{u}(k)$	34
4-4-4	The (Auxiliary) Control Signal $\tilde{v}(k)$	34
4-4-5	The General Case	35
4-5	Assumptions that Reduce the Computational Complexity	35
4-5-1	Type 1: Independent Random Variables, Deterministic Switching	36
4-5-2	Type 2: Independent Random Variables, Stochastic Switching	37
4-5-3	Type 3: Dependent Random Variables, Deterministic Switching	39
4-5-4	Type 4: Dependent Random Variables, Stochastic Switching	41
4-6	Conclusions	44
5	Computational Aspects	45
5-1	Problem Formulation	45
5-2	Mode Switching Sequence	46
5-3	Probabilities for each Mode Switching Sequence	46
5-3-1	Full Problem	47
5-3-2	Approximation – Scenario Based Approximation	47
5-4	Cost Function for each Mode Switching Sequence	48
5-4-1	Full Problem	48
5-4-2	Approximation – Norm Based Approximation	49
5-5	Convexity	51
5-6	Conclusions	51
6	Worked Example	53
6-1	Model Description	53
6-2	First Observations	54
6-3	Model	55
6-4	Implementation	55
6-5	Controller Performance	55
6-6	Small Note on Stability	56
6-7	Conclusion	59

7	Conclusions	61
7-1	Conclusion	61
7-2	Limitations	62
7-3	Recommendations	62
A	Example Code	65
A-1	Matlab	65
A-1-1	Main.m	65
A-1-2	ConvexOptimization.m	67
A-1-3	CostFunction.m	68
A-1-4	CostFunctionOutput.m	68
A-1-5	CostFunctionInput.m	69
A-1-6	NormBasedApproximation.m	69
A-1-7	Hermite.m	70
A-1-8	CostConstraints.m	70
A-1-9	ClosedLoopSimulation.m	71
A-1-10	MonteCarloSimulations.m	71
A-2	Linking Matlab and Mathematica	72
A-2-1	MathLink.m	72
A-2-2	MathLinkFunctions.m	74
	Bibliography	79
	Glossary	81
	List of Acronyms	81
	List of Symbols	81

List of Figures

1-1	Two examples of uncertainties in production systems.	4
1-1a	Parametric Uncertainty.	4
1-1b	Mode Switching Uncertainty.	4
1-2	Two examples of uncertainties in railway systems.	5
1-2a	Parametric Uncertainty.	5
1-2b	Mode Switching Uncertainty.	5
2-1	Production system.	9
2-2	Modeling approach.	14
3-1	Interconnections in the Controller.	27
5-1	Growth of number of mode switching sequences.	46
6-1	Production system.	53
6-2	Tuning the Norm Based Approximation method, with the parameter p	57
6-3	Tuning the Model Predictive Controller, with the parameter λ	58
6-4	Unstable System, where the tracking error grows unbounded.	58

List of Tables

4-1	Four types of Stochastic Switching Max-Plus-Linear systems.	35
6-1	Average time durations.	57
6-2	Average time durations with use of the Scenario Based Algorithm.	57

Preface

This Master Thesis is part of the Final Thesis, required to obtain a Master's Degree in Systems & Control. The graduation project has been carried out at Delft Center for Systems and Control (DCSC), a collaboration of three former 'Systems & Control'-groups of the departments of Electrical Engineering (EWI), Mechanical Engineering (3mE) and Applied Physics (TNW). The program is an interdisciplinary MSc-program, which is now part of the 3TU.Federation, a cooperation between the three leading universities in the Netherlands.

When I started my MSc Research Project, I was investigating which research topics I was interested in. I came along several subjects, of which System Identification was of my particular interest. Maybe because I liked the straightforwardness of the theory presented in this course, or maybe because the teacher (Xavier Bombois) was so widely enthusiastic about it and explained everything very well. However, I could not find a suitable project description in this direction. So, I decided to look a bit further, more open minded. During this process, I investigated other directions as well. One of these was the subject of MPL systems. Personally, I never heard of these kind of systems, so it drew my attention. I made an appointment with my supervisor (Ton van den Boom), and he explained me what it was all about. During our conversations, I became more and more enthusiastic about the research area. Finally, he presented me some research directions that he thought it would be interesting to spend some research on. That was the beginning of my MSc Thesis Project, that took approximately one year to complete.

This MSc Thesis Project consists of two main parts. The first part is a Literature Survey, in which I investigated the related literature. I tried to expound every aspect of the literature as precise as possible, which resulted in some very detailed and long conversations with my supervisors. Since my supervisor (Ton van den Boom) is one of the first researchers that applied the principles of MPC to these MPL systems, he has a very good understanding of every aspect I was thinking of.

The second part is this MSc Thesis report, in which I present the results I have gathered during the final stage of my MSc Project. This report is divided into several chapters, each of which deals with a specific part in the road to complete this MSc Thesis. Since most chapters will be quite theoretical (at least for the inexperienced reader), I have assumed that the reader has a good understanding of conventional systems (such as Linear Time-Invariant (LTI) systems) and has a basic understanding of probability theory.

For the reader that has a basic understanding of Max-Plus-Linear systems and is interested in the other topics in this report, a short guideline for the chapters is given below.

Chapter 2: We will introduce a new class of MPL systems, which we will call Stochastic Switching Max-Plus-Linear (SSMPL) systems. These systems will include both parametric uncertainty and mode switching uncertainty simultaneously. The theory will be guided with some examples.

Chapter 3: The basic principles of MPC are expounded. We will explain the basics of MPC and how we can apply these principles to our new class. We will see that we need to compute the expectation over both uncertainties.

Chapter 4: We will investigate how we can compute this expectation, that is required to obtain the optimal control signal. Various cases are described, where under some general assumptions, the expectation can be computed in a more efficient way.

Chapter 5: If we assume these general assumptions, we still need to perform some computations. These are already analyzed in the current literature, and discussed in this chapter.

Chapter 6: The theory is applied to a simple example. The required theory is incorporated in MATLAB and MATHEMATICA. Shortly, the results are discussed as well.

Chapter 7: We will conclude this report with some discussion and the final conclusion. In the end, few recommendations for future work are presented as well.

Acknowledgements

Without the help of my supervisors, Samira Farahani and Ton van den Boom, completing this MSc Thesis wouldn't be possible. I really appreciate the time and effort they took to help me. Further, I would like to thank Bart De Schutter and Bernd Heidergott for joining the Examining Committee.

Samira¹, as my daily supervisor, gave me useful feedback on my work and pointed me into inconsistencies. Since she is a mathematician, she helped me with the derivations, probability theory and gave me tips on notations.

Ton², on the other hand, is in principle an expert in this research area. Therefore, he could explain me all the theory and made me understand everything well. He gave me some insightful tips as well, while respecting my opinion and vision on this MSc Thesis.

Bart³, as a professor at the TU Delft, has been working extensively with Ton on this topic. Bart has many interests, under which the control of Discrete Event Systems (DES). He has many graduates and PhD students working on this topic, including Samira (as a PhD). He even replaced Ton during his absence.

Bernd⁴, from the Vrije Universiteit Amsterdam (literal translation of 'Free University'), is an associate professor at the faculty of Economics and Business Administration. One of his topics of interests are DES as well. He has written a book titled 'Max Plus at Work' (see [1]), which was very insightful for me to get to the bottom of Max-Plus Algebra (MPA) and MPL systems.

Delft, University of Technology
August 22, 2011

M.J. van Loenhout

¹Personal Homepage: <http://www.dcsc.tudelft.nl/~sfarahani/>

²Personal Homepage: <http://www.dcsc.tudelft.nl/~tvandenboom/>

³Personal Homepage: <http://www.dcsc.tudelft.nl/~deschutt/>

⁴Personal Homepage: <http://personal.vu.nl/b.f.heidergott/>

“The essence of mathematics is not to make simple things complicated, but to make complicated things simple...” — *Stanley Gudder*

Chapter 1

Introduction

This chapter mainly serves as a short introduction to this MSc Project. A more extensive analysis can be found in the Literature Survey, which is part of this MSc Project as well.

1-1 Background

Most systems can be classified as either continuous or discrete systems. Continuous systems are guided by differential equations, which represent the dynamical relation between the input and the output of a system. Discrete systems are guided by difference equations, which are the discrete representations of differential equations.

Another way of classification is based on the way that the state changes over time. For conventional time-driven systems, the state changes as time progresses. However, for event-driven systems, the state changes due to the occurrence of an event. An event is a primitive concept with a good intuitive basis. An event should be thought of as occurring instantaneously and causing transitions from one state value to another.

The class of Discrete Event Systems (DES) is a large class of systems that is event-driven. This class is discrete as well, since either an event did occur or did not occur. Time is only defined between events and events are in general not equidistant. The class of DES essentially consists of man-made systems that contain a finite number of resources (such as machines, communication channels, or processors) that are shared by several users (such as product types, information packets, or jobs) all of which contribute to some common goal (the assembly of products, the end-to-end transmission of a set of information packets, or parallel computation) [2].

In general, most of these systems are nonlinear. However, some of these systems have linear properties, both over the parameters and over time. For conventional time driven system, such systems are called Linear Time-Invariant (LTI) systems. The focus of this MSc Project is on a special subclass of DES, namely Max-Plus-Linear (MPL) systems. As the name suggests, this class of systems is linear in the Max-Plus Algebra (MPA). Loosely speaking, this class

consists of DES with synchronization but no choice, or concurrency. MPA is a different form of algebra, but has many properties that can be easily translated from conventional algebra.

1-2 Literature Overview

A first approach has been found to perform the principles of Model Predictive Control (MPC) to a deterministic MPL system [3], a system in which no uncertainty is involved in the development of future states of the system. MPC is very powerful for MPL systems, it allows easy computation of future states and constraints on the signals.

The first attempt to extend the deterministic MPL system with parametric uncertainty assumed only bounded uncertainty [4]. Noise, disturbances and modeling errors can be modeled as parametric uncertainty. Since we only know the upper and lower bounds of the uncertainty, we can only assume the worst-case model at every event step. This has a negative impact on the performance of the controller.

To solve this issue, a second approach has been made in which the uncertainty was assumed to be stochastic [5]. This assumption allows us to use more information about the parametric uncertainty, such as the expectation. This method requires (numerical) integration over all uncertainties, which is very time consuming.

Since computing the optimal control signal was still hard, an approximation method has been proposed [6]. This method uses norm inequalities to obtain an analytic expression for the expectation and accordingly to improve the computation of the optimal control signal. This approach improves the performance of the controller significantly, and still gives satisfactory results.

Next to the research on parametric uncertainty, mode switching uncertainty was also introduced [7]. Such a system has different modes, and the system can switch from one mode to another between events. For each mode, the model can be described by a (different) MPL system. This paper only investigated the technical issues, not the computational issues.

The next step was to deal with these computational issues. First, it was assumed that there was only stochastic mode switching uncertainty [8]. Stochastic mode switching uncertainty means that the probabilities for the system to switch from one mode to the other were fixed. Under this assumption, the difficulty in obtaining the optimal control signal was reduced significantly.

At a second stage, a more general case of mode switching uncertainty is investigated [9]. Here, we deal with both deterministic and stochastic mode switching uncertainty. Deterministic means that the next mode is known in the current event step, however, it might still be a function of some event-dependent parameters such as the state.

If we have a large system¹, the computation of the optimal control signal is both time consuming and expensive. Therefore, an approximation method is proposed [10], which reduces the computational burden by only investigating the most probable modes that might occur in the system. This reduces the problem significantly, and still gives satisfactory results.

¹Here, large means: large state space, large prediction horizon.

Finally, a first approach has been made that includes both the parametric uncertainty and mode switching uncertainty simultaneously [11]. It uses bounded parametric uncertainty and uncertain mode switching. This means that we again cannot use any information about these uncertainties, and we thus again need to use the worst-case model.

1-3 Project Description

In the current literature, we have seen that the class of MPL system has already been extended to include either mode switching uncertainty or parametric uncertainty. A combination of both uncertainties simultaneous has already been studied as well, but this approach only uses bounded parametric uncertainty and uncertain switching.

The goal of this MSc Project is to define a new (sub)class of MPL systems that include both deterministic and/or stochastic switching and stochastic parametric uncertainty. The use of stochastic parametric uncertainty and deterministic and/or stochastic mode switching may improve the performance of the Model Predictive Controller, since more information about the uncertainty is available.

Once we have a model available, we will design a Model Predictive Controller for this new class. The main reason that we use the principles of MPC is that the use of a DES imposes some additional constraints. In order to obtain the optimal control signal, we need to compute the expectation over two types of random variables. This might impose both theoretical and computational issues when both random variables are dependent.

1-4 Practical Examples

Once we have defined this new class of MPL systems and we have defined a controller for it, we can apply it to real systems. Here, we assume that these systems have both types of uncertainties; deterministic and/or stochastic mode switching uncertainty and stochastic parametric uncertainty. We will shortly discuss two typical examples of MPL systems and how we can model the uncertainties of these systems.

Production Plant

Goal: Manufacture and assemble products. This goal is reached with several machines that are used to produce parts or to assemble parts into the final product. Raw materials or semifinished products enter the system. Each machine performs a specific task and the goods are routed between the machines in a specific order (called recipes). If the (semifinished) product is completed, it will leave the system.

Parametric Uncertainty: The processing times of the machines and the transportation times between the machines are not precisely known. If the machines are operated by a human, the uncertainty of its duration or transportation time is in general large. On the other hand, uncertainty in the modeling phase may lead to errors in the system matrices.



1-1a: Parametric Uncertainty: The processing time of a human welder is unknown and very sensitive to disturbances.



1-1b: Mode Switching Uncertainty: The machine may have an unexpected breakdown, and rerouting might be required.

Figure 1-1: Two examples of uncertainties in production systems.

Mode Switching Uncertainty: For each recipe, different machines are required and the order in which these machines are needed may be different as well. If similar machines are present in the plant, different modes can be used to distribute the products over all the machines. If one of these machines has a breakdown, some modes might become unavailable.

Controller Objective: Design an optimal schedule for the production plant, taking all parametric uncertainties into account. The controller could then optimize the capacity utilization of the machines.

Railway System

Goal: Transport passengers between stations. This goal is reached with several trains and employees that are all used to transport passengers from one station to another. Railway tracks and switches are used to route the trains between the stations. The trains are regulated with an hourly train schedule.

Parametric Uncertainty: The current train schedule is designed such that the passengers have short waiting times. If there is a delay (disturbance), this delay might propagate through the railway system, since there are some connections between the departure times of the trains (to allow the passengers to change trains).

Mode Switching Uncertainty: To minimize this delay, operators at the station may decide to reorder the departures of some trains to minimize the effect of a single delay through the railway network. If the departure times of some trains are reordered, the train schedule must be adopted with new information (since there is a finite capacity of the tracks and stations).



1-2a: Parametric Uncertainty: The arrival times at a station are not precisely known.



1-2b: Mode Switching Uncertainty: Switches allows rescheduling of trains.

Figure 1-2: Two examples of uncertainties in railway systems.

Controller Objective: Find the most optimal reordering, taking some (unknown) delays into account. The operators could use this information to reorder the departures of some trains in a more efficient way.

Chapter 2

Stochastic Switching Max-Plus-Linear Systems

In this chapter, the class of Max-Plus-Linear (MPL) systems is extended to include two types of uncertainties, namely stochastic parametric uncertainty and deterministic and/or stochastic mode switching uncertainty. This new class will be called: Stochastic Switching Max-Plus-Linear (SSMPL) systems.

2-1 Max-Plus Algebra

In Max-Plus Algebra (MPA), the conventional additive and multiplicative operators are replaced by maximization and addition respectively. Define $\varepsilon := -\infty$ and $e := 0$ as the max-plus algebraic zero and identity elements respectively. Denote $\mathbb{R}_{\max} := \mathbb{R} \cup \{\varepsilon\}$, where $\mathbb{R}$ is the set of real numbers. For elements $a, b \in \mathbb{R}_{\max}$, the additive ($\oplus$) and multiplicative ($\otimes$) operators are defined as follows:

$$\begin{aligned}a \oplus b &:= \max(a, b) \\ a \otimes b &:= a + b\end{aligned}$$

Just like in conventional algebra, the $\otimes$ -operator has priority over the $\oplus$ -operator. So:

$$a \otimes b \oplus c \otimes d = (a \otimes b) \oplus (c \otimes d)$$

Remark 1. *In most papers and books, the $\otimes$ -operator is omitted, as is the case with the conventional multiplicative operator. In the proceedings of this report, both conventional algebra and MPA will be used simultaneously. Therefore, to prevent confusion, we keep the $\otimes$ -operator and omit the conventional multiplicative operator everywhere.*

For matrices, similar definition exists:

$$[A \oplus B]_{(i,j)} := a_{(i,j)} \oplus b_{(i,j)}$$

$$[A \otimes C]_{(i,j)} := \bigoplus_{k=1}^n a_{(i,k)} \otimes c_{(k,j)}$$

for $A, B \in \mathbb{R}_{\max}^{m \times n}$ and $C \in \mathbb{R}_{\max}^{n \times p}$. Here, the element of a general matrix $A \in \mathbb{R}_{\max}^{m \times n}$ in row i and column j is denoted as $a_{(i,j)}$. Occasionally, the element $a_{(i,j)}$ will also be denoted as $[A]_{(i,j)}$.

2-2 Deterministic Max-Plus-Linear Systems

MPL systems are Discrete Event Systems (DES) that are linear in the MPA. The basic operations for a MPL system are addition of event times and maximization of process times. In [2, 12], it has been shown that discrete event systems in which there is synchronization, but no choice or concurrency can be described by a model of the form:

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

$$y(k) = C(k) \otimes x(k) \quad (2-1b)$$

with $A(k) \in \mathbb{R}_{\max}^{n \times n}$, $B(k) \in \mathbb{R}_{\max}^{n \times n_u}$ and $C(k) \in \mathbb{R}_{\max}^{n_y \times n}$, where n is the internal state dimension, n_u is the number of inputs and n_y is the number of outputs¹. Here, $x(k)$ is the (internal) state, $u(k)$ the input and $y(k)$ the output of the system at event step k .

The index k is called the event counter. The matrices A , B and C usually consist of sums or maximization's of internal processing times, transportation times, etc. For DES, the state $x(k)$ typically contains the time instants at which the internal events occur for the k th time, the input $u(k)$ contains the time instants at which the input events occur for the k th time and the output $y(k)$ contains the time instants at which the output events occur for the k th time.

Remark 2. *Throughout this MSc Thesis, we will try to use the same example over and over again. This allows the reader to find the differences between all the examples and extensions quite easily. This example is adopted from [8].*

Example 1. *Consider the production system of figure 2-1. This system consists of three machines M_1 , M_2 and M_3 . Three products (A , B and C) can be produced by this system, each with its own recipe, meaning that the order in the production sequence is different for every product. For now, we will only consider the manufacturing of product A .*

For product A , the production order is M_1 – M_2 – M_3 , which means that the raw material is fed to machine M_1 , where it is processed. The intermediate product is sent to machine M_2 for further processing, and finally the product is finished in machine M_3 . Each machine starts working as soon as possible on each batch, i.e. as soon as the raw material or the required intermediate products are available, and as soon as the machine is idle (i.e. the previous batch

¹In the remainder of this report, we will assume that the dimensions of the matrices and vectors are appropriate. This will not be repeated for every model description.

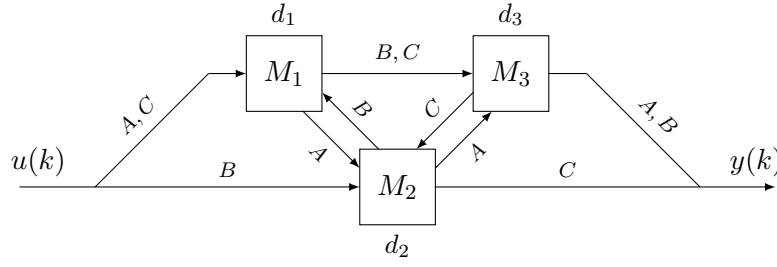


Figure 2-1: Production system.

has been finished and has left the machine). We define $u(k)$ as the time instant at which the system is fed for the k th time, $x_i(k)$ as the time instant at which machine i starts for the k th time, and $y(k)$ as time instant at which the k th product leaves the system.

We assume that all the internal buffers are large enough, and no overflow will occur. We assume that the processing time of the machines M_1 , M_2 and M_3 are given by d_1 , d_2 and d_3 respectively, and the transportation times between the machines are assumed to be negligible.

Machine M_1 can start on a new product if:

1. It has finished its previous production cycle:

$$x_1(k) \geq x_1(k-1) + d_1$$

2. New material has been transported to the machine:

$$x_1(k) \geq u(k)$$

Since it is assumed that each machine starts working as soon as possible, one of these equations should become equality, and thus:

$$x_1(k) = \max(x_1(k-1) + d_1, u(k)) \quad (2-2a)$$

Similarly, we get for the other two machines:

$$x_2(k) = \max(x_1(k) + d_1, x_2(k-1) + d_2) \quad (2-2b)$$

$$x_3(k) = \max(x_2(k) + d_2, x_3(k-1) + d_3) \quad (2-2c)$$

And the product leaves the system as soon as it finishes processing:

$$y(k) = x_3(k) + d_3 \quad (2-2d)$$

Since we want to create a model like in equation (2-1), we need to eliminate the $x_1(k)$ in (2-2b) by substituting (2-2a) in (2-2b) and eliminate $x_2(k)$ in (2-2c) by substituting (2-2b) in (2-2c) respectively. This results in:

$$x_1(k) = \max(x_1(k-1) + d_1, u(k))$$

$$x_2(k) = \max(x_1(k-1) + 2d_1, x_2(k-1) + d_2, u(k) + d_1)$$

$$x_3(k) = \max(x_1(k-1) + 2d_1 + d_2, x_2(k-1) + 2d_2, x_3(k-1) + d_3, u(k) + d_1 + d_2)$$

$$y(k) = x_3(k) + d_3$$

which leads to the following system matrices for recipe A , and can be substituted in (2-1):

$$A = \begin{bmatrix} d_1 & \varepsilon & \varepsilon \\ 2d_2 & d_2 & \varepsilon \\ 2d_1 + d_2 & 2d_2 & d_3 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ d_1 \\ d_1 + d_2 \end{bmatrix} \quad C = \begin{bmatrix} \varepsilon & \varepsilon & d_3 \end{bmatrix}$$

2-3 Mode Switching Uncertainty

We consider DES that can switch between different modes of operation. The mode switching allows us to model a change in the structure of the system, such as breaking a synchronization or changing the order of events.

For each mode, there is a different set of difference equations. So, for each mode, we can find some system matrices A , B and C . Denote the total number of modes in the system by n_L and the mode of the system at event step k by $\ell(k)$. Then, at each event k , the system must be in one of these modes, so we have:

$$\ell(k) \in \{1, \dots, n_L\} \quad \forall \quad k$$

With use of this new notation, the system can be described as:

$$x(k) = A_{\ell(k)} \otimes x(k-1) \oplus B_{\ell(k)} \otimes u(k) \quad (2-3a)$$

$$y(k) = C_{\ell(k)} \otimes x(k) \quad (2-3b)$$

where A_ℓ , B_ℓ and C_ℓ are the system matrices that corresponds to mode ℓ .

The movements of mode switching are determined by a switching mechanism. The probability of switching to mode ℓ at event step k may depend on the mode at the previous event step $\ell(k-1)$, the state at the previous event step $x(k-1)$, the input signal $u(k)$ and/or an (additional) control signal $v(k)$:

$$P[L(k) = \ell(k) \mid \ell(k-1), x(k-1), u(k), v(k)]$$

where $L(k)$ is a stochastic variable and $\ell(k)$ is its value. Since $P[L(k) = \ell(k)]$ is a probability, we have:

$$0 \leq P[L(k) = \ell(k) \mid \ell(k-1), x(k-1), u(k), v(k)] \leq 1 \quad \text{for } \ell(k) = 1, \dots, n_L$$

and:

$$\sum_{\ell(k)=1}^{n_L} P[L(k) = \ell(k) \mid \ell(k-1), x(k-1), u(k), v(k)] = 1$$

Example 2. Consider again the production system as in example 1. For product A the processing order is $M_1-M_2-M_3$, for product B , the processing order is $M_2-M_1-M_3$, and for product C the processing order is $M_1-M_3-M_2$. Using the same procedure, we find the three system descriptions for all three recipes (where the system matrices for recipe A have been calculated earlier):

$$A_1 = \begin{bmatrix} d_1 & \varepsilon & \varepsilon \\ 2d_1 & d_2 & \varepsilon \\ 2d_1 + d_2 & 2d_2 & d_3 \end{bmatrix} \quad B_1 = \begin{bmatrix} 0 \\ d_1 \\ d_1 + d_2 \end{bmatrix} \quad C_1 = \begin{bmatrix} \varepsilon & \varepsilon & d_3 \end{bmatrix}$$

$$\begin{aligned}
A_2 &= \begin{bmatrix} d_1 & 2d_2 & \varepsilon \\ \varepsilon & d_2 & \varepsilon \\ 2d_1 & d_1 + 2d_2 & d_3 \end{bmatrix} & B_2 &= \begin{bmatrix} d_2 \\ 0 \\ d_1 + d_2 \end{bmatrix} & C_2 &= \begin{bmatrix} \varepsilon & \varepsilon & d_3 \end{bmatrix} \\
A_3 &= \begin{bmatrix} d_1 & \varepsilon & \varepsilon \\ 2d_1 + d_3 & d_2 & 2d_3 \\ 2d_1 & \varepsilon & d_3 \end{bmatrix} & B_3 &= \begin{bmatrix} 0 \\ d_1 + d_3 \\ d_1 \end{bmatrix} & C_3 &= \begin{bmatrix} \varepsilon & d_2 & \varepsilon \end{bmatrix}
\end{aligned}$$

The mode switching probability from one recipe to the next one is assumed to be given by:

$$\begin{aligned}
P[L(k) = 1 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.64 \\
P[L(k) = 1 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 1 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 2 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 2 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.64 \\
P[L(k) = 2 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 3 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 3 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 3 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.64
\end{aligned}$$

which means that, if we have a specific recipe for the k th product, then the probability of having the same recipe for the $(k+1)$ th product is 64%, and the probability to switch to each other recipe is 18%.

Example 3. Consider the same production system as in example 2. Now, we assume that products B and C are equivalent, but are produced using different recipes. This means that, if we need to produce product B or C , we can choose the recipe. We therefore introduce an auxiliary binary control variable $v(k) \in \{0, 1\}$ that can be used to choose between processing recipe B and C . The mode switching probability from one recipe to the next one is now given by:

$$\begin{aligned}
P[L(k) = 1 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.64 \\
P[L(k) = 1 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 1 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.18 \\
P[L(k) = 2 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.36v(k) \\
P[L(k) = 2 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.64v(k) \\
P[L(k) = 2 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.64v(k) \\
P[L(k) = 3 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.36(1 - v(k)) \\
P[L(k) = 3 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.64(1 - v(k)) \\
P[L(k) = 3 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.64(1 - v(k))
\end{aligned}$$

2-4 Parametric Uncertainty

We consider DES that contain some uncertain parameters. In this report, we will only consider stochastic parametric uncertainty. In [4], a detailed analysis is performed with

bounded parametric uncertainty. This means that we do not know how this uncertainty behaves, therefore, we need to assume the worst-case scenario for every event step². The use of stochastic parametric uncertainty allows us to use more information about the uncertainty than just the upper and lower bounds.

Noise and modeling errors are in general additive. For MPL systems, the uncertainty is still additive in the conventional algebra. However, these systems are described in MPA, where the additive operator from conventional algebra is equivalent to the multiplicative operator from MPA. Thus, the uncertainty is multiplicative (in the MPA) and (some of the elements of) the system matrices vary over time. However, the structure of the system remains intact.

We can capture all the uncertainty of the system in a vector, denoted by $e(k)$. Since the uncertainty is stochastic, $e(k)$ is a stochastic random variable. In general, this uncertainty consists of stochastic random variables that occur in various (nearby) event steps, and has the following form:

$$e(k) = \begin{bmatrix} d_1(k-1) \\ d_1(k) \\ \vdots \\ d_2(k-1) \\ d_2(k) \\ \vdots \end{bmatrix}$$

where $d_i(k-1), d_i(k), \dots$ are independent and identically distributed (i.i.d.) random variables.

Accordingly, all elements of the system matrices are perturbed by the uncertainty $e(k)$. Thus we have the following system description:

$$x(k) = A(e(k)) \otimes x(k-1) \oplus B(e(k)) \otimes u(k) \quad (2-4a)$$

$$y(k) = C(e(k)) \otimes x(k) \quad (2-4b)$$

where $A(e(k))$, $B(e(k))$ and $C(e(k))$ are the system matrices with (shared) uncertainty vector $e(k)$.

Example 4. Consider the same production system as in example 1, where we only looked at the model for recipe A. Here, we cannot assume that d_1 , d_2 and d_3 are constant for every k , thus we rewrite the system equations with $d_1(k)$, $d_2(k)$ and $d_3(k)$:

$$\begin{aligned} x_1(k) &= \max(x_1(k-1) + d_1(k-1), u(k)) \\ x_2(k) &= \max(x_2(k-1) + d_2(k-1), x_1(k) + d_1(k)) \\ &= \max(x_1(k-1) + d_1(k-1) + d_1(k), x_2(k-1) + d_2(k-1), u(k) + d_1(k)) \\ x_3(k) &= \max(x_3(k-1) + d_3(k-1), x_2(k) + d_2(k)) \\ &= \max(x_1(k-1) + d_1(k-1) + d_1(k) + d_2(k), x_2(k-1) + d_2(k-1) + d_2(k), \\ &\quad x_3(k-1) + d_3(k-1), u(k) + d_1(k) + d_2(k)) \\ y(k) &= x_3(k) + d_3(k) \end{aligned}$$

²In fact, we need to determine the worst-case model, and then find the optimal control signal for this worst-case model.

If we now assume that the duration time of machines M_1 , M_2 and M_3 are uncertain, we have the following uncertainty vector:

$$e(k) = \begin{bmatrix} d_1(k-1) \\ d_1(k) \\ d_2(k-1) \\ d_2(k) \\ d_3(k-1) \\ d_3(k) \end{bmatrix}$$

2-5 Parametric and Mode Switching Uncertainty

The most obvious path to define the new class of DES is to first create a deterministic MPL system. Then, include mode switching uncertainty and finally, for each mode independently, determine the system matrices for that mode with parametric uncertainty. The main reason for this approach will become clear at a later stage. The benefit for modeling both uncertainties independently allows us to use methods that are already studied in literature, for example in [5, 9]. In figure 2-2, the modeling technique is visualized, where it is assumed that the probability of switching to the next mode depends on the previous mode only.

Using this modeling approach, we can introduce parametric uncertainty for each mode independently. This means that each mode has its own uncertainty vector at event step k :

$$e_1(k), \dots, e_{n_L}(k)$$

where n_L again denotes the total number of modes in the system.

The next step would be to combine all uncertainties of the system into a single uncertainty vector $e(k)$:

$$e(k) = \begin{bmatrix} e_1(k) \\ \vdots \\ e_{n_L}(k) \end{bmatrix}$$

It is very likely that each uncertainty vector for each mode contains the same stochastic random variables. Therefore, we introduce the union-operator that acts on a vector. If we want to apply this operator on an arbitrary vector X , we will denote this by: $X_{\cup}$. It creates a new vector that contains the union of all elements of this vector. An example will follow:

Consider two vectors, X_1 and X_2 . They contain the following variables: a , b and c . If we stack both vectors, we obtain Y . We see that Y contains a duplicate of the variable b . If we now apply the vector based union operator on Y , we obtain³:

$$X_1 = \begin{bmatrix} a \\ b \end{bmatrix} \quad X_2 = \begin{bmatrix} b \\ c \end{bmatrix} \quad \Rightarrow Y = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} a \\ b \\ b \\ c \end{bmatrix} \quad \Rightarrow Y_{\cup} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

³We can as well find a transformation matrix T to obtain the same results.

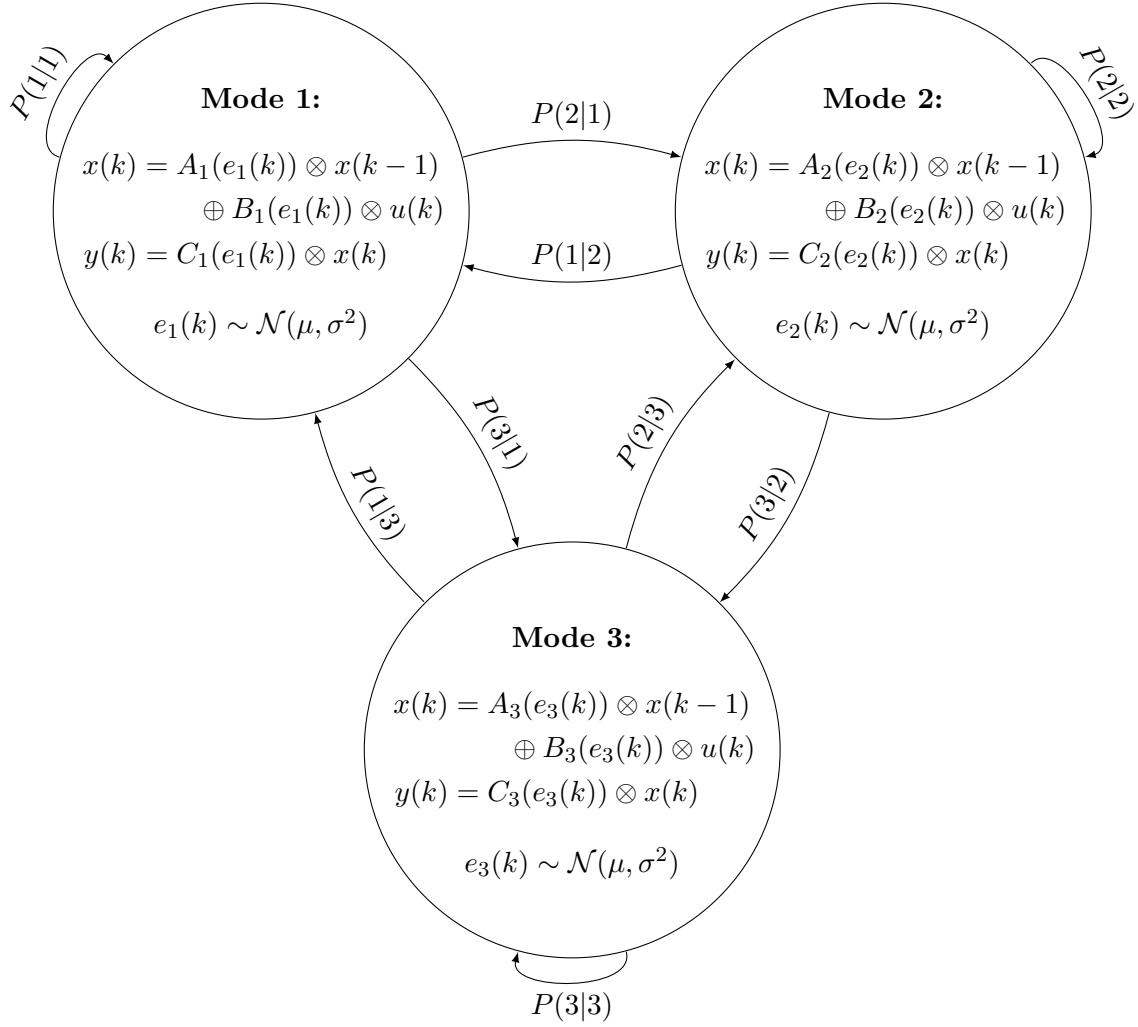


Figure 2-2: Modeling approach.

Remark 3. *In the remainder of this report, we assume that the uncertainty vector of the generalized system does not contain any duplicate entries. The above method is merely a mathematical definition to deal with these duplicate entries in the first place.*

So, now that we have the generalized uncertainty vector for the system, we can define the difference equations of the new class of MPL systems:

$$x(k) = A_{\ell(k)}(e(k)) \otimes x(k-1) \oplus B_{\ell(k)}(e(k)) \otimes u(k) \quad (2-5a)$$

$$y(k) = C_{\ell(k)}(e(k)) \otimes x(k) \quad (2-5b)$$

where $A_{\ell(k)}(e(k))$, $B_{\ell(k)}(e(k))$ and $C_{\ell(k)}(e(k))$ are the system matrices that correspond to mode $\ell(k)$ with generalized uncertainty vector $e(k)$.

Example 5. *Consider again the production system as in example 2, where the mode switching uncertainty is already presented. Following the same procedure as in example 4, we obtain the system matrices for all three recipes including the parametric uncertainty (where the system matrices for recipe A have been calculated earlier):*

$$\begin{aligned} A_1(e_1(k)) &= \begin{bmatrix} d_1(k-1) & \varepsilon & \varepsilon \\ d_1(k-1) + d_1(k) & d_2(k-1) & \varepsilon \\ d_1(k-1) + d_1(k) + d_2(k) & d_2(k-1) + d_2(k) & d_3(k-1) \end{bmatrix} \\ B_1(e_1(k)) &= \begin{bmatrix} 0 \\ d_1(k) \\ d_1(k) + d_2(k) \end{bmatrix} & C_1(e_1(k)) &= \begin{bmatrix} \varepsilon & \varepsilon & d_3(k) \end{bmatrix} \\ A_2(e_2(k)) &= \begin{bmatrix} d_1(k-1) & d_2(k-1) + d_2(k) & \varepsilon \\ \varepsilon & d_2(k-1) & \varepsilon \\ d_1(k-1) + d_1(k) & d_1(k) + d_2(k-1) + d_2(k) & d_3(k-1) \end{bmatrix} \\ B_2(e_2(k)) &= \begin{bmatrix} d_2(k) \\ 0 \\ d_1(k) + d_2(k) \end{bmatrix} & C_2(e_2(k)) &= \begin{bmatrix} \varepsilon & \varepsilon & d_3(k) \end{bmatrix} \\ A_3(e_3(k)) &= \begin{bmatrix} d_1(k-1) & \varepsilon & \varepsilon \\ d_1(k-1) + d_1(k) + d_3(k) & d_2(k-1) & d_3(k-1) + d_3(k) \\ d_1(k-1) + d_1(k) & \varepsilon & d_3(k-1) \end{bmatrix} \\ B_3(e_3(k)) &= \begin{bmatrix} 0 \\ d_1(k) + d_3(k) \\ d_1(k) \end{bmatrix} & C_3(e_3(k)) &= \begin{bmatrix} \varepsilon & d_2(k) & \varepsilon \end{bmatrix} \end{aligned}$$

One may note that the uncertainty vector for all modes are equal, since they all share the same machines:

$$e(k) = \begin{bmatrix} d_1(k-1) \\ d_1(k) \\ d_2(k-1) \\ d_2(k) \\ d_3(k-1) \\ d_3(k) \end{bmatrix} = e_1(k) = e_2(k) = e_3(k)$$

2-6 Difference Between Uncertainties

To explain the difference between both uncertainties in a mathematical way, we first introduce the notion of matrix structure. If there is a structural difference between two matrices, we mean that the underlying dynamics are related differently. For an arbitrary matrix $Q \in \mathbb{R}_{\max}^{n \times m}$, the structure of that matrix is given by:

$$[P_s(Q)]_{(i,j)} := \begin{cases} \varepsilon & \text{if } q_{(i,j)} = \varepsilon \\ e & \text{if } q_{(i,j)} \neq \varepsilon \end{cases}$$

With the previous extensions, we have two types of uncertainties in the system. We want to emphasize the difference between both uncertainties once again. First, we define a system matrix for mode ℓ , which has the following form:

$$Q_\ell := \begin{bmatrix} A_\ell & B_\ell & C_\ell^T \end{bmatrix}$$

This matrix makes no sense in practice, it is merely a theoretical way to unite the system matrices A_ℓ , B_ℓ and C_ℓ into a single matrix, so we can analyze its structure.

Consider two arbitrary representations of a system, called Q_i and Q_j . Now we have the following cases:

- If Q_i is equal to Q_j for all i, j , there is no uncertainty.
- If Q_i is not equal to Q_j for some i, j , we know that there is some uncertainty present. We can distinguish between two cases of uncertainty:
 - There is a structural difference between Q_i and Q_j for some i, j . This means that the underlying dynamics of the systems Q_i and Q_j are different, the system is in a different mode for Q_i than for Q_j . Thus we are dealing with mode switching uncertainty.
 - There is no structural difference between Q_i and Q_j for some i, j . This means that the underlying dynamics of the systems Q_i and Q_j are equal, the system is in the same mode for Q_i as for Q_j . However, since Q_i and Q_j are two different representations of the same stochastic mode, we are dealing with parametric uncertainty.

2-7 Conclusions

The basic operations for a MPL system are addition of event times and maximization of processing times. Such a MPL is a DES that is linear in the MPA. In the MPA, the conventional additive and multiplicative operators are replaced with maximization and addition respectively.

We have seen that we can extend the current classes of MPL systems to include both (stochastic) parametric uncertainty and mode switching uncertainty. First, we start with a deterministic MPL system, then we include the mode switching uncertainty to define a MPL system

with different modes. After that, we include the parametric uncertainty for each mode independently. This approach allows us to use the theory already analyzed in the current literature. Some extensive examples are presented through the chapter.

Such a system, with both types of uncertainties, will be called a Stochastic Switching Max-Plus-Linear (SSMPL) system, and can be described by:

$$x(k) = A_{\ell(k)}(e(k)) \otimes x(k-1) \oplus B_{\ell(k)}(e(k)) \otimes u(k) \quad (2-6)$$

$$y(k) = C_{\ell(k)}(e(k)) \otimes x(k) \quad (2-7)$$

where $A_{\ell(k)}(e(k))$, $B_{\ell(k)}(e(k))$ and $C_{\ell(k)}(e(k))$ are the system matrices that correspond to mode $\ell(k)$ with generalized uncertainty vector $e(k)$.

At last, to identify the different uncertainties, we investigate the ‘structure’ of the system matrices. This allows us to investigate the type of uncertainty by observing how the system matrices change after each events.

Chapter 3

Model Predictive Control

What would be a model without a controller closing the loop. This chapter deals with the application of the principles of Model Predictive Control (MPC) to our new class of Max-Plus-Linear (MPL) systems.

3-1 Why Model Predictive Control?

MPC is a well-known, advanced controller approach used in the process industry. MPC relies on a dynamic model of the process and is capable of handling constraints on inputs or outputs in a systematic way. Conventional MPC uses linear or nonlinear discrete time models. However, MPC has also been extended to Discrete Event Systems (DES) due to its attractive features [3].

Other control techniques, such as LQG , H_2 , H_∞ , etc. yield optimal controllers or control input sequences for the entire future evolution of the system. However, extending these methods to include additional constraints on the inputs and outputs is not easy. An important advantage of MPC is that the use of a finite horizon allows the inclusion of such additional constraints. Furthermore, MPC can handle structural changes, such as sensor or actuator failures, and changes in the system parameters or system structure.

3-2 Basic Ingredients

MPL is more a methodology, which makes it applicable to various types of systems. MPC has five basic ingredients [13, 14]:

Prediction Model: As the name suggests, we will obtain the optimal control input by making a prediction. In order to make a good prediction, we need a model of the system that needs to be controlled.

Cost Function: We need a mathematical method to describe which system behavior is good and which is bad. It is in general a measure of the error between a given reference signal and the output of the system. This function is sometimes called the performance index.

Constraint Handling: Most control techniques yield optimal control signals for the entire evolution of the system. The power of MPC is that it is fairly easy to incorporate constraints on the input and outputs of the system.

Optimization: We need to find the optimal control input by minimizing the cost function. The constraints can easily be applied to the optimization algorithm.

Receding Horizon Principle: At every discrete step, the controller finds the optimal control signal. Only the first element of that signal is sent to the system and updated measurements are used to find the optimal control signal for the next step. The remainder of the control signal is not used, since the updated information yields another optimal control signal.

3-3 Prediction Model

We need to find a relation between the predicted output of the system and the future input variables of the system, which can be obtained using a recursive relation between them.

In MPC, a prediction horizon is introduced. The length of the prediction is denoted by N_p . So, the controller is only able to look N_p steps ahead. If we choose N_p too low, we may miss some crucial dynamics of the model. If we choose N_p too large, the problem becomes very large and thus practically unsolvable.

The steps in obtaining the prediction are fairly straightforward. Therefore, we introduce a shorthand notation for our model in this section:

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

where $\bar{A}(k) := A_{\ell(k)}(e(k))$, $\bar{B}(k) := B_{\ell(k)}(e(k))$ and $\bar{C}(k) := C_{\ell(k)}(e(k))$. This is allowed, since we have a generalized uncertainty vector $e(k)$ for all modes ℓ .

Now, we want to obtain a future value of the output, given all information up to event k . For this, we use successive substitution of the above equations, until we reach the known parameters at step k , which are: $x(k-1)$, $u(k)$, $u(k+1)$, etc.:

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

$$\begin{aligned}
y(k+2) &= \bar{C}(k+2) \otimes \bar{A}(k+2) \otimes \bar{A}(k+1) \otimes \bar{A}(k) \otimes x(k-1) \\
&\oplus \bar{C}(k+2) \otimes \bar{A}(k+2) \otimes \bar{A}(k+1) \otimes \bar{B}(k) \otimes u(k) \\
&\oplus \bar{C}(k+2) \otimes \bar{A}(k+2) \otimes \bar{B}(k+1) \otimes u(k+1) \\
&\oplus \bar{C}(k+2) \otimes \bar{B}(k+2) \otimes u(k+2)
\end{aligned}$$

Let us first discuss some short notation that is used in the next formulas. If we write $A(k+j) \otimes \dots \otimes A(k+i)$, this is equivalent to the following:

$$A(k+j) \otimes A(k+j-1) \otimes \dots \otimes A(k+i+1) \otimes A(k+i)$$

So, we start with $A(k+j)$, then we lower the index of the matrix by one, to get $A(k+j-1)$. This index is further lowered until we reach $A(k+i)$. So, it is a requirement that $j \geq i$. If $j < i$, the matrix is equal to the identity matrix, and thus can be omitted. In the case where $j = i$, the matrix is just identical to $A(k+j) = A(k+i)$.

One should see a pattern arise:

$$\begin{aligned}
y(k+j) &= \bar{C}(k+j) \otimes \bar{A}(k+j) \otimes \dots \otimes \bar{A}(k) \otimes x(k-1) \\
&\oplus \bar{C}(k+j) \otimes \bar{A}(k+j) \otimes \dots \otimes \bar{A}(k+1) \otimes \bar{B}(k) \otimes u(k) \\
&\oplus \bar{C}(k+j) \otimes \bar{A}(k+j) \otimes \dots \otimes \bar{A}(k+2) \otimes \bar{B}(k+1) \otimes u(k+1) \\
&\oplus \dots \\
&\oplus \bar{C}(k+j) \otimes \bar{A}(k+j) \otimes \bar{B}(k+j-1) \otimes u(k+j-1) \\
&\oplus \bar{C}(k+j) \otimes \bar{B}(k+j) \otimes u(k+j)
\end{aligned}$$

Or even shorter:

$$\begin{aligned}
y(k+j) &= \bar{C}(k+j) \otimes \bar{A}(k+j) \otimes \dots \otimes \bar{A}(k) \otimes x(k-1) \\
&\oplus \bar{C}(k+j) \otimes \bigoplus_{i=0}^{i=j} \bar{A}(k+j) \otimes \dots \otimes \bar{A}(k+i+1) \otimes \bar{B}(k+i) \otimes u(k+i)
\end{aligned}$$

If we write this result in full form, we obtain:

$$\begin{aligned}
y(k+j) &= C_{\ell(k+j)}(e(k+j)) \otimes A_{\ell(k+j)}(e(k+j)) \otimes \dots \otimes A_{\ell(k)}(e(k)) \otimes x(k-1) \\
&\oplus C_{\ell(k+j)}(e(k+j)) \otimes \bigoplus_{i=0}^{i=j} \left[A_{\ell(k+j)}(e(k+j)) \otimes \dots \otimes A_{\ell(k+i+1)}(e(k+i+1)) \right. \\
&\quad \left. \otimes B_{\ell(k+i)}(e(k+i)) \otimes u(k+i) \right]
\end{aligned}$$

Let us now denote the vector that contains all future predictions of the output of the system by:

$$\tilde{y}(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k+N_p-1) \end{bmatrix}$$

One may observe that we need all future values of $u(k)$, $\ell(k)$ and $e(k)$. We can capture all these future values in vectors as well:

$$\tilde{u}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k + N_p - 1) \end{bmatrix} \quad \tilde{\ell}(k) = \begin{bmatrix} \ell(k) \\ \vdots \\ \ell(k + N_p - 1) \end{bmatrix} \quad \tilde{e}(k) = \begin{bmatrix} e(k) \\ \vdots \\ e(k + N_p - 1) \end{bmatrix}$$

Remark 4. Here again, it is very likely that $\tilde{e}(k)$ contains duplicate representations of some stochastic variables. In practice, we will again apply the vector-based union operator, and work with $\tilde{e}_{\cup}(k)$ instead. Once more, we assume in the remainder of this report that the uncertainty vector $\tilde{e}(k)$ contains no duplicate entries.

With these newly defined vectors, we can rewrite the predictor $\tilde{y}(k)$ in matrix notation. Observe that the predictor is still a function of future values of $\ell(k)$ and $e(k)$, which are still unknown. The result is:

$$\tilde{y}(k) = \tilde{y}(\tilde{\ell}(k), \tilde{e}(k)) = \tilde{C}(\tilde{\ell}(k), \tilde{e}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k), \tilde{e}(k)) \otimes \tilde{u}(k)$$

where $\tilde{C}(\tilde{\ell}(k), \tilde{e}(k))$ and $\tilde{D}(\tilde{\ell}(k), \tilde{e}(k))$ are defined as:

$$\tilde{C}(\tilde{\ell}(k), \tilde{e}(k)) = \begin{bmatrix} \tilde{C}_{(1)}(\tilde{\ell}(k), \tilde{e}(k)) \\ \vdots \\ \tilde{C}_{(N_p)}(\tilde{\ell}(k), \tilde{e}(k)) \end{bmatrix}$$

$$\tilde{D}(\tilde{\ell}(k), \tilde{e}(k)) = \begin{bmatrix} \tilde{D}_{(1,1)}(\tilde{\ell}(k), \tilde{e}(k)) & \cdots & \mathcal{E} \\ \vdots & \ddots & \vdots \\ \tilde{D}_{(N_p,1)}(\tilde{\ell}(k), \tilde{e}(k)) & \cdots & \tilde{D}_{(N_p,N_p)}(\tilde{\ell}(k), \tilde{e}(k)) \end{bmatrix}$$

with:

$$\begin{aligned} \tilde{C}_{(m)}(\tilde{\ell}(k), \tilde{e}(k)) &= C_{\ell(k+m-1)}(e(k+m-1)) \otimes A_{\ell(k+m-1)}(e(k+m-1)) \otimes \dots \otimes A_{\ell(k)}(e(k)) \\ \tilde{D}_{(m,n)}(\tilde{\ell}(k), \tilde{e}(k)) &= C_{\ell(k+j-1)}(e(k+j-1)) \\ &\quad \otimes A_{\ell(k+j-1)}(e(k+j-1)) \otimes \dots \otimes A_{\ell(k+n)}(e(k+n)) \otimes B_{\ell(k+n-1)}(e(k+n-1)) \end{aligned}$$

Please note that $\tilde{D}(\tilde{\ell}(k), \tilde{e}(k))$ is a lower triangular matrix, since the prediction at event step $k+j$ does not depend on the input at event step larger than $k+j$. In Max-Plus Algebra (MPA), the zero-matrix $\mathcal{E}$ is a matrix where all the elements of that matrix are equal to ε .

3-4 Cost Function

In the previous section, we have found a mathematical framework to express the predicted output $\tilde{y}(k)$. Now, we want to compare this signal with a given reference signal $r(k)$. It is obvious that we want to compare $y(k+j)$ with $r(k+j)$, for all j in the prediction horizon. Therefore, we introduce another vector that contains the future values of $r(k)$:

$$\tilde{r}(k) = \begin{bmatrix} r(k) \\ \vdots \\ r(k + N_p - 1) \end{bmatrix}$$

The cost function in general consists of two parts; an output cost function, that represents a measure of the error between the predicted output and a given reference signal, and an input cost function, which allows us to penalize the input effort as well. The ratio between both cost functions can be determined with the variable λ . The cost function then becomes:

$$J(k) = J_{\text{out}}(k) + \lambda J_{\text{in}}(k)$$

The most used output cost function is:

$$\begin{aligned} J_{\text{out}}(k) &= \sum_{j=0}^{N_p-1} \sum_{i=1}^{n_y} \max(y_i(k+j) - r_i(k+j), 0) \\ &= \sum_{i=1}^{N_p n_y} \max(\tilde{y}_i(k) - \tilde{r}_i(k), 0) \end{aligned}$$

where n_y is the number of outputs in our model. If we have multiple outputs in our model, this means that we calculate, for each prediction step and output separately, the maximum delay. Then we add all these delays to obtain the output cost function. We only investigate delays, since the maximization is bounded from below by the zero-term. In practice, minimizing this term means that only delays are penalized. The term $\kappa(k) = \max(y_i(k) - r_i(k), 0)$ in the above equation is called the tardiness error, for the i th input at event step k .

The most used input cost function is:

$$\begin{aligned} J_{\text{in}}(k) &= - \sum_{j=0}^{N_p-1} \sum_{i=1}^{n_u} u_i(k+j) \\ &= - \sum_{i=1}^{N_p n_u} \tilde{u}(k) \end{aligned}$$

where n_u is the number of inputs in our model. If we have multiple inputs, we just add all the inputs for every prediction step, and then add all these sums over the whole prediction horizon. In practice, minimizing this function means that we maximize the time instances at which the future products enter the system.

Remark 5. *Note that the cost function can be defined in other forms as well. The choice depends on the problem that needs to be solved. However, as we will see later, this particular cost function has the property, under some assumptions, of being convex in the input signal $\tilde{u}(k)$. This which allows us to calculate the optimal solution in an efficient way, by using the convex optimization algorithms.*

Since the future values of $\ell(k)$ and $e(k)$ are not known, we cannot compute this cost function directly. However, if we assume that both uncertainties are stochastic random variables, we can compute the expectation of the cost function. These random variables occur in the computation of the predicted output only, therefore we use a modified version of the output

cost function $J_{\text{out},\mathbb{E}}(k)$, which is defined as:

$$\begin{aligned} J_{\text{out},\mathbb{E}}(k) &= \sum_{j=0}^{N_p-1} \sum_{i=1}^{n_y} \mathbb{E} [\max(y_i(k+j) - r_i(k+j), 0)] \\ &= \sum_{i=1}^{N_p n_y} \mathbb{E} [\max(\tilde{y}_i(k) - \tilde{r}_i(k), 0)] \end{aligned}$$

3-5 Constraint Handling

The power of MPC is that it uses some optimization techniques, and therefore, it is fairly simple to impose some constraints as well. In general, there are two different kinds of constraints applied to the optimization problem.

The first one has a practical background. In theory, it would be possible that the input events for the $(k+1)$ th event occur before the k th event. In fact, it does not matter for the output events, since the output still depends on the internal state events. If we, for example, look at a production plant, a machine can start on batch k as soon as it has finished its previous batch $(k-1)$. So, if the material for batch $k+1$ arrives at the machine earlier than the one for batch k , it still has to wait for the material for batch k . The machine will not start with the batch that arrives earlier, in this example the material for batch k .

In practice, it would be impossible to interchange the input event times. Therefore, the following (inequality) constraint is added:

$$\Delta u(k+j) \geq 0 \quad \text{for } j = 0, \dots, N_p - 1$$

where the Δ -operator is defined as: $\Delta u(k) = u(k) - u(k-1)$. It can be compared with a scaled version of the discrete derivative operator. Since all future input event times should be larger than the previous one, we have a non-strictly increasing input signal $\tilde{u}(k)$ for the prediction. Since, for $j = 0$, this constraint takes $u(k-1)$ into account as well, the global input sequence $u(k)$ (after closing the loop) will be a non-strictly increasing input signal as well.

The second one has a more theoretical background. The prediction horizon should be sufficiently large, so that the important dynamics of the system are present in the prediction. Choosing N_p too large will make the problem hard to solve. Since the variable N_p represents the number of free variables used in the optimization problem, the difficulty of the problem grows as N_p becomes larger.

Therefore, we introduce a control horizon, denoted by N_c . We assume that the control signal $\tilde{u}(k)$ cannot be changed after this control horizon; i.e. it is assumed to be constant. This will reduce the number of variables in the optimization problem. Thus, we add the following (equality) constraint as well:

$$\Delta^2 u(k+j) = 0 \quad \text{for } j = N_c, \dots, N_p - 1$$

where the Δ^2 -operator is defined as: $\Delta^2 u(k) = \Delta u(k) - \Delta u(k-1)$. With substitution of the Δ -operator, we obtain: $\Delta^2 u(k) = u(k) - 2u(k-1) + u(k-2)$. It can be compared with a

scaled version of the discrete second derivative operator. One may note that if $N_p = N_c$, the constraint is not applied. In this case, there is no reduction of the number of variables for the optimization.

An additional third (inequality) constraint can be added as well.

$$A_{\text{con}}\tilde{u}(k) + B_{\text{con}}\mathbb{E}[\tilde{y}(k)] \leq c_{\text{con}}(k)$$

Since $\tilde{y}$ is a stochastic variable, we need to compute the expectation of $\tilde{y}$. Here, A_{con} and B_{con} are matrices, which represents the linear constraints on the evolution of the input and output signal.

3-6 Optimization

Now that we have defined the optimization function (cost function), and the constraints, we can define the (nonlinear) optimization problem:

$$\min_{\tilde{u}(k)} J(k) = \min_{\tilde{u}(k)} \sum_{i=1}^{N_p n_y} \mathbb{E}[\max(\tilde{y}_i(k) - \tilde{r}_i(k), 0)] - \lambda \sum_{i=1}^{N_p n_u} \tilde{u}(k)$$

$$\begin{aligned} \text{s.t.} \quad & \Delta u(k+j) \geq 0 & \text{for } j = 0, \dots, N_p - 1 \\ & \Delta^2 u(k+j) = 0 & \text{for } j = N_c, \dots, N_p - 1 \\ & A_{\text{con}}\tilde{u}(k) + B_{\text{con}}\mathbb{E}[\tilde{y}(k)] \leq c_{\text{con}}(k) \end{aligned}$$

This optimization problem can be solved in various ways. However, it is beyond the scope of this MSc research project to find an optimal (non-linear) optimization algorithm.

3-7 Receding Horizon Principle – Timing Aspects

Now that we have defined how the controller works (with the optimization problem), it is time to interconnect it with the real system. For conventional time-driven systems, as soon as the controller obtains some new information of the system¹ (usually a new value of $y(k)$), the controller computes the optimal control signal. Then, we apply the first element of this signal to the real system. The time counter k is shifted in time, and the same procedure starts all over again.

However, for DES, this is not the case. In the most ideal situation, we could apply the same principles. So, we first compute the occurrence times of the first event. Then we compute the optimal control signal, and apply this signal to the real system. However, the system is defined in the event domain, while the controller works in the conventional time domain. Since the system vectors $x(k)$, $u(k)$ and $y(k)$ contains time information about the k th event, it is very likely that some elements of these system vectors are in the past, while others are

¹ Assuming that the controller is able to compute the optimal control signal before it obtains new information about the system.

in the future. The events that already occurred, cannot be changed anymore. This limits the controller in obtaining the optimal control signal for the future events.

In the most general case, it is assumed that this problem can be solved by applying some hard constraints on the events that already occurred in the past. This allows the controller to obtain the optimal control signal, subject to the events that cannot be changed anymore.

3-8 Interconnections and Implementation

Now that we have defined the basic building blocks, we need to interconnect them. This process is visualized in figure 3-1.

The goal of the Model Predictive Controller will be to obtain an optimal input signal $\tilde{u}(k)$ that minimizes the cost function $J(k)$ subject to some constraints. The optimizer starts with an initial $\tilde{u}_0(k)$, evaluates the cost function and its gradient for $\tilde{u}_0(k)$. Using this information, it obtains a better estimate of the $\tilde{u}(k)$ that minimizes $J(k)$. If the optimizer has found the optimal control signal $\tilde{u}(k)$, the first element is passed on to the real system. As soon as new information is available, the controller starts the same procedure all over again, with the updated information (Receding Horizon Principle).

3-9 Conclusions

In this chapter, we have introduced the basic properties of MPC, and how we can apply them to our new class of MPL systems. First, we need a predictor model, to compute the future output signal, as a function of the future input signal, the current state and both uncertainties. After that, we can use this future output signal to compute the cost function. This functions expresses the optimality of the system behavior into a computable value. Since we are dealing with two types of random variables, we must use the expectation in order to get a computable expression.

An optimization algorithm will try to find the optimal solution of this cost function, subject to some constraints. These constraints are required for DES. Once the controller has found the optimal solution, the loop is closed and the first element of this solution is send to the system. The controller retrieves new information about the system (usually a new event that occurred), and recalculates the next optimal solution.

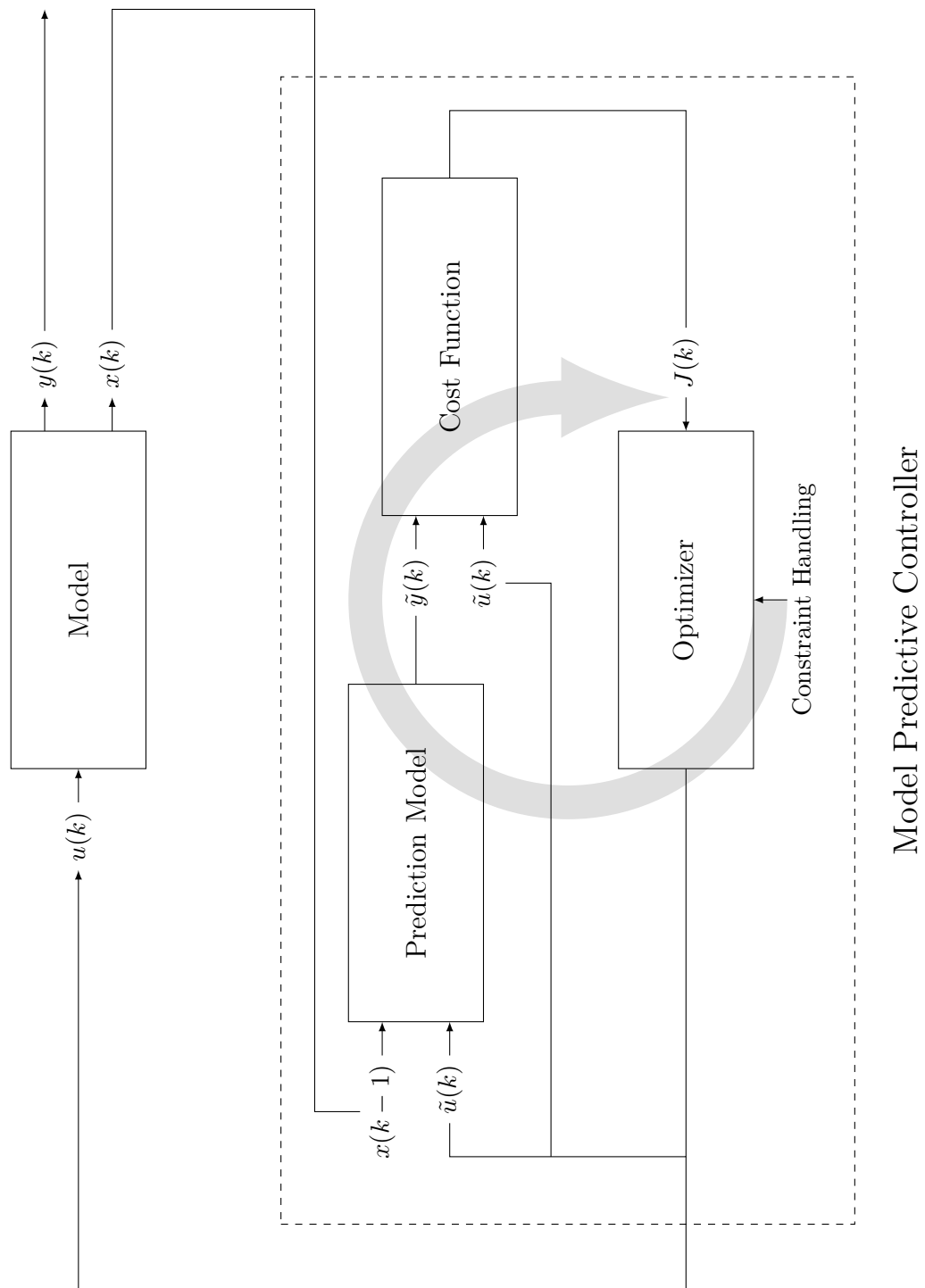


Figure 3-1: Interconnections in the Controller.

Chapter 4

Cost Function

In the previous chapter, we have found that computing the cost function requires the evaluation of the expectation over two types of random variables. This section shows how we can compute this expectation, for various conditions.

4-1 Joint Density Function

In order to find the optimal control signal $\tilde{u}(k)$, we need to express $J(k)$ as a function of $\tilde{u}(k)$. The cost function was defined as:

$$\begin{aligned} J(k) &= J_{\text{out},\mathbb{E}}(k) + \lambda J_{\text{in}}(k) \\ &= \sum_{i=1}^{N_p n_y} \mathbb{E}[\max(\tilde{y}_i(\tilde{\ell}(k), \tilde{e}(k)) - \tilde{r}_i(k), 0)] - \lambda \sum_{i=1}^{N_p n_u} \tilde{u}(k) \end{aligned}$$

Here, we see that we need to compute the expectation over two stochastic random variables, i.e., the mode switching uncertainty $\tilde{\ell}(k)$ and the parametric uncertainty $\tilde{e}(k)$. Therefore, we need to define the joint density function (jdf) of both stochastic random variables. Here, a problem arises:

A Stochastic Switching Max-Plus-Linear (SSMPL) system contains two types of random variables, namely:

- The mode switching variable $\tilde{\ell}(k)$. Since there is a finite number of modes n_L , there is only a finite number of possible mode switching sequence vectors $\tilde{\ell}(k)$ over the prediction horizon. This makes $\tilde{\ell}(k)$ a discrete random variable.
- The uncertainty vector $\tilde{e}(k)$. Since the uncertainty in the system is related to time, this uncertainty can take any value. This value is of course limited to the support (i.e., where the function is not equal to zero) given by the probability density function (pdf). This makes $\tilde{e}(k)$ a continuous random variable.

In order to compute the expectation of the output cost function, we need to obtain an expression for the joint probability density function. First, we introduce $\mathcal{L}(k)$, which is the (discrete) sample space of all the mode switching sequence vectors $\tilde{\ell}(k)$, and $\mathcal{E}(k)$, which is the (continuous) sample space of the uncertainty $\tilde{e}(k)$ of the prediction of $\tilde{y}(k)$. In order to compute the jdf of $\tilde{\ell}(k)$ and $\tilde{e}(k)$, we use the theory of conditional probabilities [15]:

$$f_{\mathcal{L}(k), \mathcal{E}(k)}(\tilde{\ell}(k), \tilde{e}(k)) = f_{\mathcal{E}(k)}(\tilde{e}(k)) \tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)]$$

where $f_{\mathcal{E}(k)}(\tilde{e}(k))$ is the continuous pdf of $\tilde{e}(k)$ that we have obtained during the modeling phase, and $\tilde{P}[\mathcal{L} = \tilde{\ell}(k) \mid \mathcal{E} = \tilde{e}(k)]$ is the probability that we have mode switching sequence $\tilde{\ell}(k)$, given the uncertainty $\tilde{e}(k)$.

Now that we have obtained an expression for the jdf, we can define an expression for the expectation of the (output) cost function. Since $\tilde{\ell}(k)$ is a discrete random variable, we need to add up all the $\tilde{\ell}(k) \in \mathcal{L}(k)$, and since $\tilde{e}(k)$ is a continuous random variable, we need to integrate over all $d\tilde{e}(k) \in \mathcal{E}(k)$:

$$\begin{aligned} J_{\text{out}, \mathbb{E}}(k) &= \mathbb{E}[J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k))] \\ &= \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\int_{\mathcal{E}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{L}(k), \mathcal{E}(k)}(\tilde{\ell}(k), \tilde{e}(k)) d\tilde{e}(k) \right] \\ &= \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\int_{\mathcal{E}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] d\tilde{e}(k) \right] \end{aligned}$$

4-2 Switching Probability

As stated before, it is assumed that the pdf of $\tilde{e}(k)$ does not depend on the mode switching sequence $\tilde{\ell}(k)$. We have defined $\tilde{e}(k)$ such that the cost function $J(k)$ for every mode switching sequence can be written as a function of the same $\tilde{e}(k)$. This means that, for every mode switching sequence $\tilde{\ell}(k)$, we have the same uncertainty vector $\tilde{e}(k)$ and thus is independent of the mode switching sequence $\tilde{\ell}(k)$.

However, in the most general case, the mode switching mechanism may depend on the uncertainty vector $\tilde{e}(k)$. In the current literature [7–9], it is stated that the mode switching mechanism at event k may depend on the previous mode $\ell(k-1)$, the previous state $x(k-1)$, the input signal $u(k)$ and a (auxiliary) control signal $v(k)$. We can write this as:

$$P[L(k) = \ell(k) \mid \ell(k-1), x(k-1), u(k), v(k)]$$

where $L(k)$ is again the stochastic random variable and $\ell(k)$ is its value. This means that we can compute the probability of mode $\ell(k)$ to occur, if we know the previous mode $\ell(k-1)$, the previous state $x(k-1)$, the control signal $u(k)$ and a (auxiliary) control signal $v(k)$.

As Model Predictive Control (MPC) methodology suggests, we need to compute the probability of $\tilde{\ell}(k)$. Naturally, we would like to write for the ‘predicted’ probability of $\tilde{\ell}(k)$:

$$\begin{aligned} \tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \ell(k-1), x(k-1), \tilde{u}(k), \tilde{v}(k)] \\ = \prod_{j=0}^{N_p-1} P[L(k+j) = \ell(k+j) \mid \ell(k+j-1), x(k+j-1), u(k+j), v(k+j)] \end{aligned}$$

One may observe that we need the following variables available for computation:

- $\ell(k-1), \ell(k), \dots, \ell(k+N_p-2)$
- $x(k-1), x(k), \dots, x(k+N_p-2)$
- $u(k), \dots, u(k+N_p-1)$
- $v(k), \dots, v(k+N_p-1)$

Remark 6. *The main reason that we do not need the terms $\ell(k+N_p-1)$ and $x(k+N_p-1)$ is that these terms only influence the next mode $\ell(k+N_p)$, which is not included in the prediction horizon.*

Some of these variables are known at event step k , such as $\ell(k-1)$ and $x(k-1)$. Further we assume that we know $\tilde{u}(k)$ and $\tilde{v}(k)$. In reality, these variables are not known in advance. However, the algorithm that computes the minimum value of the cost function proposes ‘fixed’ values for $\tilde{u}(k)$ and $\tilde{v}(k)$. This means, that in the evaluation of the expectation of the output cost function, $\tilde{u}(k)$ and $\tilde{v}(k)$ are fixed and known in advance. Further, by definition, we compute the probability for a given $\tilde{\ell}(k)$, so these values are known as well when determining the ‘predicted’ probability. The only terms that are still unknown are the future values of the state. We will deal with this case in the next section.

4-3 Classification I – Mode Switching Behavior

First, we will introduce a new type of classification, based on the mode switching behavior of the system. In general, we can distinguish two types of mode switching, deterministic and stochastic mode switching. We want to investigate the difference between these two cases, since it imposes assumptions on the computation of the expectation of the cost function.

4-3-1 Deterministic Mode Switching

In the most simplest form, we have deterministic mode switching. This means that, if the parametric uncertainty $\tilde{e}(k)$ in the system is known, there is only one possible mode switching sequence. This means that we have:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \dots] \in \{0, 1\}$$

where the set $\{0, 1\}$ means that the switching probability $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$ can only take the value zero or one. Thus, if one of these probabilities is equal to one, we know that all other probabilities must be equal to zero. This means that we have only one possible next mode, which is referred as deterministic switching.

4-3-2 Stochastic Mode Switching

In the most general case, we have stochastic switching. This means, that if the parametric uncertainty $\tilde{e}(k)$ in the system is known, there are finite many mode switching sequences, each with its own probability. This means that we have:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \dots] \in [0, 1]$$

where the interval $[0, 1]$ means that the switching probability $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$ can take any value between zero and one, including zero and one.

To distinguish this case from the previous (deterministic) case, we assume that there are at least two¹ mode switching sequences $\tilde{\ell}_i(k)$ and $\tilde{\ell}_j(k)$ for which the switching probabilities $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}_i(k)]$ and $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}_j(k)]$ are not equal to either zero or one. If all switching probabilities are in the set $\{0, 1\}$, it can be categorized as deterministic mode switching.

So, we know that there are at least two probabilities that are not in the set $\{0, 1\}$. This means that there are multiple modes that the system can switch to. Thus, there are finite many mode switching sequences.

4-4 Classification II – Dependency between Random Variables

In the next subsections, we will discuss various cases in which we analyze the dependency between the mode switching uncertainty $\tilde{\ell}(k)$ and the parametric uncertainty $\tilde{e}(k)$. We are in particular interested under which conditions the probability of the occurrence of a particular mode switching sequence, is or is not dependent on the parametric uncertainty $\tilde{e}(k)$ in the system.

4-4-1 The Previous Mode $\ell(k - 1)$

If the mode switching probability depends on the previous mode, we have:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \ell(k - 1)] = \prod_{j=0}^{N_p-1} \tilde{P}[L(k + j) = \ell(k + j) \mid \ell(k + j - 1)]$$

One may note that, if both the previous mode $\ell(k - 1)$, the future modes $\ell(k + j)$ and all intermediate probabilities $P[L(k + j) = \ell(k + j) \mid \ell(k + j - 1)]$ are known for $j = 0, \dots, N_p - 1$, the probability for a particular mode switching sequence vector $\tilde{\ell}(k)$ is deterministic and known. This sequence does not depend on the parametric uncertainty $\tilde{e}(k)$, and is therefore omitted in the definition of $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \ell(k - 1)]$. Thus, we have two independent stochastic random variables.

¹There cannot be a single one. Assume that there exists a single mode switching sequence $\tilde{\ell}_i(k)$ with switching probability $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}_i(k)]$ equal to a value between zero and one (not including zero and one). All other mode switching sequences then must have a switching probability that is equal to zero. However, the sum of all probabilities cannot be equal to one, unless that single probability is equal to one. This is a contradiction.

4-4-2 The Previous State $x(k-1)$

If the mode switching probability depends on the previous state, we have:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid x(k-1), \tilde{e}(k)] = \prod_{j=0}^{N_p-1} P[L(k+j) = \ell(k+j) \mid x(k+j-1)]$$

Here, we observe that we need the future values of $x(k+j)$ for $k = 0, \dots, N_p-2$. These values however depend on both the mode switching variable $\tilde{\ell}(k)$ and the parametric uncertainty $\tilde{e}(k)$. In order to determine the predicted states, we use the same principle as obtained in section 3-3, where we computed the predictor matrices.

We can write $x(k+j)$ for $j = 0, \dots, N_p-2$ as:

$$\begin{aligned} x(k+j) &= A_{\ell(k+j)}(e(k+j)) \otimes \dots \otimes A_{\ell(k)}(e(k)) \otimes x(k-1) \\ &\oplus \bigoplus_{i=0}^{i=j} \left[A_{\ell(k+j)}(e(k+j)) \otimes \dots \otimes A_{\ell(k+i+1)}(e(k+i+1)) \otimes B_{\ell(k+i)}(e(k+i)) \otimes u(k+i) \right] \end{aligned}$$

If we now define the predicted states as:

$$\tilde{x}(k) = \begin{bmatrix} x(k) \\ \vdots \\ x(k+N_p-1) \end{bmatrix}$$

we can write the predicted states in the matrix form:

$$\tilde{x}(k) = \tilde{x}(\tilde{\ell}(k), \tilde{e}(k)) = \tilde{A}(\tilde{\ell}(k), \tilde{e}(k)) \otimes x(k-1) \oplus \tilde{B}(\tilde{\ell}(k), \tilde{e}(k)) \otimes \tilde{u}(k)$$

where $\tilde{A}(\tilde{\ell}(k), \tilde{e}(k))$ and $\tilde{B}(\tilde{\ell}(k), \tilde{e}(k))$ are defined as:

$$\begin{aligned} \tilde{A}(\tilde{\ell}(k), \tilde{e}(k)) &= \begin{bmatrix} \tilde{A}_{(1)}(\tilde{\ell}(k), \tilde{e}(k)) \\ \vdots \\ \tilde{A}_{(N_p)}(\tilde{\ell}(k), \tilde{e}(k)) \end{bmatrix} \\ \tilde{B}(\tilde{\ell}(k), \tilde{e}(k)) &= \begin{bmatrix} \tilde{B}_{(1,1)}(\tilde{\ell}(k), \tilde{e}(k)) & \dots & \varepsilon \\ \vdots & \ddots & \vdots \\ \tilde{B}_{(N_p,1)}(\tilde{\ell}(k), \tilde{e}(k)) & \dots & \tilde{B}_{(N_p,N_p)}(\tilde{\ell}(k), \tilde{e}(k)) \end{bmatrix} \end{aligned}$$

with:

$$\begin{aligned} \tilde{A}_{(m)}(\tilde{\ell}(k), \tilde{e}(k)) &= A_{\ell(k+m-1)}(e(k+m-1)) \otimes \dots \otimes A_{\ell(k)}(e(k)) \\ \tilde{B}_{(m,n)}(\tilde{\ell}(k), \tilde{e}(k)) &= A_{\ell(k+j-1)}(e(k+j-1)) \otimes \dots \otimes A_{\ell(k+n)}(e(k+n)) \\ &\quad \otimes B_{\ell(k+n-1)}(e(k+n-1)) \end{aligned}$$

Remark 7. As noted before, we do not need the value of $x(k+N_p-1)$. This value is included in the above notation, since it simplifies the notation. In the computation, this term can be omitted. With this notation, the computation of $\tilde{x}(k)$ has a high similarity with the computation of $\tilde{y}(k)$ in section 3-3.

One may note that, to obtain the future states, we need both the mode switching uncertainty $\ell(k+j)$ and the parametric uncertainty $e(k+j)$ for $j = 0, \dots, N_p - 1$. Then, the probability for a particular mode switching sequence vector $\tilde{\ell}(k)$ is deterministic and known. This sequence thus does depend on the parametric uncertainty $\tilde{e}(k)$, and cannot be omitted in the definition of $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid x(k-1), \tilde{e}(k)]$. Thus, we have two dependent stochastic random variables.

4-4-3 The Input Signal $\tilde{u}(k)$

If the mode switching probability depends on the input signal, we have:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \tilde{u}(k)] = \prod_{j=0}^{N_p-1} P[L(k+j) = \ell(k+j) \mid u(k+j)]$$

One may note that, if the future inputs $u(k+j)$ (since we compute the expectation for a fixed control signal $\tilde{u}(k)$) and all intermediate probabilities $P[L(k+j) = \ell(k+j) \mid u(k+j)]$ are known for $j = 0, \dots, N_p - 1$, the probability for a particular mode switching sequence vector $\tilde{\ell}(k)$ is deterministic and known. This sequence does not depend on the parametric uncertainty $\tilde{e}(k)$, and is therefore omitted in the definition of $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \tilde{u}(k)]$. Thus, we have two independent stochastic random variables.

Remark 8. *One possible point of confusion is to say that the predictor $\tilde{y}(k)$ depends on $\tilde{u}(k)$, with the following relation:*

$$\tilde{y}(k) = \tilde{C}(\tilde{\ell}(k), \tilde{e}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k), \tilde{e}(k)) \otimes \tilde{u}(k)$$

Thus, we see that there is some stochastic relation between $\tilde{y}(k)$ and $\tilde{u}(k)$, or equivalently between $\tilde{x}(k)$ and $\tilde{u}(k)$. This relation is used in the computation of the output cost function, but is not present in the mode switching probability.

4-4-4 The (Auxiliary) Control Signal $\tilde{v}(k)$

If the mode switching probability depends on the (auxiliary) control signal, we have:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \tilde{v}(k)] = \prod_{j=0}^{N_p-1} P[L(k+j) = \ell(k+j) \mid v(k+j)]$$

One may note that, if the future inputs $v(k+j)$ (since we compute the expectation for a fixed control signal $\tilde{v}(k)$) and all intermediate probabilities $P[L(k+j) = \ell(k+j) \mid v(k+j)]$ are known for $j = 0, \dots, N_p - 1$, the probability for a particular mode switching sequence vector $\tilde{\ell}(k)$ is deterministic and known. This sequence does not depend on the parametric uncertainty $\tilde{e}(k)$, and is therefore omitted in the definition of $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \tilde{v}(k)]$. Thus, we have two independent stochastic random variables.

4-4-5 The General Case

Of course, we can extend the above in a more general case. If the mode switching probability depends on more than one variable², and one of these variables depends on the parametric uncertainty $\tilde{e}(k)$, then the mode switching sequence probability $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$ depends on the parametric uncertainty $\tilde{e}(k)$ as well. If none of the variables of the mode switching sequence probability depends on the parametric uncertainty $\tilde{e}(k)$, then the mode switching sequence probability $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$ does not depend on the parametric uncertainty $\tilde{e}(k)$.

We see that, if the mode switching probability depends on any (future) value of $\tilde{x}(k)$, then it depends on the parametric uncertainty $\tilde{e}(k)$ as well. There might be a case where the switching probability depends on the previous state in the first prediction step only. This means that we do not need to compute $\tilde{x}(k)$ to obtain the (future) values of the states, and thus we know that the switching probability does not depend on the parametric uncertainty $\tilde{e}(k)$.

Thus, in the general case, we have:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \ell(k-1), x(k-1), \tilde{u}(k), \tilde{v}(k), \tilde{e}(k)]$$

4-5 Assumptions that Reduce the Computational Complexity

In the previous sections, we have seen that we can distinguish four types of SSMPL systems. These four types are summarized in table 4-1. For the two cases of mode switching, the reader is referred to section 4-3. For the dependency of the random variables, the reader is referred to section 4-4.

	Deterministic Mode Switching	Stochastic Mode Switching
Independent Random Variables	Type 1	Type 2
Dependent Random Variables	Type 3	Type 4

Table 4-1: Four types of Stochastic Switching Max-Plus-Linear systems.

In the next subsections, we will see the effect of these assumptions on the computation of the expectation of the cost function. As a quick reminder, we need to evaluate the following expression in order to compute the expectation of the (output) cost function:

$$J_{\text{out}, \mathbb{E}}(k) = \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\int_{\mathcal{E}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] d\tilde{e}(k) \right]$$

We will assume that both $\ell(k-1)$ and $x(k-1)$ are known in event step k , they will be treated as initial conditions. Since the optimizer evaluates the cost function for given $\tilde{u}(k)$ and $\tilde{v}(k)$, we assume them to be known when we are evaluating the switching probability.

²Here, the variables might be: $\ell(k-1)$, $x(k-1)$, $\tilde{u}(k)$ or $\tilde{v}(k)$

4-5-1 Type 1: Independent Random Variables, Deterministic Switching

If we assume that both random variables are independent, then we know that:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] = \tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$$

Further, since we assume deterministic switching, there is only one possible mode switching sequence, with probability one:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)] = 1$$

Thus, we have:

$$\begin{aligned} J_{\text{out},\mathbb{E}}(k) &= \int_{\mathcal{E}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) d\tilde{e}(k) \\ &= \mathbb{E}[J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k))] \end{aligned}$$

Here, we can observe that the last equation is in fact equal to the expectation of the output cost function³ that we have found in [5]. The only difference is that here we have a different sequence of consecutive systems, where in [5], they use the same system for every prediction step.

As a practical example, one can consider a production system. Assume that we have multiple recipes, and that the order of processing these recipes is known. For example, we assume that we first produce 100 products with recipe A, then 50 products with recipe B, and continue until we reach the last recipe. Then, we start with 50 products of recipe A again, etc.

Another example can be for a railway system. Assume that we have two modes, one with the daytime schedule and one with the nighttime schedule. The system switches between both schedules, based on the current time.

Example 6. Assume that we have the following autonomous⁴ first order system with two modes:

$$x(k) = A_{\ell} \otimes x(k-1) \quad \text{where} \quad A_{\ell} = \begin{cases} 1 + e(k) & \text{for } \ell(k) = 1 \\ 2 + e(k) & \text{for } \ell(k) = 2 \end{cases}$$

with $e(k) \sim \mathcal{N}(0, 1)$ and initial conditions $x(k-1) = 0$ and $\ell(k-1) = 1$. We will use a prediction horizon $N_p = 2$.

Here, we assume that the model can only switch from the current mode to the next mode. Thus we have:

$$\begin{aligned} P[L(k) = 1 \mid \ell(k-1) = 1] &= 0 \\ P[L(k) = 2 \mid \ell(k-1) = 1] &= 1 \\ P[L(k) = 1 \mid \ell(k-1) = 2] &= 1 \\ P[L(k) = 2 \mid \ell(k-1) = 2] &= 0 \end{aligned}$$

³We will explain in the next chapter how we can compute this cost function.

⁴An autonomous system is a system without any inputs.

Please note that $P[L(k) = \ell(k) \mid \ell(k-1)] \in \{0, 1\}$ for all $\ell(k), \ell(k-1)$ and that all probabilities for each $\ell(k-1)$ add up to one, which means that we have deterministic switching.

This means that, since $\ell(k-1) = 1$, we have one single possible mode switching sequence⁵ at this stage, with probability one. All other mode switching sequences have zero probability. Thus:

$$\tilde{\ell}(k) = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Now, we can simply compute:

$$J_{\text{out},\mathbb{E}}(k) = \mathbb{E}[J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k))]$$

4-5-2 Type 2: Independent Random Variables, Stochastic Switching

If we assume that both random variables are independent, then we know that:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] = \tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$$

Since we have stochastic switching, we know that there are finitely many mode switching sequences, each with probability:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)] \in [0, 1]$$

Thus we have:

$$J_{\text{out},\mathbb{E}}(k) = \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\int_{\mathcal{E}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)] d\tilde{e}(k) \right]$$

Since the term $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$ does not depend on $\tilde{e}(k)$, it is constant inside the integral. Therefore, we can write it outside the integral:

$$\begin{aligned} J_{\text{out},\mathbb{E}}(k) &= \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)] \int_{\mathcal{E}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) d\tilde{e}(k) \right] \\ &= \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)] \mathbb{E}[J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k))] \right] \end{aligned}$$

Here, we can observe that the last equation is again a nice result. We can see that we first need to obtain all possible mode switching sequences $\tilde{\ell}(k) \in \mathcal{L}(k)$. Then, we can compute for each of these mode switching sequences the probability of that mode switching sequence to occur, and multiply it with the expectation of output cost function that corresponds to that particular mode switching sequence⁶. In this case, we can use the theory from [7–10] for the mode switching part and [5, 6] for the stochastic parametric uncertainty part.

⁵For $\tilde{\ell}(k)$, this means that we start in mode $\ell(k-1) = 1$. The system will switch to mode $\ell(k) = 2$ in at event k , and to mode $\ell(k+1) = 1$ in the mode at event $k+1$.

⁶We will explain in the next chapter how we can compute this cost function.

As a practical example, one can consider a production system. Assume that we have multiple recipes. We do not assume a fixed production scheme, which means that, at each event, we assume that each recipe has a given probability to occur. We must take into account that both recipes can occur for the next events.

Another example can be for a railway system. Assume that we have two modes, one regular schedule and one schedule during extreme weather⁷. As the weather becomes worse, the switching probability to switch to the schedule for bad weather becomes larger.

Example 7. Assume that we again have the following autonomous first order system with two modes:

$$x(k) = A_\ell \otimes x(k-1) \quad \text{where} \quad A_\ell = \begin{cases} 1 + e(k) & \text{for } \ell(k) = 1 \\ 2 + e(k) & \text{for } \ell(k) = 2 \end{cases}$$

with $e(k) \sim \mathcal{N}(0, 1)$ and initial conditions $x(k-1) = 0$ and $\ell(k-1) = 1$. We will use a prediction horizon $N_p = 2$.

Here, we assume that the model can only switch from the current mode to any next mode, with a given probability. Thus we have:

$$P[L(k) = 1 \mid \ell(k-1) = 1] = 0.6$$

$$P[L(k) = 2 \mid \ell(k-1) = 1] = 0.4$$

$$P[L(k) = 1 \mid \ell(k-1) = 2] = 0.8$$

$$P[L(k) = 2 \mid \ell(k-1) = 2] = 0.2$$

Please note that $P[L(k) = \ell(k) \mid \ell(k-1)] \in [0, 1]$ for all $\ell(k), \ell(k-1)$ and that all probabilities for each $\ell(k-1)$ add up to one, which means that we have stochastic switching.

This means that, since $\ell(k-1) = 1$, we have four possible mode switching sequences at this stage, each with its own probability. Thus:

$$\tilde{\ell}_1(k) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \tilde{\ell}_2(k) = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \tilde{\ell}_3(k) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \tilde{\ell}_4(k) = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

each with probability:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}_1(k) \mid \ell(k-1)] = 0.6 \cdot 0.6 = 0.36$$

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}_2(k) \mid \ell(k-1)] = 0.6 \cdot 0.4 = 0.24$$

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}_3(k) \mid \ell(k-1)] = 0.4 \cdot 0.8 = 0.32$$

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}_4(k) \mid \ell(k-1)] = 0.4 \cdot 0.2 = 0.08$$

Please note (again) that $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \ell(k-1)] \in [0, 1]$ for all $\tilde{\ell}(k), \ell(k-1)$ and that all probabilities for all $\ell(k-1)$ add up to one, which means that we have stochastic switching.

Now, we can simply compute:

$$J_{\text{out}, \mathbb{E}}(k) = 0.36 \mathbb{E}[J_{\text{out}}(\tilde{\ell}_1(k), \tilde{e}(k))] + 0.24 \mathbb{E}[J_{\text{out}}(\tilde{\ell}_2(k), \tilde{e}(k))] + 0.32 \mathbb{E}[J_{\text{out}}(\tilde{\ell}_3(k), \tilde{e}(k))] + 0.08 \mathbb{E}[J_{\text{out}}(\tilde{\ell}_4(k), \tilde{e}(k))]$$

⁷For many Dutch cities known as the ‘winterdienstregeling’.

4-5-3 Type 3: Dependent Random Variables, Deterministic Switching

Since we have deterministic switching, we know that there are finitely many mode switching sequences, each with probability:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] \in \{0, 1\}$$

where the switching probability still depends on $\tilde{e}(k)$.

Since we have deterministic switching, the mode switching probabilities $P[L(k) = \ell(k)]$ are constant (either zero or one). Therefore, we can (for each mode switching sequence $\tilde{\ell}(k)$ independently) define a set $\mathcal{E}_{\tilde{\ell}(k),1}(k)$ for which the switching for a particular mode switching sequence $\tilde{\ell}(k)$ is possible and we can define a set $\mathcal{E}_{\tilde{\ell}(k),0}(k)$ for which the switching is not possible.

Remark 9. *Since we assume that the mode switching probabilities $P[L(k) = \ell(k)]$ are constant, we can divide $\mathcal{E}_{\tilde{\ell}(k),1}(k)$ and $\mathcal{E}_{\tilde{\ell}(k),0}(k)$ into several polyhedral subsets. This makes the computation much easier. Both solutions are however equivalent.*

This means that we can write the mode switching probability as:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] = \begin{cases} 1 & \text{for } \tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),1}(k) \\ 0 & \text{for } \tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),0}(k) \end{cases}$$

Since the mode switching sequences for all $\tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),0}(k)$ have zero probability, we do not need to evaluate the cost function for these values of $\tilde{e}(k)$. The output cost function will then become:

$$J_{\text{out},\mathbb{E}}(k) = \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[1 \int_{\mathcal{E}_{\tilde{\ell}(k),1}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \, d\tilde{e}(k) \right. \\ \left. + 0 \int_{\mathcal{E}_{\tilde{\ell}(k),0}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \, d\tilde{e}(k) \right]$$

Here, we may observe that we only need to evaluate the output cost function for $\tilde{e}(k) \in \mathcal{E}(k)$, for which the probability is equal to one. If we have zero probability, we do not need to compute the cost function for these values of $\tilde{e}(k)$, which reduces the problem. Since the mode switching probabilities are constant, we can pull the constant probabilities outside the integral.

Remark 10. *These integrals can be computed with use of numerical integration. More details can be found in section 5-4-1. Please note that we cannot apply the Norm Based Approximation method from section 5-4-2, since we compute the integral over a subset of $\mathcal{E}(k)$.*

As a practical example, one can consider a production system. Assume that we have a production plant with multiple machines that can perform the same operation. We assume that there is a human operator that decides which machine (for that operation) will be used, for example to the first available machine.

Another example can be for a railway system. Assume that we have a simple railway system, where we take delays into account. If the delay of a train becomes too large, we can cancel that particular train. Then, we have to switch to a different mode.

Example 8. Assume that we have the following autonomous first order system with two modes:

$$x(k) = A_\ell \otimes x(k-1) \quad \text{where} \quad A_\ell = \begin{cases} 1 + e(k) & \text{for } \ell(k) = 1 \\ 2 + e(k) & \text{for } \ell(k) = 2 \end{cases}$$

with $e(k) \sim \mathcal{N}(0, 1)$ and initial conditions $x(k-1) = 0$ and $\ell(k-1) = 1$. We will use a prediction horizon $N_p = 2$.

Here, we assume that the model can only switch from the current mode to the next mode. However, the switching depends on the state. Thus we have:

$$\begin{aligned} P[L(k) = 1 \mid x(k-1) \leq 2] &= 1 \\ P[L(k) = 1 \mid x(k-1) > 2] &= 0 \\ P[L(k) = 2 \mid x(k-1) \leq 2] &= 0 \\ P[L(k) = 2 \mid x(k-1) > 2] &= 1 \end{aligned}$$

Please note that $P[L(k) = \ell(k) \mid \ell(k-1)] \in [0, 1]$ for all $\ell(k), x(k-1)$ and that all probabilities for each $x(k-1)$ equals one, which means that we have deterministic switching.

This means that we have four possible mode switching sequence at this stage, each with its own probability. The probability depends on $\tilde{x}(k)$, which is defined as:

$$\tilde{x}(k) = \begin{cases} 1 + e(k) & \text{for } \ell(k) = 1 \\ 2 + e(k) & \text{for } \ell(k) = 2 \end{cases}$$

Thus:

$$\tilde{\ell}_1(k) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \tilde{\ell}_2(k) = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \tilde{\ell}_3(k) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \tilde{\ell}_4(k) = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

each with probability:

$$\begin{aligned} \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_1(k) \mid x(k-1), e(k)] &= \begin{cases} 1 \cdot 1 = 1 & \text{for } 1 + e(k) \leq 2 \\ 1 \cdot 0 = 0 & \text{for } 1 + e(k) > 2 \end{cases} \\ \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_2(k) \mid x(k-1), e(k)] &= \begin{cases} 1 \cdot 0 = 0 & \text{for } 1 + e(k) \leq 2 \\ 1 \cdot 1 = 1 & \text{for } 1 + e(k) > 2 \end{cases} \\ \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_3(k) \mid x(k-1), e(k)] &= \begin{cases} 0 \cdot 1 = 0 & \text{for } 2 + e(k) \leq 2 \\ 0 \cdot 0 = 0 & \text{for } 2 + e(k) > 2 \end{cases} \\ \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_4(k) \mid x(k-1), e(k)] &= \begin{cases} 0 \cdot 0 = 0 & \text{for } 2 + e(k) \leq 2 \\ 0 \cdot 1 = 0 & \text{for } 2 + e(k) > 2 \end{cases} \end{aligned}$$

Please note (again) that $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid x(k-1), e(k)] \in \{0, 1\}$ for all $\tilde{\ell}(k), x(k-1)$ and that all probabilities for each $x(k-1)$ add up to one, which means that we have stochastic switching.

Here, we observe that the probabilities are piecewise constant. This means that the probability is constant over a fixed region. For one of these regions, we have zero probability, and for the other region, we have probability one to occur. So, we can simply integrate over all the $e(k) \in \mathcal{E}(k)$ for which the probability is equal to one. Now, we can simply compute:

$$J_{\text{out},\mathbb{E}}(k) = \int_{-\infty}^1 J_{\text{out}}(\tilde{\ell}_1(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) + \int_1^{\infty} J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k)$$

which means here that if $e(k) \leq 1$, we have mode switching sequence $\tilde{\ell}_1(k)$ and if $e(k) > 1$, we have mode switching sequence $\tilde{\ell}_2(k)$.

Remark 11. The last equation is not exactly equal to the real output cost function. Since we have the constraint $e(k) > 1$, we should not include $e(k) = 1$ in this integral. Therefore, we can introduce ζ , which is equal to the machine precision, such that $e(k) > 1$ is equivalent to $e(k) \geq 1 + \zeta$. The output cost function will then be equal to:

$$J_{\text{out},\mathbb{E}}(k) = \int_{-\infty}^1 J_{\text{out}}(\tilde{\ell}_1(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) + \int_{1+\zeta}^{\infty} J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k)$$

4-5-4 Type 4: Dependent Random Variables, Stochastic Switching

Since we have deterministic switching, we know that there are finitely many mode switching sequences, each with probability:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] \in \{0, 1\}$$

where the switching probability still depends on $\tilde{e}(k)$.

If we assume that the mode switching probabilities $P[L(k) = \ell(k)]$ are constant or even linear, we can (for each mode switching sequence $\tilde{\ell}(k)$ independently) define sets $\mathcal{E}_{\tilde{\ell}(k),1}(k), \dots, \mathcal{E}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k)$, where the switching probability for each $\tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),i}(k)$ is constant or linear.

Remark 12. Since we assume that the mode switching probabilities $P[L(k) = \ell(k)]$ are constant or even linear, we can divide each $\mathcal{E}_{\tilde{\ell}(k),1}(k), \dots, \mathcal{E}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k)$ into several polyhedral subsets. This makes the computation much easier. Both solutions are however equivalent.

This means that we can write the mode switching probability as:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)] = \begin{cases} \tilde{P}_{\tilde{\ell}(k),1}(k) & \text{for } \tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),1}(k) \\ \vdots & \\ \tilde{P}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k) & \text{for } \tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k) \end{cases}$$

Since the mode switching sequences for all $\tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),i}(k)$ are assumed to be constant or linear, we can split up the integral. The output cost function will then become:

$$\begin{aligned} J_{\text{out},\mathbb{E}}(k) = & \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\int_{\mathcal{E}_{\tilde{\ell}(k),1}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \tilde{P}_{\tilde{\ell}(k),1}(k) \, d\tilde{e}(k) \right. \\ & + \dots \\ & \left. + \int_{\mathcal{E}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \tilde{P}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k) \, d\tilde{e}(k) \right] \end{aligned}$$

If we can assume that the mode switching sequences are constant, we can write the switching probabilities $\tilde{P}_{\tilde{\ell}(k),i}(k)$ outside the integral:

$$J_{\text{out},\mathbb{E}}(k) = \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\tilde{P}_{\tilde{\ell}(k),1}(k) \int_{\mathcal{E}_{\tilde{\ell}(k),1}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) d\tilde{e}(k) \right. \\ \left. + \dots + \tilde{P}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k) \int_{\mathcal{E}_{\tilde{\ell}(k),n_{\mathcal{E}}}(k)} J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) d\tilde{e}(k) \right]$$

Here, we may observe that we need to evaluate the output cost function for each $\tilde{e}(k) \in \mathcal{E}_{\tilde{\ell}(k),i}(k)$ independently. This is possible, since we can write $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \mathcal{E}(k) = \tilde{e}(k)]$ as a piecewise linear function.

Remark 13. *These integrals can be computed with use of numerical integration. More details can be found in section 5-4-1. Please note that we cannot apply the Norm Based Approximation method from section 5-4-2, since we compute the integral over a subset of $\mathcal{E}(k)$.*

As a practical example, one can consider a production system. Assume that we have a single machine with two different modes. The first mode is the operational mode, the second mode is the breakdown mode. At every event, there is a probability that the machine falls back to the breakdown mode. If the duration time increases over the event, we may assume that the probability of switching to the breakdown mode increases.

Example 9. *Assume that we have the following autonomous first order system with two modes:*

$$x(k) = A_{\ell} \otimes x(k-1) \quad \text{where} \quad A_{\ell} = \begin{cases} 1 + e(k) & \text{for } \ell(k) = 1 \\ 2 + e(k) & \text{for } \ell(k) = 2 \end{cases}$$

with $e(k) \sim \mathcal{N}(0,1)$ and initial conditions $x(k-1) = 0$ and $\ell(k-1) = 1$. We will use a prediction horizon $N_p = 2$.

Here, we assume that the model can only switch from the current mode to the next mode. However, the switching depends on the state. Thus we have:

$$\begin{aligned} P[L(k) = 1 \mid x(k-1) \leq 2] &= 0.6 \\ P[L(k) = 1 \mid x(k-1) > 2] &= 0.8 \\ P[L(k) = 2 \mid x(k-1) \leq 2] &= 0.4 \\ P[L(k) = 2 \mid x(k-1) > 2] &= 0.2 \end{aligned}$$

Please note that $P[L(k) = \ell(k) \mid \ell(k-1)] \in [0,1]$ for all $\ell(k), x(k-1)$ and that all probabilities for each $x(k-1)$ equals one, which means that we have stochastic switching.

This means that we have four possible mode switching sequence at this stage, each with its own probability. The probability depends on $\tilde{x}(k)$, which is defined as:

$$\tilde{x}(k) = \begin{cases} 1 + e(k) & \text{for } \ell(k) = 1 \\ 2 + e(k) & \text{for } \ell(k) = 2 \end{cases}$$

Thus:

$$\tilde{\ell}_1(k) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \tilde{\ell}_2(k) = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \tilde{\ell}_3(k) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \tilde{\ell}_4(k) = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

each with probability:

$$\begin{aligned} \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_1(k) \mid x(k-1), e(k)] &= \begin{cases} 0.6 \cdot 0.6 = 0.36 & \text{for } 1 + e(k) \leq 2 \\ 0.6 \cdot 0.8 = 0.48 & \text{for } 1 + e(k) > 2 \end{cases} \\ \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_2(k) \mid x(k-1), e(k)] &= \begin{cases} 0.6 \cdot 0.4 = 0.24 & \text{for } 1 + e(k) \leq 2 \\ 0.6 \cdot 0.2 = 0.12 & \text{for } 1 + e(k) > 2 \end{cases} \\ \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_3(k) \mid x(k-1), e(k)] &= \begin{cases} 0.4 \cdot 0.6 = 0.24 & \text{for } 2 + e(k) \leq 2 \\ 0.4 \cdot 0.8 = 0.32 & \text{for } 2 + e(k) > 2 \end{cases} \\ \tilde{P}[\mathcal{L}(k) = \tilde{\ell}_4(k) \mid x(k-1), e(k)] &= \begin{cases} 0.4 \cdot 0.4 = 0.16 & \text{for } 2 + e(k) \leq 2 \\ 0.4 \cdot 0.2 = 0.08 & \text{for } 2 + e(k) > 2 \end{cases} \end{aligned}$$

Please note (again) that $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid x(k-1), e(k)] \in \{0, 1\}$ for all $\tilde{\ell}(k), x(k-1), e(k)$ and that all probabilities for each $e(k)$ add up to one, which means that we have stochastic switching.

For each mode sequence, we can distinguish two cases. For $\tilde{\ell}_1(k)$ and $\tilde{\ell}_2(k)$ we can distinguish the case where $1 + e(k) \leq 2$ and the case where $1 + e(k) > 2$. For $\tilde{\ell}_3(k)$ and $\tilde{\ell}_4(k)$ we can distinguish the case where $2 + e(k) \leq 2$ and the case where $2 + e(k) > 2$. So, we have:

$$\begin{aligned} J_{\text{out}, \mathbb{E}}(k) &= 0.36 \int_{-\infty}^1 J_{\text{out}}(\tilde{\ell}_1(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\ &\quad + 0.48 \int_1^{\infty} J_{\text{out}}(\tilde{\ell}_1(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\ &\quad + 0.24 \int_{-\infty}^1 J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\ &\quad + 0.12 \int_1^{\infty} J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\ &\quad + 0.24 \int_{-\infty}^2 J_{\text{out}}(\tilde{\ell}_3(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\ &\quad + 0.32 \int_2^{\infty} J_{\text{out}}(\tilde{\ell}_3(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\ &\quad + 0.16 \int_{-\infty}^2 J_{\text{out}}(\tilde{\ell}_4(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\ &\quad + 0.08 \int_2^{\infty} J_{\text{out}}(\tilde{\ell}_4(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \end{aligned}$$

In fact we can define three regions: $\mathcal{E}_1(k) = \{e(k) \mid e(k) \leq 1\}$, $\mathcal{E}_2(k) = \{e(k) \mid 1 < e(k) \leq 2\}$ and $\mathcal{E}_3(k) = \{e(k) \mid e(k) > 2\}$. We can see fairly simple that the total probability (of all mode switching sequences) for each set $\mathcal{E}_i$ for $i = 1, 2, 3$ equals one.

Remark 14. *Again, the last equation is not exactly equal to the real output cost function. Likewise remark 11, we can introduce ζ , which is equal to the machine precision. The output cost function will then be equal to:*

$$\begin{aligned}
 J_{\text{out},\mathbb{E}}(k) = & 0.36 \int_{-\infty}^1 J_{\text{out}}(\tilde{\ell}_1(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\
 & + 0.48 \int_{1+\zeta}^{\infty} J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\
 & + 0.24 \int_{-\infty}^1 J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\
 & + 0.12 \int_{1+\zeta}^{\infty} J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\
 & + 0.24 \int_{-\infty}^2 J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\
 & + 0.32 \int_{2+\zeta}^{\infty} J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\
 & + 0.16 \int_{-\infty}^2 J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k) \\
 & + 0.08 \int_{2+\zeta}^{\infty} J_{\text{out}}(\tilde{\ell}_2(k), e(k)) f_{\mathcal{E}(k)}(e(k)) \, de(k)
 \end{aligned}$$

Remark 15. *In this example, we assumed that the probabilities were piecewise constant. If the probabilities are piecewise affine functions, such as in:*

$$P[L(k) = 1 \mid x(k-1)] = \begin{cases} 0 & \text{for } x < 0 \\ x(k-1) & \text{for } 0 \leq x(k-1) \leq 1 \\ 1 & \text{for } x(k-1) > 1 \end{cases}$$

the probability for the region $\mathcal{E}_i = \{x(k-1) \mid 0 \leq x(k-1) \leq 1\}$ needs to remain inside the integral.

4-6 Conclusions

We have seen that, in order to compute the expectation of the output cost function, we have to deal with two different random variables, the mode switching uncertainty $\tilde{\ell}(k)$ and the parametric uncertainty $\tilde{e}(k)$. The first one is discrete, while the second one is continuous. To compute the expectation, we need to define the joint density function (jdf). With use of the theory of conditional probabilities, we have rewritten this jdf in a usable format.

We have defined two types of switching: deterministic switching and stochastic switching. In the first case, there is only one possible next mode, while in the second case, there are finitely many next modes. Adjoining it, we have defined various switching mechanisms for which both random variables are dependent or independent.

If both random variables are independent, computing the expectation of the cost function results in two subproblems, that are already analyzed in the current literature. If both random variables are dependent, computing the expectation of the cost function becomes somewhat more complex.

Computational Aspects

In this chapter, we will present the required knowledge in order to compute the expectation of the cost function if the mode switching uncertainty $\tilde{\ell}(k)$ and parametric uncertainty $\tilde{e}(k)$ are independent (Type 1 and Type 2, see section 4-5). For these cases, the problem can be split up in two sub problems, that are already presented in [5–10]. This chapter will mainly serve as a guideline of these papers.

5-1 Problem Formulation

In the previous chapter, we found out that we needed to compute the following optimization problem for a Type 2 Stochastic Switching Max-Plus-Linear (SSMPL) system:

$$\begin{aligned}
 & \min_{\tilde{u}(k)} \sum_{\tilde{\ell}(k) \in \mathcal{L}(k)} \left[\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)] \mathbb{E}[J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k))] \right] - \lambda \sum_{i=1}^{N_p n_u} \tilde{u}(k) \\
 & \text{s.t.} \quad \Delta u(k+j) \geq 0 \quad \text{for } j = 0, \dots, N_p - 1 \\
 & \quad \quad \Delta^2 u(k+j) = 0 \quad \text{for } j = N_c, \dots, N_p - 1 \\
 & \quad \quad A_{\text{con}} \tilde{u}(k) + B_{\text{con}} \mathbb{E}[\tilde{y}(k)] \leq c_{\text{con}}(k)
 \end{aligned}$$

Remark 16. *The outlined theory is valid for a Type 1 SSMPL system as well, since it is a special case of a Type 2 SSMPL system.*

So, we need to perform the following computations:

- $\mathcal{L}(k)$: The set of all (possible) mode switching sequences.
- $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k)]$: The probability of each mode switching sequence to occur.
- $\mathbb{E}[J_{\text{out}}(\tilde{\ell}(k), \tilde{e}(k))]$: The output cost function for each mode switching sequence¹.

¹Taking the parametric uncertainty $\tilde{e}(k)$ into account.

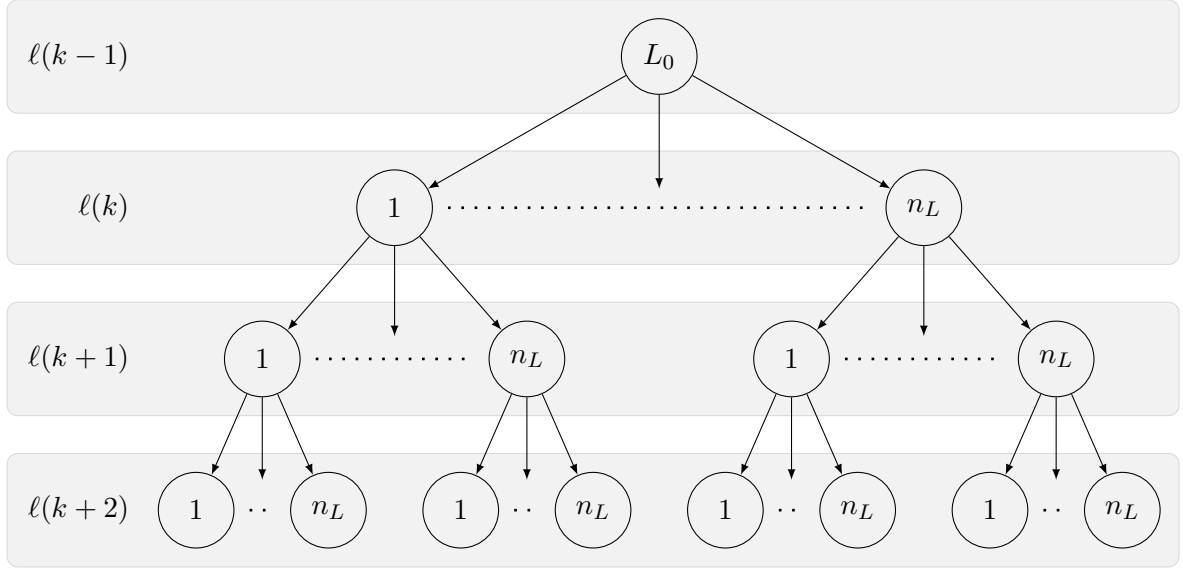


Figure 5-1: Growth of number of mode switching sequences.

5-2 Mode Switching Sequence

We know from the modeling phase that the model has n_L different modes. Assume that we are in the k th event, then there are n_L possible mode transitions for the $(k+1)$ th event. For each of these n_L different modes, there are again n_L possible mode transitions². This process can be compared with a search tree, and is visualized in figure 5-1. We assume we know the previous mode $\ell(k-1)$, denoted as L_0 .

Further, during the Model Predictive Control (MPC) design phase, we decided to use a prediction length of N_p event steps. This means that we have:

$$n_{\mathcal{L}} = (n_L)^{N_p}$$

where $n_{\mathcal{L}}$ is the number of mode switching sequences (each with length N_p).

Remark 17. The reader should already notice that the number of mode switching sequences grows exponentially with respect to N_p . This might impose numerical problems in a later stage. An approximation method to reduce the set of mode switching sequences, and thus the computational burden, will be discussed in section 5-3-2.

5-3 Probabilities for each Mode Switching Sequence

As already explained in the previous section, we will not deal with the case where the mode switching sequence $\tilde{\ell}(k)$ depends on the parametric uncertainty $\tilde{e}(k)$ (Type 3 and Type 4

²We assume that every mode transition is possible in this step. If one of the mode transitions is not possible, we can assign it a zero probability in the next step.

systems, see section 4-5). Thus, we know that the switching probability is defined as:

$$\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \ell(k-1), \tilde{u}(k), \tilde{v}(k)] = \prod_{j=0}^{N_p-1} P[L(k+j) = \ell(k+j) \mid \ell(k+j-1), u(k+j), v(k+j)]$$

In the following sections, we will first discuss the full problem, and introduce an approximation method afterwards.

5-3-1 Full Problem

Since both $\tilde{u}(k)$ and $\tilde{v}(k)$ are assumed to be known, we can compute all mode switching probabilities. If we assume pure stochastic switching, i.e. the switching does not depend on $\tilde{u}(k)$ or $\tilde{v}(k)$, then the switching probabilities no longer depend on the current event step k . If the system currently is in mode $\ell(k-1) = i$ and we want to know the probability that the system might switch to mode $\ell(k) = j$, we need to obtain:

$$P[L(k) = j \mid i]$$

We can rewrite this in a more generalized way, with a switching matrix. Let the variable i represents the i th row, and j represents the j th column of this matrix. Thus, we have:

$$[P_M]_{(i,j)} = P[L(k) = j \mid i]$$

and we obtain³:

$$P_M = \begin{bmatrix} P[L(k) = 1 \mid 1] & \cdots & P[L(k) = n_L \mid 1] \\ \vdots & \ddots & \vdots \\ P[L(k) = 1 \mid n_L] & \cdots & P[L(k) = n_L \mid n_L] \end{bmatrix}$$

So, if we want to compute the mode switching probability for a particular mode switching sequence, we can simply multiply the corresponding elements of matrix P_M .

5-3-2 Approximation – Scenario Based Approximation

As mentioned before, the number of mode sequence vectors grows exponentially with respect to N_p . This means, that if we increase the prediction horizon with only one event step, the number of mode sequence vectors grows with a factor n_L . In [10], an approximation method is proposed that reduces the total number of mode switching sequences drastically, while still maintaining satisfactory results. The main idea behind this approximation is that we only have to investigate the most probable ones. The less probable ones have a negligible impact on the outcome of the cost function, and therefore can be neglected.

³Since we assume purely stochastic switching, the mode switching probabilities no longer depend on the event step k , and thus we may write P_M instead of $P_M(k)$.

In the example in [10], it has been shown that, if we only consider the 50 most probable mode switching sequences, the cumulative probability of all the selected mode switching sequences is still 83%. If we compare the size of this reduced set of mode switching sequences with the full set ($n_{\mathcal{L}} = 6561$), we see that we only investigate 0.8 % of all mode sequence vectors. This has a drastic improvement on the performance of the controller.

The method proposed in [10] is merely the most optimal way to obtain the $n_{\mathcal{L},\text{red}}$ most probable mode switching sequences. If we assume purely stochastic switching, we can compute the probabilities of each mode switching sequence in advance.⁴ This allows us to investigate the $n_{\mathcal{L},\text{red}}$ most probable mode switching sequences in advance as well.

Remark 18. *In [10], the value of $n_{\mathcal{L},\text{red}}$ is unrestricted. However, it might be the case that by choosing an arbitrary value for $n_{\mathcal{L},\text{red}}$, some mode switching sequences with probability p are included in the reduced set, while others with the same probability p might not be included. This might cause some strange results. Therefore, at least in my opinion, it should be chosen such that there is no mode switching sequence in the $n_{\mathcal{L},\text{red}}$ most probable mode switching sequences that has an equal probability as a (different) mode switching sequence that is not inside the $n_{\mathcal{L},\text{red}}$ most probable mode switching sequences.*

5-4 Cost Function for each Mode Switching Sequence

The last step would be to compute the cost function for each mode switching sequence $\tilde{\ell}(k) \in \mathcal{L}(k)$ independently, while still taking the parametric uncertainty $\tilde{e}(k)$ into account. For this, we refer to the theory in [5].

5-4-1 Full Problem

In order to obtain the optimal control signal, we need to find an expression for the expectation of $J_{\text{out},\mathbb{E}}(k)$ and $\tilde{y}(k)$. It has been shown that there exist scalars $\alpha_j(k)$ and non-negative vectors $\beta_j(k)$ and $\gamma_j(k)$ for $j = 1, \dots, n_w$, such that both $J_{\text{out}}(k)$ and $\tilde{y}(k)$ can be written in the following form:

$$w(k) = \max_j (\alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k))$$

If we now want to compute the expectation of $w(k)$, denoted by $\hat{w}(k)$, we would obtain:

$$\hat{w}(k) = \mathbb{E}[w(k)] = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \max_j (\alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) \, d\tilde{e}(k)$$

where $d\tilde{e}(k) = d\tilde{e}_1(k) \dots d\tilde{e}_{n_{\tilde{e}}}(k)$.

Now, we can define the sets $\Phi_j(\tilde{u}(k))$ for $j = 1, \dots, n_w$, such that for all $\tilde{e}(k) \in \Phi_j(\tilde{u}(k))$, it holds that:

$$w(k) = \alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k)$$

⁴Since $\tilde{P}[\mathcal{L}(k) = \tilde{\ell}(k) \mid \ell(k-1)]$ only depends on $\ell(k-1)$, not on k .

where:

$$\bigcup_{j=1}^{n_w} \Phi_j(\tilde{u}(k)) = \mathbb{R}^{n_{\tilde{e}}} \quad \text{and} \quad \Phi_i(\tilde{u}(k)) \cap \Phi_j(\tilde{u}(k)) = \emptyset \quad \forall i \neq j$$

The first constraint means that the union of all sets should cover all the uncertainties, i.e. all uncertainties are modeled. The second constraint means that there is no interior overlap between the sets, i.e. all regions should be unique and non-overlapping, except on the boundaries.

Now, the expectation of $w(k)$ can be rewritten as:

$$\hat{w}(k) = \sum_{j=1}^{n_w} \int \cdots \int_{\tilde{e}(k) \in \Phi_j(\tilde{u}(k))} (\alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k)) f_{\mathcal{E}(k)}(\tilde{e}(k)) d\tilde{e}(k)$$

where $f_{\mathcal{E}(k)}(\tilde{e}(k))$ is the probability density function (pdf) of $\tilde{e}(k)$. In [5], it is proved that $\mathbb{E}[w(k)]$ is convex in $\tilde{u}(k)$ and the subgradient of $\mathbb{E}[w(k)]$ is given by:

$$\nabla_{\tilde{u}(k)} \hat{w}(k) = \sum_{j=1}^{n_w} \beta_j(k) \int \cdots \int_{\tilde{e}(k) \in \Phi_j(\tilde{u}(k))} f_{\mathcal{E}(k)}(\tilde{e}(k)) d\tilde{e}(k)$$

5-4-2 Approximation – Norm Based Approximation

The reader should notice that the above method uses numerical integration over all the uncertainties, i.e. over $n_{\tilde{e}(k)}$ variables. Numerical integration is the most time consuming part in the above method. In [6], an approximation method is proposed that reduces the computation time significantly by approximating these integrals, considering the relation between (vector) norms, to avoid the numerical integrations.

The max-operator from the previous section is equivalent to the infinity norm of a vector. By considering vector norm inequalities, $\hat{w}(k)$ can be approximated by its upper bound. First, we define the p - and ∞ -norm, respectively, and the related inequality:

$$\begin{aligned} \|y\|_p &= (|y_1|^p + \cdots + |y_n|^p)^{1/p} \\ \|y\|_\infty &= \max(|y_1|, \dots, |y_n|) \\ \|y\|_p &\leq \|y\|_\infty \end{aligned}$$

where $y = [y_1 \ \dots \ y_n]$ is a vector in $\mathbb{R}^n$. However, in the computation of $\hat{w}(k)$ in the previous section, we did not consider the absolute values. Therefore, we introduce a scalar L , such that $L \leq y_j$ for almost all j 's. The role of L is to decrease the error of approximating y_j with $|y_j|$.

Considering the above norm inequality and Jensen's Inequality, we obtain:

$$\mathbb{E}[\max(y_1 - L, \dots, y_n - L)] + L \leq \left(\sum_{j=1}^n \mathbb{E}[|y_j - L|^p] \right)^{1/p} + L$$

where y_j is defined as⁵:

$$y_j = \alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k)$$

The approximation of $\hat{w}(k)$, denoted by $\hat{w}_{\text{app}}(k)$ is thus defined as:

$$\hat{w}_{\text{app}}(k) = \left(\sum_{j=1}^{n_w} \mathbb{E}[(\alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k) - L)^p] \right)^{1/p} + L$$

If we further assume that each element of $\tilde{e}(k)$ is normally distributed, i.e. $\tilde{e}_i(k) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ for $i = 1, \dots, n_{\tilde{e}}(k)$, we can define L as:

$$L = \min_j (\mu_j - 3\sigma_j)$$

Under the assumption that p is an even integer greater than or equal to two, we can write the expectation of the p th moment of every $y_j - L$ as⁶:

$$\mathbb{E}[(\alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k) - L)^p] = \sigma_j^p i^{-p} H_p\left(\frac{i\mu_j}{\sigma_j}\right)$$

where

$$\begin{aligned} H_p(x) &\equiv (-1)^p \exp\left(\frac{x^2}{2}\right) \frac{d^p}{dx^p} \exp\left(-\frac{x^2}{2}\right) \\ &= p! \sum_{k=0}^{p/2} \frac{(-1)^k x^{p-2k}}{2^k k! (p-2k)!} \end{aligned}$$

is the p th Hermite polynomial⁷. This last step reduces the computational complexity significantly, since we have an analytic expression for the obtained upper bound. The approximation method in closed form is thus:

$$\hat{w}_{\text{app}}(k) = \left(\sum_{j=1}^{n_w} \sigma_j^p i^{-p} H_p\left(\frac{i\mu_j}{\sigma_j}\right) \right)^{1/p} + L$$

In [6], it is proved that $\hat{w}_{\text{app}}(k)$ is convex in $\tilde{u}(k)$ and the subgradient is given by:

$$\nabla_{\tilde{u}(k)} \hat{w}_{\text{app}}(k) = \left(\sum_{j=1}^{n_w} \sigma_j^p i^{-p} H_p\left(\frac{i\mu_j}{\sigma_j}\right) \right)^{1/p-1} \sum_{j=1}^{n_w} \beta_j(k) \sigma_j^{p-1} i^{-p+1} H_{p-1}\left(\frac{i\mu_j}{\sigma_j}\right)$$

where:

$$\begin{aligned} \mu_j &= \alpha_j(k) + \beta_j^T(k)\tilde{u}(k) + \gamma_j^T(k)\tilde{e}(k) - L \\ \sigma_j^2 &= \gamma_j^T \begin{bmatrix} \sigma_1^2 & \dots & \sigma_{n_{\tilde{e}}}^2 \end{bmatrix}^T \end{aligned}$$

⁵For the intermediate steps, the interested reader is referred to [6].

⁶The term $\mathbb{E}[x^p]$ is called the p th moment of x .

⁷This p th Hermite polynomial can be computed using the following recursive relations:

$$H_0(x) = 1, \quad H_1(x) = x, \quad H_{p+1}(x) = xH_p(x) - pH_{p-1}(x)$$

5-5 Convexity

So, we have seen that computing the (output) cost function for a particular mode switching sequence is a convex problem in $\tilde{u}(k)$. This means that the problem can be solved efficiently [16]. A fundamental property of convex optimization problems is that any locally optimal solution is also (globally) optimal. Further, since the gradient is known, the (negative) gradient always points in the direction of the optimum.

Under the assumptions that the constraints are a nondecreasing affine function of the output [5], the resulting optimization problem is a convex optimization problem as well, the full optimization problem can be solved with (efficient) convex optimization algorithms.

5-6 Conclusions

In this chapter, we have seen how we can compute the (output) cost function. First, we need to obtain all (possible) mode switching sequences, and calculate their corresponding probabilities. Next, we can calculate the output cost function for each mode switching sequence independently. Then we can simply multiply all probabilities with their corresponding output cost function, and add them all up.

Computing the cost function for each mode independently is very time consuming. The number of mode switching sequences grows exponentially with respect to the prediction horizon. Therefore, a reduction in the total number of mode switching sequences is proposed. This method reduces the problem significantly, while maintaining good results.

Secondly, computing the cost function itself requires numerical integration over all the uncertainties, which is very time consuming as well. Therefore, an approximation method is proposed that reduces the computation time considerably by estimating these integrals with (vector) norms, eliminating the need for numerical integration.

Both solutions are proved to be convex. Therefore, the full optimization problem is convex as well. Thus, the full optimization problem can be solved with (efficient) convex optimization algorithms.

Chapter 6

Worked Example

In this chapter, we will show how we can apply the theory from this MSc Thesis to a real example. This controller is implemented in MATLAB, and the results are discussed shortly.

6-1 Model Description

We use the same system as in example 5, slightly modified for this application:

Example 10. Consider the production system of figure 6-1. This system consists of three machines M_1 , M_2 and M_3 . Three products (A, B and C) can be produced by this system, each with its own recipe, meaning that the order in the production sequence is different for every product.

For product A, the production order is M_1 – M_2 – M_3 , which means that the raw material is fed to machine M_1 , where it is processed. The intermediate product is sent to machine M_2 for further processing, and finally the product is finished in machine M_3 . Similarly, for product B, the processing order is M_2 – M_1 – M_3 , and for product C the processing order is M_1 – M_3 – M_2 . We assume that the type of the k th product (A, B or C) only becomes available at the start of the production, so that we do not know $\ell(k)$ when computing $u(k)$.

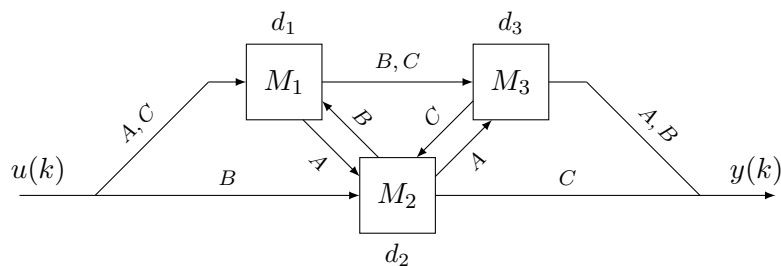


Figure 6-1: Production system.

Each machine starts working as soon as possible on each batch, i.e. as soon as the raw material or the required intermediate products are available, and as soon as the machine is idle (i.e. the previous batch has been finished and has left the machine). We define $u(k)$ as the time instant at which the system is fed for the k th time, $x_i(k)$ as the time instant at which machine i starts for the k th time, and $y(k)$ as time instant at which the k th product leaves the system. We assume that all the internal buffers are large enough, and no overflow will occur.

Machine M_3 performs a quality inspection on the intermediate product. Therefore, we assume that $d_3(k) = 5 + e(k)$ is a normally distributed random variable. The processing times for the other machines (M_1 and M_2) are assumed to be deterministic and given by $d_1 = 4$ and $d_2 = 6$ respectively. Further, we assume the transportation times between the machines are assumed to be negligible.

The mode switching probability from one recipe (mode) to the next one is assumed to be purely stochastic, and given by:

$$\begin{aligned}
 P[L(k) = 1 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.6 \\
 P[L(k) = 1 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.2 \\
 P[L(k) = 1 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.2 \\
 P[L(k) = 2 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.2 \\
 P[L(k) = 2 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.6 \\
 P[L(k) = 2 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.2 \\
 P[L(k) = 3 \mid \ell(k-1) = 1, x(k-1), u(k), v(k)] &= 0.2 \\
 P[L(k) = 3 \mid \ell(k-1) = 2, x(k-1), u(k), v(k)] &= 0.2 \\
 P[L(k) = 3 \mid \ell(k-1) = 3, x(k-1), u(k), v(k)] &= 0.6
 \end{aligned}$$

which means that, if we have a specific recipe for the k th product, then the probability of having the same recipe for the $(k+1)$ th product is 60%, and the probability to switch to the other recipes is 20%.

All systems start at time equals zero, and we require the model to follow a given reference signal:

$$x(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad r(k) = 15 + 10k$$

6-2 First Observations

In the example, we assume purely stochastic switching. Since the mode switching depends only on the previous mode $\ell(k-1)$, the mode switching uncertainty $\tilde{\ell}(k)$ and the parametric uncertainty $\tilde{e}(k)$ are independent. We observe that we have a Type 2 Stochastic Switching Max-Plus-Linear (SSMPL) system, and therefore, we can use the theory from chapter 5.

Since $d_3(k)$ is a normally distributed random variable, we can use the Norm-Based Approximation. This approximation reduces the computational complexity significantly.

Finally, since the stochastic model and the mode switching mechanism does not change over k , we can compute most of the equations in advance. This will increase the performance as well.

6-3 Model

There is a small difference between example 5 and the above example. Here, we assume that d_1 and d_2 are deterministic, while $d_3(k)$ is a stochastic random variable. So, we can rewrite the model with the following assumptions:

$$\begin{aligned} d_1(k) &:= d_1 & \forall k \\ d_2(k) &:= d_2 & \forall k \end{aligned}$$

6-4 Implementation

Most of the required mathematics are coded in MATLAB. The main reason why we chose MATLAB is that it has strong optimization algorithms. Besides that, we are only interested in the numerical solution, and we have a lot more experience with MATLAB than with other programs.

However, in the computation of the prediction model, we need a symbolic¹ representation of the model. MATLAB has a Symbolic Toolbox, but this toolbox won't compute the maximum of two symbols (variables). Therefore, we used MATHEMATICA. We have linked both programs using an interface called MATHLINK [17].

6-5 Controller Performance

The actual implementation of the Model Predictive Controller can be found in the appendices A-1 and A-2. We can divide the calculations in two stages. In the first stage, called the 'Prediction Stage', we compute all the variables that do not vary over the different events. In the second stage, namely the 'Optimization Stage', we optimize the input sequence for every event step. The code ran on my personal laptop².

Running the simulations, we discovered that the computation time of the 'Prediction Stage' was quite long. The computation time for different prediction horizons can be found in table 6-1. Since obtaining the prediction matrices for $N_p = 4$ took more than 20 minutes, we did not investigate any further increase in the prediction horizon.

Remark 19. *The bad performance of the prediction stage is most likely due to the inefficiency of the MATHEMATICA code. One could observe an exponential growth, which is mainly due to the increase of the number of mode switching sequences and the fact that each mode switching sequence requires more function evaluations. On the other hand, the link between MATLAB and MATHEMATICA is a bit slow as well. Since the `MathLink()` libraries were written for older versions of MATLAB and MATHEMATICA, the automatic conversion between both languages failed and was performed by rewriting strings manually. Finally, MATHEMATICA caches results to prevent recomputing. However, due to memory limitations, this might not work as expected for large systems.*

¹With a symbolic model, we mean that we express the predicted output as function of all variables.

²Intel Core2Duo (T9600, 2.8 GHz), 4.0 GB of RAM

Since the previous stage can be computed offline, we are merely interested in the performance of the ‘Optimization Stage’. Please note that the Norm Based Approximation method (see section 5-4-2) is already incorporated. It has been shown [6] that this method reduces the computation time significantly. Furthermore, it makes the coding much more easier. We know that the parameter p influences the quality of the approximation. For now, there is no analytic method to find the optimum p . Therefore, by means of experiments (see figure 6-2), we choose $p = 30$.

Remark 20. *In figure 6-2, the exact solution is denoted with the abbreviation ‘MC’, which stands for Monte Carlo Simulations. This simulation took more than a few days. We used one million random variables for each iteration.*

So, first we ran the ‘Optimization Stage’ for various combinations of the prediction horizon and control horizon. The results can again be found in table 6-1. Please note that the time information in the ‘Optimization stage is the average computation time per event, out of 40 events in total.

As expected, the computation time increases as the prediction horizon increases, and decreases as the control horizon decreases (for constant prediction horizons). The next step will be to introduce the Scenario Based Approximation method (see section 5-3-2). As a design criteria, we will only analyze the most probable mode switching sequences, with a cumulative probability of at least 80 %. The results can be found in table 6-2.

For this example, we need a computation time of less than one second. Based on the results presented in table 6-1 and 6-2, the most optimal choice is a prediction horizon of three, i.e. $N_p = 3$, and a control horizon of two, i.e. $N_c = 2$. Since the computation time for these variables is already half a second, we do not need the Scenario Based Approximation method. No analysis on the quality of the approximation methods can be performed at this stage.

The last parameter is λ , which is related to the ratio in which $J_{\text{out},\mathbb{E}}(k)$ and $J_{\text{in}}(k)$ contribute to the cost function $J(k)$. If N_p increases, $J_{\text{out}}(k)$ becomes larger as well (since we sum up the errors over the whole prediction horizon). In this example, we want to minimize the delay penalty costs. Experiments (see figure 6-3) have shown that $\lambda = 0.1$ minimizes the error between the output signal $y(k)$ and the reference signal $r(k)$.

6-6 Small Note on Stability

In this report, we did not address the issue of stability. From the current literature [8, 9], we know that stability is not an intrinsic feature of a Max-Plus-Linear (MPL) system, since it also depends on the reference signal of the system. Each system has a (lower) production capacity, or equivalently an (upper) production rate. We cannot require that the system works at a faster production rate. If the reference signal demands a faster production rate, the system becomes unstable. In figure 6-4, we can indeed see that if we require a too fast production rate, the system becomes unstable.

	Prediction Stage	Optimization Stage		
		$N_c = 2$	$N_c = 3$	$N_c = 4$
$N_p = 2$	4.8	0.21	★	★
$N_p = 3$	62.1	0.87	1.56	★
$N_p = 4$	1400	3.31	7.29	10.68

Table 6-1: Average time durations for both the 'Prediction Stage' and the 'Optimization Stage'.

	$n_{\text{red}}/n_{\text{full}}$	p_{cum}	$N_c = 2$	$N_c = 3$	$N_c = 4$
$N_p = 2$	5 / 9 (55.6 %)	84.0 %	0.14 (67 %)	★	★
$N_p = 3$	19 / 27 (70.4 %)	93.6 %	0.61 (70 %)	1.21 (78 %)	★
$N_p = 4$	33 / 81 (40.8 %)	82.1 %	1.47 (44 %)	2.75 (37 %)	4.21 (39 %)

Table 6-2: Average time durations with use of the Scenario Based Algorithm for the 'Optimization Stage'.

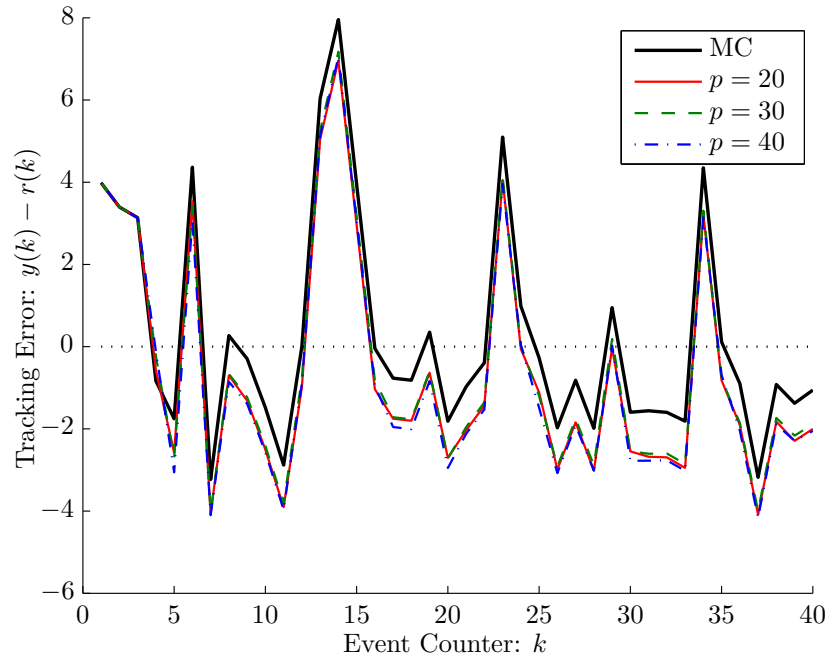


Figure 6-2: Tuning the Norm Based Approximation method, with the parameter p . ($N_p = 3$, $N_c = 2$, $\lambda = 0.1$)

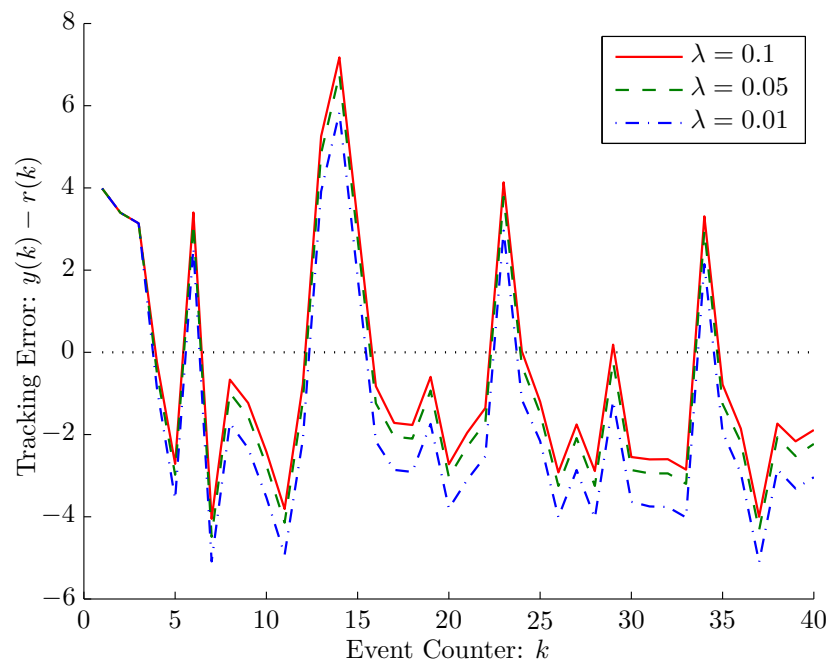


Figure 6-3: Tuning the Model Predictive Controller, with the parameter λ .
 $(N_p = 3, N_c = 2, p = 30)$

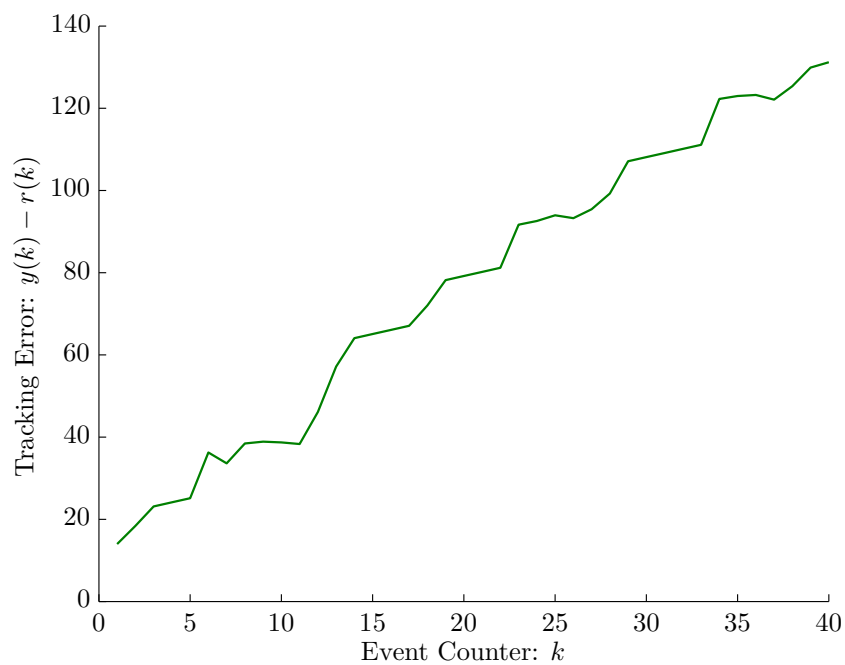


Figure 6-4: Unstable System, where the tracking error grows unbounded.
 $(r(k) = 5k)$

6-7 Conclusion

In this chapter, we have seen that the presented theory indeed can be applied in practice. However, the computational burden of the optimization problem hinders the applicability. Even for such a small system (with three states and one uncertain parameter), the application of the controller is limited. However, we have seen that even for a simple controller (with a small prediction and control horizon), the controller is able to track the reference signal sufficiently.

Chapter 7

Conclusions

This final section will conclude this report with a conclusion and some recommendations for future work.

7-1 Conclusion

In this MSc Thesis, we have considered the control of Stochastic Switching Max-Plus-Linear (SSMPL) systems. These systems are a subclass of the Discrete Event Systems (DES), which contains two types of uncertainty. At first, there is mode uncertainty, which means that the system can switch between different modes of operation. The switching between the modes can be deterministic or stochastic. Secondly, there is parametric uncertainty, which means that we include noise and modeling errors in a stochastic setting. Since the entries of the system matrices correspond to processing or transportation times, this uncertainty perturbs the system as additive noise. We have presented an unified framework that deals with both uncertainties simultaneously.

We have extended the popular Model Predictive Control (MPC) framework to the new class of Max-Plus-Linear (MPL) systems. The main reason for using a MPC approach is that it allows the inclusion of constraints on the inputs and outputs. Since DES are event-driven, these constraints follow naturally (since, for example, the system cannot go back in time). MPC computes the most optimal control signal by minimizing the cost function. Since the model contains stochastic uncertainties, we consider the expectation of the output cost function. In general, the optimization will be complex and time consuming, due to the presence of numerical integration that appears in the calculation of the expectation.

If both types of uncertainty are independent, the evaluation of the expectation can be rewritten in a much simpler way. We have shown that, if both uncertainties are independent, the computation of the cost function can be divided in two subproblems, that are already analyzed in literature. Further, it has been shown that these subproblems are convex, and thus can be solved efficiently. It has been shown that we can compute the cost function for each

mode switching sequence independently, and multiply it with the probability of that particular mode switching sequence. If we add all these cost functions, we obtain the cost function for the full problem.

If both types of uncertainty are dependent, the evaluation of the expectation becomes much more difficult. In general, for each possible mode switching sequence, we can define sets for which the switching probability is constant or affine. Then, we can evaluate the expectation for each set independently, and add them up.

7-2 Limitations

As far as we know, this is the first attempt to include both stochastic parametric uncertainty and mode switching uncertainty in a single framework. This means that we started to develop a unified framework, that can be used for different kind of applications. However, the proposed approach has some major practical limitations. The main limitation is computational complexity. Since the cost function needs to be re-optimized every time new information about the system becomes available, the time instances between the optimizations become a major limitation. Further, the computational complexity grows exponentially as we investigate larger systems.

Another limitation is that we are dealing with DES systems. This implies that the state contains time information about the internal events. It is very likely, that some of this information is not available, while other information about the same event occurred in the past. Since the controller is time-driven, some uncertainties are no longer uncertain, while others are bounded in some way. This introduces some implications, which are not analyzed in this MSc Thesis.

Some approximations are discussed in this MSc Thesis as well. They have some limitations as well. For example, the approximation of the numerical integration over the parametric uncertainties is only valid for limited probability distributions. If we want to apply the presented theory to, for example, a railway system (which is one of the main applications my supervisor (Ton van den Boom) is working on), this approximation is not valid anymore. This means that the computation time increases significantly.

Although we are familiar with these practical limitations, the presented theory is applicable. These conditions are not too strict, which means that there are still many applications to which the presented theory is applicable. One should notice that this MSc Thesis is a first approach, there is still a lot of work to do.

7-3 Recommendations

In this MSc Thesis, we did not investigate the cost function for Type 3 and 4 SSMP systems in depth. The next step could be to further investigate these types. The main problem is that both uncertainties are dependent. Computing the cost function will become very time consuming, since these problems are probably not convex problems. Therefore, we need to search for approximations methods to reduce the computational complexity.

From the current literature [8, 9], we know that stability is not an intrinsic feature of a SSMPL system, since it also depends on the reference signal of the system. Each system has a (lower) production capacity, or equivalently an (upper) production rate. We cannot require that the system works at a faster production rate. If the reference signal demands a faster production rate, the system becomes unstable. Conditions for stability for stochastic MPL systems haven't been found in current literature. In [18], stochastic MPL systems are analyzed, but the theory is limited to autonomous systems (i.e., with no input).

As already presented in the previous section, there are still some major limitations with the timing aspects. Since the model operates at the event domain and the controller operates at the time domain, there is a mismatch between the available information that occurred in the past and in the future. This MSc Thesis does not deal with this issue, it only assumes that all information up to a particular event is known, and all information of future events is unknown. In reality, this is of course not applicable.

At last, we have presented some tuning parameters. One could investigate some tuning rules for these parameters. The influence of these parameters on the performance of the controller are known, but we still do not know what values of the parameters result in the desired behavior of the system.

Appendix A

Example Code

This chapter presents the actual implementation of the example from chapter 6. To obtain a copy of the code, please send an email to the corresponding author: Stefan@Loenhoet.nl.

A-1 Matlab

In this section, we will present the used MATLAB code. As discussed in chapter 6, MATLAB is linked with MATHEMATICA. Both programs are linked with the `Mathlink()` function, more information can be found in appendix A-2. The actual version of MATLAB is R2011a (7.12).

A-1-1 Main.m

```
1 close all
2 clear all
3 clc
4
5 % Simulation Time
6 kend = 40;
7
8 % Define MPC-er Parameters
9 Np = 3; % Prediction Horizon (Np > 1)
10 Nc = 2; % Control Horizon (Nc > 1)
11 L = 3; % Number of modes
12 lambda = 0.1; % Ratio between Jout and Jin
13 p = 30; % Hermite Approximation Order (Norm Approximation Method) (-1 := ...
    Disabled, using Monte Carlo)
14 nRed = -1; % Size of Reduced Set (Scenario Based Algorithm) (-1 := Disabled)
15
16 % Define Signals
17 x0 = [0; 0; 0]; % Initial Condition
18 x = [x0 zeros(size(x0, 1), kend)];
19 y = zeros(kend, 1);
20 u = zeros(kend, 1);
```

```

21 r = 5 + 10*(1:kend+Np)'; % r = 5*(1:kend+Np)';
22 clear x0;
23
24 % Connect with Mathematica to Calculate Predictor Matrices
25 [xi beta delta gamma Lcal] = MathLink(L, Np);
26
27 % Generate Switching Probability Matrix
28 Pii = 0.6; % P(i|i)
29 Pij = (1-Pii)/(L-1); % (P(i|j)
30 PM = Pij*ones(L)+(Pii-Pij)*eye(L);
31
32 % Generate Mode Switching Sequence, for Simulation
33 load ell; % d = rand(2*kend, 1); save ell d
34 dell = ceil(max((d-Pii)*(L-1)/(1-Pii), 0));
35 ell = mod(cumsum(dell), L) + 1;
36 clear d dell Pii Pij;
37
38 % Precalculate P(Lcal)
39 P = cell(L, 1);
40 for L0 = 1:L
41     P{L0} = zeros(size(Lcal{L0}, 1), 1);
42     for i = 1:size(Lcal{L0}, 1)
43         P{L0}(i) = PM(Lcal{L0}(i, 1), L0);
44         for j = 1:size(Lcal{L0}, 2) - 1
45             P{L0}(i) = P{L0}(i) * PM(Lcal{L0}(i, j+1), Lcal{L0}(i, j));
46         end
47     end
48 end
49 clear i j k
50
51 % Apply Scenario Based Algorithm
52 for L0 = 1:L
53     if (nRed > -1) && (nRed < size(Lcal{L0}, 1))
54         [~, iPs] = sort(P{L0}, 'descend');
55         Lcal{L0} = Lcal{L0}(P{L0} >= P{L0}(iPs(nRed)), :);
56         xi{L0} = xi{L0}(P{L0} >= P{L0}(iPs(nRed)), :);
57         beta{L0} = beta{L0}(P{L0} >= P{L0}(iPs(nRed)), :);
58         delta{L0} = delta{L0}(P{L0} >= P{L0}(iPs(nRed)), :);
59         gamma{L0} = gamma{L0}(P{L0} >= P{L0}(iPs(nRed)), :);
60         P{L0} = P{L0}(P{L0} >= P{L0}(iPs(nRed)));
61     end
62 end
63
64 % Constraint Matrix for Nc (\Delta^2 u(k+j) = 0)
65 T = [eye(Nc);
66     zeros(Np-Nc, Nc)];
67 for i = Nc+1:Np
68     T(i, Nc-1) = T(i-1, Nc-1) - 1;
69     T(i, Nc) = T(i-1, Nc) + 1;
70 end
71
72 % Constraint Matrix for \Delta u(k+j) >= 0
73 S = -1*eye(Np);
74 S = S + diag(ones(1, Np-1), -1);
75
76 % Define Mean and Standard Defiation of e(k)
77 muE = zeros(Np+1, 1);

```

```

78 sigmaE = ones(Np+1, 1);
79
80 % Noise Signals
81 load enor; % e = randn(43, 3); save enor e
82 d1 = 4 + 0*e(:,1);
83 d2 = 6 + 0*e(:,2);
84 d3 = 5 + 1*e(:,3);
85
86 uPrev = 0;
87 for k=1:kend
88     % Step Counter
89     disp(['k = ' num2str(k)])
90
91     % Generate Estimate of Optimum (Initial Guess)
92     upp = r(k:k+Nc-1);
93     low = uPrev*ones(Nc, 1);
94     tu0 = (upp+low)/2;
95     clear upp low
96
97     % Compute w
98     w = [-r(k:k+Np-1); x(:,k)];
99
100    % Obtain Optimal Input Ssequence
101    L0 = ell(k);
102    u(k) = ConvexOptimization(tu0, w, uPrev, muE, sigmaE, lambda, T, S, p, ...
        xi{L0}, beta{L0}, delta{L0}, gamma{L0}, P{L0});
103    uPrev = u(k);
104
105    % Closed Loop Simulation
106    [x(:, k+1), y(k)] = ClosedLoopSimulation(x(:, k), u(k), L0, d1(k:k+1), ...
        d2(k:k+1), d3(k:k+1));
107
108 end

```

A-1-2 ConvexOptimization.m

```

1 function uopt = ConvexOptimization(u0, w, uPrev, muE, sigmaE, lambda, T, S, ...
    p, xi, beta, delta, gamma, P)
2
3     if p > 0 % Norm Based Approximation Method
4
5         % Set Optimization Options
6         options = optimset;
7         options.LargeScale = 'on';
8         options.Algorithm = 'Active-Set';
9         options.GradObj = 'on';
10        options.GradConstr = 'on';
11
12        uopt = fmincon(@(u)CostFunction(u, xi, beta, delta, gamma, uPrev, ...
            lambda, muE, sigmaE, p, T, P, w), ...
            u0, [], [], [], [], [], [], ...
            @(u)CostConstraints(u, uPrev, T, S),...
            options);
13
14    else % Monte Carlo Simulation
15
16
17
18

```

```

19     % Set Optimization Options
20     options = optimset;
21     options.LargeScale = 'on';
22     options.GradObj = 'off';
23     options.GradConstr = 'on';
24     options.CompletePoll = 'on';
25     options.UseParallel = 'always';
26
27     uopt = patternsearch(@(u)CostFunction(u, xi, beta, delta, gamma, ...
28         uPrev, lambda, muE, sigmaE, p, T, P, w), ...
29         u0, [], [], [], [], [], [], ...
30         @(u)CostConstraints(u, uPrev, T, S),...
31         options);
32
33     % Receding Horizon Principle
34     uopt = uopt(1);
35
36 end

```

A-1-3 CostFunction.m

```

1 function [J, dJ] = CostFunction(tu, xi, beta, delta, gamma, uPrev, lambda, ...
2     muE, sigmaE, p, T, P, w)
3
4     % Transformation for Nc ( $\Delta^2 u(k+j) = 0$ )
5     uc = T*tu;
6
7     % Compute Output and Input Cost Functions
8     [Jout, dJout] = CostFunctionOutput(uc, xi, beta, delta, gamma, muE, ...
9         sigmaE, p, T, P, w);
10    [Jin, dJin] = CostFunctionInput(uc, uPrev, lambda, T);
11
12    % Cost Function
13    J = Jout+Jin;
14    dJ = (dJout+dJin)*T;
15
16 end

```

A-1-4 CostFunctionOutput.m

```

1 function [J, dJ] = CostFunctionOutput(tu, xi, beta, delta, gamma, muE, ...
2     sigmaE, p, T, P, w)
3
4     % Preallocate
5     J = zeros(size(P, 1), 1);
6     dJ = zeros(size(P, 1), size(T, 1));
7
8     %For Every Mode Switching Sequence
9     for j = 1:size(P, 1)
10
11         % Write as Single Random Variable X
12         muX = xi{j} + delta{j}*w + beta{j}*tu + gamma{j}*muE;
13         sigmaX = sqrt(gamma{j}.^2*sigmaE);
14
15         % Preallocate

```

```

15     E = zeros(size(T, 1), 1);
16     dE = zeros(size(T, 1));
17
18     % For every Prediction Step
19     for i = 1:size(T, 1)
20         % Select Appropriate Rows
21         muXi = [muX(delta{j}(:, i) == 1); 0];
22         sigmaXi = [sigmaX(delta{j}(:, i) == 1); 0];
23         betai = [beta{j}(delta{j}(:, i) == 1, :); zeros(1, size(beta{j}, ...
24             2))];
25
26         if p > 0 % Norm Based Approximation Method
27             [E(i), dE(i, :)] = NormBasedApproximation(muXi, sigmaXi, p, ...
28                 betai);
29         else % Monte Carlo Simulation
30             gammai = [gamma{j}(delta{j}(:, i) == 1, :); zeros(1, ...
31                 size(gamma{j}, 2))];
32             [E(i), dE(i, :)] = MonteCarloSimulations(muXi, sigmaXi, ...
33                 betai, gammai);
34         end
35     end
36
37     % Output Cost Function
38     J(j) = sum(E)*P(j);
39     dJ(j, :) = sum(dE)*P(j);
40
41 end
42
43 % Full Output Cost Function
44 J = sum(J);
45 dJ = sum(dJ, 1);
46
47 end

```

A-1-5 CostFunctionInput.m

```

1 function [J, dJ] = CostFunctionInput(tu, uPrev, lambda, T)
2
3 % Input Cost Function
4 J = -lambda*sum(tu-uPrev);
5 dJ = -lambda*ones(1, size(T, 1));
6
7 end

```

A-1-6 NormBasedApproximation.m

```

1 function [E, dE] = NormBasedApproximation(muX, sigmaX, p, beta)
2
3 % Compute Lower Bound
4 L = min([muX-3*sigmaX; 0]);
5 muX = muX - L;
6
7 % Find Indices
8 dIndex = (sigmaX == 0);
9 sIndex = (sigmaX ~= 0);

```

```

10
11 % Calculate Required Hermite Polynomials
12 [Hn Hm] = Hermite(p, 1i*muX(sIndex)./sigmaX(sIndex));
13 % Hn := Hermite(p, 1i*muX(sIndex)./sigmaX(sIndex));
14 % Hm := Hermite(p-1, 1i*muX(sIndex)./sigmaX(sIndex));
15
16 % Compute Expectation
17 Ed = muX(dIndex).^p;
18 Es = sigmaX(sIndex).^p .* 1i^(-p) .* Hn;
19 E = sum([Ed; Es])^(1/p) + L;
20
21 % Compute Subgradient of Expectation
22 dEd = muX(dIndex).^(p-1) * ones(1, size(beta(dIndex, :), 2)) .* ...
    beta(dIndex, :);
23 dEs = sigmaX(sIndex).^(p-1) * 1i^(-p+1) .* Hm * ones(1, size(beta(dIndex, ...
    :), 2)) .* beta(sIndex, :);
24 dE = sum([Ed; Es])^(1/p-1) * sum([dEd; dEs]);
25
26 end

```

A-1-7 Hermite.m

```

1 function [Hn, Hm] = Hermite(n, x)
2
3     switch n % Order of Polynomial
4     case 0
5         % H(0, x) = 1
6         Hn = ones(size(x));
7     case 1
8         % H(1, x) = x
9         Hn = x;
10    otherwise
11        % H(n+1, x) = x*Hermite(n,x) - (n-1)*Hermite(n-1, x)
12        H(:, 1) = Hermite(0, x);
13        H(:, 2) = Hermite(1, x);
14        for ni = 2:n
15            H(:, ni+1) = x.*H(:, ni) - (ni-1).*H(:, ni-1);
16        end
17        Hn = H(:, end);
18        Hm = H(:, end-1);
19    end
20
21 end

```

A-1-8 CostConstraints.m

```

1 function [C, Ceq, dC, dCeq] = CostConstraints(tu, uPrev, T, S)
2
3     % Constraint for Nc (\Delta^2 u(k+j) = 0)
4     uc = T*tu;
5
6     % Inequality Constraints
7     C = S*uc + [uPrev; zeros(size(T, 1) - 1, 1)];
8     dC = T'*S';
9
10    % Equality Constraints

```

```

11     Ceq = [];
12     dCeq = [];
13
14 end

```

A-1-9 ClosedLoopSimulation.m

```

1 function [xkplusone, yk] = ClosedLoopSimulation(xk, uk, ell, d1, d2, d3)
2
3 % Define System Matrices for every Mode
4 if ell == 1
5     A = [d1(1)           -Inf           -Inf ;
6          d1(1)+d1(2)      d2(1)         -Inf ;
7          d1(1)+d1(2)+d2(2) d2(1)+d2(2) d3(1)];
8     B = [0 ;
9          d1(2) ;
10         d1(2)+d2(2)];
11     C = [-Inf -Inf d3(2)];
12 elseif ell == 2
13     A = [d1(1)           d2(1)+d2(2)      -Inf ;
14          -Inf            d2(1)            -Inf ;
15          d1(1)+d1(2)      d1(2)+d2(1)+d2(2) d2(1)];
16     B = [d2(2) ;
17          0 ;
18          d1(2)+d2(2)];
19     C = [-Inf -Inf d3(2)];
20 elseif ell == 3
21     A = [d1(1)           -Inf -Inf ;
22          d1(1)+d1(2)+d3(2) d2(1) d3(1)+d3(2);
23          d1(1)+d1(2)      -Inf d3(1) ];
24     B = [0 ;
25          d1(2)+d3(2);
26          d1(2) ];
27     C = [-Inf d2(2) -Inf];
28 end
29
30 % Compute x(k+1) and y(k)
31 xkplusone = otimes([A B], [xk; uk]);
32 yk = otimes(C, xkplusone);
33
34 end

```

A-1-10 MonteCarloSimulations.m

```

1 function [E, dE] = MonteCarloSimulations(muX, sigmaX, beta, gamma)
2
3 % Compute Lower Bound
4 L = min([muX-3*sigmaX; 0]);
5 muX = muX - L;
6
7 % Monte-Carlo Simulations
8 nMC = 1E6; % Nr. of Monte-Carlo Simultions
9 X = muX * ones(1, nMC) + gamma*randn(size(gamma, 2), nMC);
10 X = max(X - L);
11 E = mean(X) + L;
12 dE = zeros(1, size(beta, 2));

```

```

13
14 end

```

A-2 Linking Matlab and Mathematica

In this section, we will explain how we can use MATHEMATICA commands inside MATLAB. We have used MATHLINK, which allows us to communicate with the MATHEMATICA KERNEL to perform calculations from any program. From [17], we have downloaded and compiled a MEX file. Since we cannot load a file in the MATHEMATICA KERNEL, we need to define the functions from MATLAB. The actual version of MATHEMATICA is 8.0.

A-2-1 MathLink.m

```

1 function [xi beta delta gamma Lcal] = MathLink(L, Np)
2
3     FileName = ['Predictor_Np_' int2str(Np) '.mat'];
4     if exist(FileName, 'file') ~= 0
5         % Use Cached Results
6         load(FileName)
7     else
8         % Find all Possible Switching Sequences
9         Lcal = dec2base(0:L^Np-1, L)-47;
10
11        % Open New MathLink Connection, please select your version of ...
12        'MathKernel.exe'
13        math('');
14
15        % Add Functions to Obtain the Predictor Matrices to Mathematica
16        MathLinkFunctions;
17
18        % Define System Matrices in Mathematica
19        EQ = sprintf(['(* Define System Matrices for Mode 1 *)\n' ...
20                    'A[1, k_] := {{d1[k-1], Epsilon, Epsilon}, ' ...
21                    '{d1[k-1] + d1[k], d2[k-1], Epsilon}, ' ...
22                    '{d1[k-1] + d1[k] + d2[k], d2[k-1] + ...
23                    d2[k], d3[k-1]}};\n' ...
24                    'B[1, k_] := {{0}, ' ...
25                    '{d1[k]}, ' ...
26                    '{d1[k] + d2[k]}};\n' ...
27                    'C[1, k_] := {{Epsilon, Epsilon, d3[k]}};\n' ...
28                    '\n' ...
29                    '(* Define System Matrices for Mode 2 *)\n' ...
30                    'A[2, k_] := {{d1[k-1], d2[k-1] + d2[k], Epsilon}, ' ...
31                    '{Epsilon, d2[k-1], Epsilon}, ' ...
32                    '{d1[k-1] + d1[k], d1[k] + d2[k-1] + ...
33                    d2[k], d3[k-1]}};\n' ...
34                    'B[2, k_] := {{d2[k]}, ' ...
35                    '{0}, ' ...
36                    '{d1[k] + d2[k]}};\n' ...
37                    'C[2, k_] := {{Epsilon, Epsilon, d3[k]}};\n' ...
38                    '\n' ...
39                    '(* Define System Matrices for Mode 3 *)\n' ...
40                    'A[3, k_] := {{d1[k-1], Epsilon, Epsilon}, ' ...

```

```

38             '{d1[k-1] + d1[k] + d3[k], d2[k-1], ...
39                 d3[k-1] + d3[k]}}', ' ...
40             '{d1[k-1] + d1[k], Epsilon, d3[k-1]}};\n' ...
41             'B[3, k_] := {{0}, ' ...
42                 '{d1[k] + d3[k]}}', ' ...
43                 '{d1[k]}};\n' ...
44             'C[3, k_] := {{Epsilon, d2[k], Epsilon}};\n']]);
45
46 % Find predictor vector without constants
47 EQ = sprintf(['PredictorConst[A_, B_, C_, L_] := Module[{M},\n' ...
48             ' M = Predictor[A, B, C, L];\n' ...
49             ' Np = Dimensions[L][[1]];\n' ...
50             ' \n' ...
51             ' For[i = -1, i <= Np, i++,\n' ...
52             '     M = M /. d1[k + i] -> 4;\n' ...
53             '     M = M /. d2[k + i] -> 6;\n' ...
54             '     M = M /. d3[k + i] -> 5 + e3[k + i];\n' ...
55             ' ];\n' ...
56             ' Return[M];\n' ...
57             '];\n']);
58 math(EQ);
59
60 % Preallocate
61 xi = cell(size(Lcal, 1), 1);
62 beta = cell(size(Lcal, 2), 1);
63 delta = cell(size(Lcal, 1), 1);
64 gamma = cell(size(Lcal, 1), 1);
65
66 for i = 1:size(Lcal, 1)
67
68     % Store Lcal(i,:) in Mathematica Variable 'Lcal' and convert to ...
69     integer
70     math('matlab2math', 'Lcal', Lcal(i,:));
71     math('Lcal = IntegerPart[Lcal[[1, All]]];');
72
73     % Compute Constants-Matrix
74     % M0 = math('math2matlab', 'M0[A, B, C, Lcal]') => $Failed
75     M0 = math('M0[A, B, C, Lcal]');
76     M0 = regexprep(M0, '\n', '');
77     M0 = regexprep(M0, '>', ',');
78     M0 = strrep(M0, '{', '[');
79     M0 = strrep(M0, '}', ']');
80     M0 = eval(M0);
81
82     % Compute Variables-Matrix
83     % M = math('math2matlab', 'M[A, B, C, Lcal]')=> $Failed
84     M = math('M[A, B, C, Lcal]');
85     M = regexprep(M, '\n', '');
86     M = regexprep(M, '>', ',');
87     M = strrep(M, '{{', '[');
88     M = strrep(M, '}}', ']');
89     M = strrep(M, '}', ',');
90     M = strrep(M, '{', '');
91     M = eval(M);
92
93     % Define which colums refers to which kind of signals

```

```

93     Var = math('VarM[A, B, C, Lcal]');
94     Var = regexprep(Var, '\n', '');
95     Var = regexprep(Var, '>', '');
96     Var = strrep(Var, ',', ' ');
97     Var = strrep(Var, '{', '{');
98     Var = strrep(Var, '}', '}');
99     Var = eval(Var)';
100
101     % Determine which column of M goes in which variable
102     xi{i} = M0;
103     beta{i} = [];
104     delta{i} = [];
105     gamma{i} = [];
106     for j = 1:length(Var)
107         Type = Var{j}(1);
108         if strcmp(Type, 'u')
109             beta{i} = [beta{i} M(:, j)];
110         elseif or(strcmp(Type, 'r'), strcmp(Type, 'x'))
111             delta{i} = [delta{i} M(:, j)];
112         elseif strcmp(Type, 'e')
113             gamma{i} = [gamma{i} M(:, j)];
114         end
115     end
116 end
117
118 % Duplicate for all initial L0
119 xii = xi; betai = beta; deltai = delta; gammai = gamma; Lcali = Lcal;
120 xi = cell(size(Lcal, 2), 1);
121 beta = cell(size(Lcal, 2), 1);
122 delta = cell(size(Lcal, 2), 1);
123 gamma = cell(size(Lcal, 2), 1);
124 Lcal = cell(size(Lcal, 2), 1);
125 for i = 1:L
126     xi{i} = xii;
127     beta{i} = betai;
128     delta{i} = deltai;
129     gamma{i} = gammai;
130     Lcal{i} = Lcali;
131 end
132
133 % Save Results
134 save(FileName, 'xi', 'beta', 'delta', 'gamma', 'Lcal');
135
136 % Mathematica is Finished, Close MathLink Connection
137 math('quit');
138
139 end
140
141 end

```

A-2-2 MathLinkFunctions.m

```

1 % Initialize
2 EQ = sprintf(['(* Clear Internal Memory *)\n' ...
3             'ClearAll["Global`*"];\n' ...
4             'Unprotect["C"];\n' ...

```

```

5         'Unprotect["D"];\n' ...
6         'Epsilon := -Infinity;\n']]);
7 math(EQ);
8
9 % Define o-plus operator
10 EQ = sprintf(['oplus[A_, B_] := Module[{AB, ii, jj},\n' ...
11             '\n' ...
12             '(* Scalar Addition *)\n' ...
13             'If[And[ArrayDepth[A] <= 1, ArrayDepth[B] <= 1],\n' ...
14             '    AB = Distribute[Max[A, B], Max];\n' ...
15             '    Return[Simplify[AB]];\n' ...
16             '];\n' ...
17             '\n' ...
18             '(* Matrix Addition *)\n' ...
19             'If[Equal[Dimensions[A], Dimensions[B]],\n' ...
20             '    AB = Table[0, {Dimensions[A][[1]]}, ...
21             '    {Dimensions[A][[2]]}];\n' ...
22             '    For[ii = 1, ii <= Dimensions[A][[1]], ii++,\n' ...
23             '        For[jj = 1, jj <= Dimensions[A][[2]], jj++,\n' ...
24             '            AB[[ii, jj]] = Distribute[Max[A[[ii, jj]], B[[ii, ...
25             '            jj]], Max];\n' ...
26             '        ];\n' ...
27             '    ];\n' ...
28             '    Return[Simplify[AB]];\n' ...
29             '];\n' ...
30             ''];\n']]);
31 math(EQ);
32
33 % Define o-times operator
34 EQ = sprintf(['otimes[A_, B_] := Module[{AB, ii, jj, kk},\n' ...
35             '\n' ...
36             '(* Scalar Multiplication *)\n' ...
37             'If[And[ArrayDepth[A] <= 1, ArrayDepth[B] <= 1],\n' ...
38             '    AB = Distribute[Plus[A, B], Max];\n' ...
39             '    Return[Simplify[AB]];\n' ...
40             '];\n' ...
41             '\n' ...
42             '(* Matrix Multiplication *)\n' ...
43             'If[Equal[Dimensions[A][[2]], Dimensions[B][[1]]],\n' ...
44             '    AB = Table[0, {Dimensions[A][[1]]}, ...
45             '    {Dimensions[B][[2]]}];\n' ...
46             '    For[ii = 1, ii <= Dimensions[A][[1]], ii++,\n' ...
47             '        For[jj = 1, jj <= Dimensions[B][[2]], jj++,\n' ...
48             '            AB[[ii, jj]] = Distribute[Max[Table[Distribute[A[[ii, ...
49             '            kk]] + B[[kk, jj]], Max], {kk, Dimensions[A][[2]]}]], ...
50             '            Max];\n' ...
51             '        ];\n' ...
52             '    ];\n' ...
53             '    Return[Simplify[AB]];\n' ...
54             '];\n' ...
55             ''];\n']]);
56 math(EQ);
57
58 % Define \tilde{C} matrix
59 EQ = sprintf(['Ctil[A_, B_, C_, L_] := Module[{k = k, ii, jj, kk, Cm, Ctilde, ...
60             'Np},\n' ...
61             '    Np = Dimensions[L][[1]];\n' ...

```

```

56         ' Ctilde = Table[0, {Np}, {Dimensions[A[1, 1]][[2]]}];\n' ...
57         ' For[ii = 1, ii <= Np, ii++,\n' ...
58         '     Cm = C[L[[ii]], k + ii - 1];\n' ...
59         '     For[jj = ii, jj >= 1, jj--, \n' ...
60         '         Cm = otimes[Cm, A[L[[ii]], k + jj - 1]];\n' ...
61         '     ];\n' ...
62         '     For[kk = 1, kk <= Dimensions[A[1, 1]][[1]], kk++,\n' ...
63         '         Ctilde[[ii, kk]] = Cm[[1, kk]];\n' ...
64         '     ];\n' ...
65         ' ];\n' ...
66         ' Return[Ctilde];\n' ...
67     '];\n' ]]);
68 math(EQ);
69
70 % Define \tilde{D} matrix
71 EQ = sprintf(['Dtil[A_, B_, C_, L_] := Module[{k = k, ii, jj, kk, Dmn, ...
72     Dtilde, Np},\n' ...
73     ' Np = Dimensions[L][[1]];\n' ...
74     ' Dtilde = Table[0, {Np}, {Np}];\n' ...
75     ' For[ii = 1, ii <= Np, ii++,\n' ...
76     '     For[jj = 1, jj <= Np, jj++,\n' ...
77     '         If[ii < jj,\n' ...
78     '             Dmn = Epsilon;\n' ...
79     '             Dtilde[[ii, jj]] = Dmn;\n' ...
80     '         ];\n' ...
81     '         If[ii == jj,\n' ...
82     '             Dmn = otimes[C[L[[ii]], k + ii - 1], B[L[[ii]], k + ii ...
83     '             - 1]];\n' ...
84     '             Dtilde[[ii, jj]] = Dmn[[1, 1]];\n' ...
85     '         ];\n' ...
86     '         If[ii > jj,\n' ...
87     '             Dmn = C[L[[ii]], k + ii - 1];\n' ...
88     '             For[kk = ii, kk > jj, kk--, \n' ...
89     '                 Dmn = otimes[Dmn, A[L[[ii]], k + kk - 1]];\n' ...
90     '             ];\n' ...
91     '             Dmn = otimes[Dmn, B[L[[ii]], k + jj - 1]];\n' ...
92     '             Dtilde[[ii, jj]] = Dmn[[1, 1]];\n' ...
93     '         ];\n' ...
94     '         ];\n' ...
95     '     ];\n' ...
96     ' Return[Dtilde];\n' ...
97     '];\n' ]]);
98 math(EQ);
99
100 % Define \tilde{y} vector
101 EQ = sprintf(['Predictor[A_, B_, C_, L_] := Module[{ii, MaxArg, Yt, Np},\n' ...
102     ' Np = Dimensions[L][[1]];\n' ...
103     ' \n' ...
104     ' (* Generate Input Vector *)\n' ...
105     ' uk = {{u[k]}};\n' ...
106     ' For[ii = 1, ii <= Np - 1, ii++,\n' ...
107     '     uk = Append[uk, {u[k + ii]}];\n' ...
108     ' ];\n' ...
109     ' \n' ...
110     ' (* Generate State Vector *)\n' ...
111     ' If[Dimensions[A[1, k]][[2]] == 2,\n' ...
112     '     xk = {{x1[k - 1]}, {x2[k - 1]}};\n' ...

```

```

111         ];\n' ...
112         If[Dimensions[A[1, k]][[2]] == 3,\n' ...
113         xk = {{x1[k - 1]}, {x2[k - 1]}, {x3[k - 1]}};\n' ...
114         ];\n' ...
115         \n' ...
116         (* Generate Reference Vector *)\n' ...
117         rk = {r[k]};\n' ...
118         For[ii = 1, ii <= Np - 1, ii++,\n' ...
119         rk = Append[rk, {r[k + ii]}];\n' ...
120         ];\n' ...
121         \n' ...
122         (* Calculate Predicted Output *)\n' ...
123         Yt = oplus[otimes[Ctil[A, B, C, L], xk], otimes[Dtil[A, B, ...
124         C, L], uk]];\n' ...
125         \n' ...
126         (* Subtract Reference Signal *)\n' ...
127         Yt = Yt + rk;\n' ...
128         For[ii = 1, ii <= Length[Yt], ii++,\n' ...
129         Yt[[ii, 1]] = Distribute[Yt[[ii, 1]], Max];\n' ...
130         ];\n' ...
131         \n' ...
132         (* Yt = Max[Yt, 0] *)\n' ...
133         Yt = Flatten[Max[Yt]];\n' ...
134         Yt = Distribute[oplus[Yt, 0], Max];\n' ...
135         \n' ...
136         (* Retrieve Arguments of Max-function *)\n' ...
137         MaxArg = Table[0, {Length[Yt]}];\n' ...
138         For[ii = 1, ii <= Length[Yt], ii++,\n' ...
139         MaxArg[[ii]] = Yt[[ii]];\n' ...
140         ];\n' ...
141         \n' ...
142         Return[MaxArg];\n' ...
143     ];\n' ]));
144
145 math(EQ);
146
147 % M0-Matrices
148 EQ = sprintf(['M0[A_, B_, C_, L_] := Module[{M, VarM, CAsp},\n' ...
149 M = PredictorConst[A, B, C, L];\n' ...
150 \n' ...
151 (* Obtain all remaining variables *)\n' ...
152 VarM = Variables[M];\n' ...
153 CAsp = CoefficientArrays[M, VarM];\n' ...
154 Return[Normal[CAsp[[1]]]];\n' ...
155 ];\n' ]));
156
157 math(EQ);
158
159 % M-Matrices
160 EQ = sprintf(['M[A_, B_, C_, L_] := Module[{M, VarM, CAsp},\n' ...
161 M = PredictorConst[A, B, C, L];\n' ...
162 \n' ...
163 (* Obtain all remaining variables *)\n' ...
164 VarM = Variables[M];\n' ...
165 CAsp = CoefficientArrays[M, VarM];\n' ...
166 Return[Normal[CAsp[[2]]]];\n' ...
167 ];\n' ]));
168
169 math(EQ);
170
171

```

```
167 % Var-Matrices
168 EQ = sprintf(['VarM[A_, B_, C_, L_] := Module[{M, VarM, CAsp},\n' ...
169             ' M = PredictorConst[A, B, C, L];\n' ...
170             ' \n' ...
171             ' (* Obtain all remaining variables *)\n' ...
172             ' Return[Variables[M]];\n' ...
173             '];\n']);
174 math(EQ);
```

Bibliography

- [1] B. Heidergott, G. Olsder, and J. van der Woude, *Max Plus at Work*. Princeton University Press, 2006.
- [2] F. Baccelli, G. Cohen, G. Olsder, and J. Quadrat, *Synchronization and Linearity*. John Wiley & Sons, 2001. Freely available at: <http://www-rocq.inria.fr/metalau/cohen/documents/BCOQ-book.pdf>.
- [3] B. De Schutter and T. van den Boom, “Model predictive control for max-plus-linear discrete event systems,” *Automatica*, vol. 37, pp. 1049–1056, July 2001.
- [4] T. van den Boom and B. De Schutter, “Model predictive control for perturbed max-plus-linear systems,” *Systems & Control Letters*, vol. 45, pp. 21–33, Jan. 2002.
- [5] T. van den Boom and B. De Schutter, “Model predictive control for perturbed max-plus-linear systems: A stochastic approach,” *International Journal of Control*, vol. 77, pp. 302–309, Feb. 2004.
- [6] S. Farahani, T. van den Boom, H. van der Weide, and B. De Schutter, “An approximation approach for model predictive control of stochastic max-plus linear systems,” in *Proceedings of the 10th International Workshop on Discrete Event Systems (WODES 2010)*, (Berlin, Germany), pp. 386–391, Aug.–Sept. 2010.
- [7] T. van den Boom and B. De Schutter, “Modelling and control of discrete event systems using switching max-plus-linear systems,” *Control Engineering Practice*, vol. 14, pp. 1199–1211, Oct. 2006.
- [8] T. van den Boom and B. De Schutter, “Stabilizing model predictive controllers for randomly switching max-plus-linear systems,” in *Proceedings of the European Control Conference 2007 (ECC’07)*, (Kos, Greece), pp. 4952–4959, July 2007.
- [9] T. van den Boom and B. De Schutter, “Model predictive control for switching max-plus-linear systems with random and deterministic switching,” in *Proceedings of the 17th IFAC World Congress*, (Seoul, Korea), pp. 7660–7665, July 2008.

- [10] T. van den Boom and B. De Schutter, “Model predictive control for randomly switching max-plus-linear systems using a scenario-based algorithm,” in *Proceedings of the 49th IEEE Conference on Decision and Control*, (Atlanta, Georgia), pp. 2298–2303, Dec. 2010.
- [11] T. van den Boom and B. De Schutter, “A stabilizing model predictive controller for uncertain max-plus-linear systems and uncertain switching max-plus-linear systems,” Tech. Rep. 11-010, Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, Mar. 2011. Accepted for the *8th IFAC World Congress*, Milan, Italy, Aug.–Sept. 2011.
- [12] R. Cuninghame-Green, *Minimax Algebra*, vol. 166 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, Germany, 1979.
- [13] E. F. Camacho and C. A. Bordons, *Model Predictive Control in the Process Industry*. Springer, 1995.
- [14] J. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.
- [15] J. Kingman and S. Taylor, *Introduction to Measure and Probability*. Cambridge University Press, 1966.
- [16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009. Freely available at: http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.
- [17] B. Barrowes, *Mathematica Symbolic Toolbox for MATLAB, version 2.0*. Freely available at: <http://library.wolfram.com/infocenter/MathSource/5344/>.
- [18] B. F. Heidergott, *Max-Plus Linear Stochastic Systems and Perturbation Analysis*. Springer, 2006.

Glossary

List of Acronyms

DES	Discrete Event Systems
LTl	Linear Time-Invariant
MPC	Model Predictive Control
MPA	Max-Plus Algebra
MPL	Max-Plus-Linear
SSMPL	Stochastic Switching Max-Plus-Linear
i.i.d.	independent and identically distributed
pdf	probability density function
jdf	joint density function

