Throughput and Stabilisability Analysis of Mode-Constrained Stochastic Switching Max-Plus Linear Systems

Bart de Jong

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Throughput and Stabilisability Analysis of Mode-Constrained Stochastic Switching Max-Plus Linear Systems

Master of Science Thesis

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

Bart de Jong

June 27, 2022

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

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Throughput and Stabilisability Analysis of Mode-Constrained Stochastic Switching Max-Plus Linear Systems

by

Bart de Jong

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: June 27, 2022

Supervisor(s):

dr.ir. A.J.J. van den Boom (Ton)

ir. A. Gupta (Abhimanyu)

Reader(s):

dr. J.W. van der Woude (Jacob)

Abstract

Switching max-plus linear (SMPL) systems written in max-plus algebra (MPA) form a robust framework to model discrete-event systems (DESs) governed by synchronisation but no concurrency whose behaviour may switch over time. Their evolution is described by a max-plus linear state-space representation that may change by switching modes. In their typical form, switching may depend on the system's previous state, previous mode, a discrete control signal, and exogenous stochastic signals.

In this work, we investigate modelling options and performance analyses for SMPL systems and the application of control to such systems. We propose to model stochastic systems whose mode sequences are constrained in some form as discrete hybrid stochastic automata (DHSAs). For such systems, we offer definitions with which to predict system performance measured in throughput in the form of finite-horizon approximations of a class of asymptotic performance metrics. We validate these approximations of a system's growth rate using a Monte Carlo (MC) method and corresponding statistical analyses. We use these analyses to form stabilisability guarantees for SMPL systems as a function of the growth rate of their reference signal. Lastly, we propose a model predictive control (MPC) framework for stabilising stochastic mode-constrained SMPL systems. We validate these contributions by considering three control cases regarding stabilising mode-constrained deterministic and stochastic SMPL systems under discrete and hybrid control.

Table of Contents

Preface and Acknowledgements

Like all other control engineers and enthusiasts, at Delft Center for Systems and Control (DCSC), we concern ourselves with predicting the future. Not only do we aim to accurately predict future states and events, we also try to manipulate them to our liking. Where regular people would focus these efforts on large-scale and impactful areas such as politics or the stock market, we dedicate significant portions of our lives to optimising production systems or imaginary inland waterway networks [1].

To continue that tradition, I explored the remarkable mathematical concept of constrained stochastic switching discrete-event systems represented in the max-plus algebra. These one and a half years of psychological adventure have been both exhilarating and exhausting. However, above all, it has been enlightening. For instance, I learned that I do not need sunlight to survive, that minus infinity is zero and zero is one, and that one can obtain a Master's degree in the highly mathematical field of control engineering while relying solely on a few alphabets—numbers are for novices. Still, the most obvious lesson I drew from trading in 6 % of my life[1] for this report is that time away from a project is as essential as time spent on a project. It is fair to say that the majority of my 'Eureka'-moments happened either during free time, at my new job, or the day after I took a day off. That is not to say that I have successfully learned to take days off. If anything, in the past year or so, I too often declined invitations from friends because 'I was very close to graduating'. I would like to take this opportunity to express my gratitude to my friends for inviting me nonetheless and to encourage them to keep doing this. Please remind me to read this paragraph if I ever decline invitations because I supposedly cannot find the time. I will then consider reconsidering.

Although all my friends deserve individual acknowledgements, I would like to give prominence to a special one. Diede, you have made this lonely journey in the midst of a pandemic rewarding and genuinely enjoyable. You care so much about what I do, even while you could not care less about constrained stochastic switching discrete-event systems represented in the max-plus algebra. You motivate me to do the things I like to do and finish the things I have to finish through your encouraging words and the promise of all the fun things we will do in the future. I cannot wait to hand in this report and go on our long-awaited holiday trip together.

[1] 1.5 years / 25 years = 0.06. The Master's programme has even entertained me for 10 % of my life.

Additionally, I want to spotlight the people who have improved this report's contents through their stimulating ideas and critical feedback.

Mike and Ruby, although we should have kept up the tradition of presenting our weekly progress to each other, I thoroughly enjoyed our time working together, and I am grateful for all the inspiring max-plus discussions we have had over the months. I trust we can continue our friendship despite having nothing left to talk about.

Abhimanyu, I greatly appreciate how you helped me get up to speed at the beginning of the project and your admirable willingness to explain such complex topics to an utterly confused student. I enjoyed all our meetings together.

Ton, I am envious of your ability to turn discussions on challenging and dry mathematical concepts into pleasant, informative and entertaining ones. Your appreciation of the beauty of max-plus algebra convinced me, too, and motivated me to contribute to the field.

Mr Van der Woude, I acknowledge the significant time and effort it takes you to review my work, and I would like to thank you in advance for your undoubtedly valuable perspective and feedback.

Thank you all,

Bart

Chapter 1

Introduction

The modelling framework of discrete-event systems (DESs) is often employed for situations in which engineers are not concerned with small-scale continuous dynamics but instead care about system behaviour governed by discrete actions, spontaneous natural occurrences or situations in which a particular set of conditions is suddenly met, i.e., *events*. Such conditions can, for instance, relate to the synchronisation of other events. In production systems, two parts can only be joined after they both have been produced in earlier steps. Thus, these events—production of the parts—need to be synchronised as a condition for a future event—joining the parts. These discrete and inherently nonlinear dynamics can be elegantly captured by traversing from the conventional algebra to the max-plus algebra (MPA). A specific type of system described in the MPA, a switching max-plus linear (SMPL) system, offers a way of capturing these dynamics into a single set of matrix equalities that can be used for control purposes. These systems, that may be subject to uncertain dynamics and external constraints, are the subject of this research. Specifically, this report explores their modelling, performance evaluation and stabilisability.

First, this chapter introduces the current state of the art regarding the concepts discussed above and reveals the gaps in the existing research that this thesis aims to fill. Then, Chapter 2 on page 5 provides the reader with a sound mathematical foundation needed to understand the main body of the report. Chapter 3 on page 27 discusses the modelling framework of SMPL systems used throughout this work, after which Chapter 4 on page 39 explains and approximates various performance metrics. Chapter 5 on page 65 builds on these metrics and offers a stabilising model predictive control (MPC) strategy and related conditions for stability. Chapter 6 on page 87 validates these results by investigating a series of case studies through simulations. Finally, Chapter 7 on page 117 reflects on the work outlined in this thesis and recommends directions for future research.

The following two sections of this chapter justify the research direction.

1-1 The State of the Art

Recognised by many, the 1868 article of James Clark Maxwell [2] marked the beginning of the field of control theory. It provides a stability analysis of an apparatus used to maintain a constant rotational velocity of a fictional machine and was often applied to windmills. In the succeeding 150 years, as chemical and mechanical machines became more complex and safety concerns were given higher regard, the field has matured and grown to incorporate many types of mathematical system descriptions. Still, the toolbox we have for analysing nonlinear systems pales compared to that for linear systems. This deficiency, too, is the case for systems whose nonlinearity originates from synchronisation.

During the late seventies, however, researchers made strides to translate such systems to a description that allows for the application of many linear systems theory tools [3]. These systems, now represented using MPA, could be written in the well-known linear state-space representation. At the beginning of the millennium, Van den Boom and De Schutter extended the framework to SMPL systems to improve its modelling potency. As a result of this extension, it could represent systems with discrete and sudden changes in their behaviour through switching. The seminal 2012 work of Van den Boom and De Schutter reviewed their MPC strategy to control these deterministic and stochastic systems within a limited set of achievable throughput [4]. In the last few years, Gupta et al. developed robust controllability and stabilisability frameworks for SMPL systems that offer additional insight into the conditions under which they can be effectively controlled [5, 6].

These efforts, however, all assume restrictive conditions on the systems, such as a strict upper bound on throughput or an unconstrained set of permissible switching sequences. Furthermore, the number of practical frameworks to control such systems is marginal. Section 1-2 discusses the areas of research this thesis aims to explore by proposing research questions. Throughout the work, we review the state of the art with more specificity. At the end of the report, Chapter 7 on page 117 reflects on the efforts and proposes additional research directions.

1-2 The Aim of this Thesis

As discussed in Section 1-1, we identified specific research areas that are yet to be explored. This section defines the scope of this thesis by posing four research questions and corresponding subquestions. These questions are used to guide the theory provided in Chapters 3–6. Chapter 7 on page 117 reflects on these questions and evaluates the answers as submitted by the individual sections.

1-2-1 Formulation of the research questions

The four research questions and supporting subquestions are listed below. The paragraphs thereafter detail the establishment and justification of these questions.

1. What is the state of the art of stabilising deterministic and stochastic SMPL systems?
   (a) What control strategies exist for stabilising SMPL systems?
   (b) What is the achievable set of growth rates under discrete or hybrid control?

2. How can we incorporate mode constraints into a control framework?
   (a) What frameworks exist for modelling mode constraints?
   (b) How can we ensure a control algorithm does not violate these constraints?

3. How can we quantify the performance of SMPL systems?
   (a) What distinctions in throughput metrics can we make?
   (b) How can we calculate, approximate and validate these values?

4. How can we stabilise deterministic and stochastic mode-constrained SMPL systems using discrete and hybrid control?
   (a) Under what conditions can an MPC controller stabilise SMPL systems below their maximum growth rate?
   (b) What is the performance advantage of hybrid control over discrete control?

1-2-2 Justification of the research questions

The report's title shows that this thesis mainly concerns the stabilisation and throughput analysis of constrained stochastic SMPL systems. As a first approach to performing this task, Research Question 1 aims to find the state of the art regarding the stabilisation of SMPL systems. It is meant as a probe for finding existing control strategies and their expected performance. In this report, as will become clear, we use a system's growth rate as a metric for performance, justifying Research Question 1b. These questions are answered throughout the report.

In order to generalise the framework, we aim to include constraints on the systems' mode sequences into the control algorithm. Due to several reasons, many systems may not be allowed to operate in the same mode indefinitely. For example, a production system sometimes needs to change recipes because there is a limited number of parts that are needed for a specific product. Research Question 2a and 2b are used to investigate ways of achieving this incorporation of constraints. Chapter 3 on page 27 is the primary source for answering these questions.

An SMPL system's stability and performance are often related to its growth rate. With Research Question 3 and its subquestions, we investigate ways of quantifying this metric to aid in stability and throughput analyses. Chapter 4 on page 39 investigates these topics.

Finally, Chapter 5 on page 65 and Chapter 6 on page 87 aim to answer Research Question 4 and its subquestions. They analyse conditions for stability and propose a framework that can be used to control SMPL systems. Lastly, they apply the strategy to three case studies to demonstrate its efficacy.

Together, these research questions demarcate the scope of the research and highlight its contributions. They are evaluated in Chapter 7 on page 117.

Chapter 2

Preliminaries

This chapter introduces the concepts of discrete-event systems (DESs) described with max-plus algebra (MPA), model predictive control (MPC), graph theory and Monte Carlo (MC) randomised algorithms. These concepts form the basis of the research in this report, and a general understanding of them is a prerequisite for reading the remaining main-body chapters. The theory on these notions is explained in Section 2-1 on page 6, Section 2-2 on page 13, Section 2-3 on page 16 and Section 2-4 on page 21, respectively.

2-1 The Max-Plus Algebra

Many linear control theory tools rely on a general state-space description of a system. This set of tools includes, for instance, modal decomposition for stability analyses, a large part of Lyapunov theory [7, 8] and controllability studies. If such a linear state-space description is not readily available, for instance, for the DESs introduced below, these tools are useless. This section introduces a mathematical language called max-plus algebra (MPA) to "linearise" a class of DESs and reallow the application of many of these tools. This class of max-plus linear (MPL) systems is characterised by synchronisation and no concurrency or choice.

Consider first the definition of a DES posed by Cassandras [9]. The reader is referred to their work for a broader explanation of DESs and the interpretation of *events*.

Definition 2.1. [9] A discrete-event system (DES) is a discrete-state, event-driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time.

A noteworthy curiosity in modelling these systems is that the state vector generally represents the time instances of event occurrences. For instance, $x_1(k)$ may represent the time instance at which event 1 in cycle k occurred. Then, the time instance of the occurrence of the next cycle $x_1(k+1)$ may occur as soon as a certain delay a after the previous occurrence, such that $x_1(k+1) \geq x_1(k) + a$. Incorporating multiple possible events and assuming no unnecessary delays, the state-update equation of a DES in conventional algebra generally follows the nonlinear form of Eq. (2-1). This chapter demonstrates that these equations can be rewritten in a seemingly linear way, allowing for applying otherwise incompatible tools from linear systems theory.

$$x_i(k+1) = \max_{j=1,\ldots,n} (x_j(k) + a_{ij}), \quad i = 1, \ldots, n, \tag{2-1}$$

Here, the state vector $x(k)$ contains occurrence times for event k, and a_{ij} represents a delay, i.e., an offset to the simultaneity of event occurrences. The time delay between different events in the same event cycle k is often referred to as the system's *buffer level*.

It is essential to realise that synchronisation, modelled through the max-operation, remains nonlinear in conventional algebra. It is, however, its representation in the MPA that appears linear. Unless mentioned otherwise, the ideas and explanations of this section are primarily based on the book by Heidergott et al. [10].

2-1-1 An introduction to the algebra

Denote by $\mathbb{R}_\varepsilon$ the set $\mathbb{R} \cup \{\varepsilon\}$, where $\mathbb{R}$ is the set of real numbers and $\varepsilon \stackrel{\text{def}}{=} -\infty$. As discussed further in Section 2-1-2, ε acts as a zero element in the MPA. For $a, b \in \mathbb{R}_\varepsilon$, define the operations $\oplus$ (pronounced 'oplus') and $\otimes$ (pronounced 'otimes') as

$$a \oplus b \stackrel{\text{def}}{=} \max(a, b) \qquad \text{and} \qquad a \otimes b \stackrel{\text{def}}{=} a + b$$

Then, Eq. (2-1) can be written as

$$x_i(k+1) = \bigoplus_{j=1}^{n} x_j(k) \otimes a_{ij}, \quad i = 1, \ldots, n$$

Notable are the similarities with a general linear expression of the form:

$$x_i(k+1) = \sum_{j=1}^{n} x_j(k) \times a_{ij}, \quad i = 1, \ldots, n,$$

where $\times$ represents multiplication in the conventional sense. This linear expression is the expansion of conventional matrix-vector multiplication Ax, for any $A \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$.

The similarities with these common expressions can be exploited to apply many fundamental linear algebra concepts to systems written in the MPA. These concepts include matrices and their eigenvectors and eigenvalues. The promise of applying these concepts to a broader set of systems has driven researchers to explore the field for over four decades.

During the late seventies, Cunninghame-Green has been influential in developing the MPA theory [3]. Early steps in the development were taken by the same author in [11]. In the nineties, Baccelli et al. [12] and Gunawardena [13] wrote two commendable books on the subject. A more recent book around the MPA is used extensively in this section by Heidergott et al. [10]. A recent survey on the history of the algebra was published in 2018 [14]. The first work to feature its application to a railway network was in 1991, in [15]. To explore its application in this report, the algebra needs to be defined to a greater extent. The following section aims to achieve this goal. First, the definition of the MPA is augmented with a zero and a unit element. Then, algebraic properties, including eigenvalues and eigenvectors, are uncovered.

2-1-2 Properties of the algebra

Recall the zero-element $\varepsilon \stackrel{\text{def}}{=} -\infty$ and define $e \stackrel{\text{def}}{=} 0$ as the unit element so that the following holds:

$$a \oplus \varepsilon = \varepsilon \oplus a = a \qquad \text{and} \qquad a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon$$

for all $a \in \mathbb{R}_\varepsilon$.

The set $\mathbb{R}_\varepsilon$, the operations $\oplus$ and $\otimes$, and the elements defined above constitute the MPA, which is denoted as

$$\mathcal{R}_{\max} = (\mathbb{R}_\varepsilon, \oplus, \otimes, \varepsilon, e)$$

As introduced before, the algebra shares similarities with conventional linear algebra. This property constitutes the convenience of exploiting it in settings where conventional linear algebra is ineffective but the application of linear tools is desirable. The similarities start with the definition of the order of operations. Analogous to conventional algebra, multiplication ($\otimes$) comes before addition ($\oplus$). The following expression exemplifies this order:

$$4 \oplus 3 \otimes 5 = 4 \oplus (3 \otimes 5) = 8$$

Additional algebraic properties of the MPA are defined below. The variables x, y, z are assumed to be in the set $\mathbb{R}_\varepsilon$.

Associativity:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \qquad \text{and} \qquad x \otimes (y \otimes z) = (x \otimes y) \otimes z$$

Commutativity:

$$x \oplus y = y \oplus x \qquad \text{and} \qquad x \otimes y = y \otimes x$$

Distributivity of $\otimes$ over $\oplus$:

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$$

Existence of a zero element:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \oplus \varepsilon = \varepsilon \oplus x = x$$

Existence of a unit element:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \otimes e = e \otimes x = x$$

The zero is absorbing for $\otimes$:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \otimes \varepsilon = \varepsilon \otimes x = \varepsilon$$

Idempotency of $\oplus$:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \oplus x = x$$

Excluding idempotency, all properties also apply to the conventional algebra. Accordingly, the MPA is considered an *idempotent semiring*. This classification is substantiated by the definition used in [10]:

Definition 2.2. A semiring is a nonempty set R endowed with two binary operations $\oplus_R$ and $\otimes_R$, such that

- $\oplus_R$ is associative and commutative with zero element ε_R;
- $\otimes_R$ is associative, distributes over $\oplus_R$, and has unit element e_R;
- ε_R is absorbing for $\otimes_R$.

A semiring is denoted by $\mathcal{R} = (R, \oplus_R, \otimes_R, \varepsilon_R, e_R)$. If $\otimes_R$ is commutative, then $\mathcal{R}$ is called commutative, and if $\oplus_R$ is idempotent, then it is called idempotent.

2-1-3 Matrices and spectral theory

The max-plus algebraic framework is also applicable to matrices. Consider the following succinct set of properties. For all matrices $A \in \mathbb{R}_\varepsilon^{q \times r}$ and $B \in \mathbb{R}_\varepsilon^{q \times r}$, their sum in the MPA is defined by

$$[A \oplus B]_{ij} = a_{ij} \oplus b_{ij}$$

for $i \in \{1, \ldots, q\}$ and $j \in \{1, \ldots, r\}$. In this report, $\mathbb{R}^q$ denotes the vector space of q-tuples of real numbers. Similarly, $\mathbb{R}^{q \times q}$ is the vector space of $q \times q$ real matrices. For any $\alpha \in \mathbb{R}_\varepsilon$, the scalar multiple is defined by

$$[\alpha \otimes A]_{ij} = \alpha \otimes A_{ij}$$

The max-plus Schur product, denoted by "$\odot$", is defined as

$$[A \odot B]_{i,j} = [A]_{i,j} + [B]_{i,j}$$

For matrices $A \in \mathbb{R}_{\varepsilon}^{q\times r}$ and $B \in \mathbb{R}_{\varepsilon}^{r\times p}$, the matrix product is defined as

$$\begin{aligned}[A \otimes B]_{ik} &= \bigoplus_{j=1}^{r} a_{ij} \otimes b_{jk} \\ &= \max_{j\in\{1,\dots,r\}} \{a_{ij} + b_{jk}\}\end{aligned}$$

for $i \in \{1,\dots,q\}$ and $k \in \{1,\dots,p\}$. Notice the analogy with conventional algebra, where $\oplus$ would be replaced with $+$ and $\otimes$ with $\times$.

A zero-matrix in the conventional algebra is analogous to the matrix $\mathcal{E}(q,r)$ in the MPA, whose elements are equal to ε. The identity matrix has the $E(q,r)$-matrix as its counterpart and has e along its main diagonal and ε elsewhere. The definition can also be extended to nonsquare matrices. These two matrices exhibit the following two properties:

$$\begin{aligned}A \oplus \mathcal{E}(q,r) &= \mathcal{E}(q,r) \oplus A &&= A \\ A \otimes E(q,q) &= E(q,q) \otimes A &&= A\end{aligned}$$

for any matrix $A \in \mathbb{R}_{\varepsilon}^{q\times r}$. Additionally, for $k \geq 1$, also the following hold:

$$\begin{aligned}A \otimes \mathcal{E}(r,k) &= \mathcal{E}(q,k) \\ \mathcal{E}(k,q) \otimes A &= \mathcal{E}(k,r)\end{aligned}$$

The concept of eigenvalues and corresponding eigenvectors also exists in the MPA. The definition is analogous to the conventional one, where, for eigenvalue λ and eigenvector v, the following holds:

$$A \otimes v = \lambda \otimes v$$

For any matrix A with size $n \times n$, eigenvector v has n entries that are not all equal to ε. Recall that $\lambda \otimes v$ denotes the element-wise addition of λ to the entries in v. The most straightforward way of obtaining the unique eigenvalue of an irreducible matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ is by considering that it is equal to the finite maximal average circuit weight of the strongly connected communication graph $\mathcal{G}(A)$ [12, Theorem 3.23]. For a general matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$, any eigenvalue is equal to some cycle mean. As a result, the largest eigenvalue coincides with the maximum cycle mean.

Lastly, powers translate to the MPA in the following way:

$$x^{\otimes q} \stackrel{\text{def}}{=} \underbrace{x \otimes x \otimes \dots \otimes x}_{q \text{ times}}$$

for every $x \in \mathbb{R}_{\varepsilon}$ and for all $q \neq 0 \in \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. For $q = 0$, we define $x^{\otimes 0} \stackrel{\text{def}}{=} e = 0$. The reader is advised to consult the book by Heidergott et al. [10] to attain a deeper understanding of the topics discussed in this section.

2-1-4 Norms in the max-plus algebra

We define various norms in the MPA that will help quantify the size of vectors and matrices. Please note that, more so than for conventional algebra, these norms tend to be infinite due to the nature of the max-plus zero element ε.

The projective norm The *projective norm*, also, Hilbert's projective norm, is defined for vectors $x \in \mathbb{R}_\varepsilon^n$ in the MPA as [10, Sec. 4.4]:

$$\|x\|_{\mathbb{P}} \stackrel{\text{def}}{=} \bigoplus_{i=1}^{n} x_i \otimes \bigoplus_{i=1}^{n} (-x_i) = \max_{i \in \underline{n}} x_i - \min_{i \in \underline{n}} x_i$$

For matrices $A \in \mathbb{R}_\varepsilon^{n \times m}$, the norm is defined as

$$\|A\|_{\mathbb{P}} = \max \{ \|[A]_{\cdot i}\|_{\mathbb{P}} \mid i \in \underline{m} \}$$

Furthermore, we have that $\|x + y\|_{\mathbb{P}} \leq \|x\|_{\mathbb{P}} + \|y\|_{\mathbb{P}}$.

The supremum norm The *supremum norm*, also, l^∞-norm, is defined for vectors $x \in \mathbb{R}_\varepsilon^n$ as the maximum of the absolute value of all entries of x [10, Sec. 3.2.1]:

$$\|x\|_\infty \stackrel{\text{def}}{=} \max_{i \in \underline{n}} |x_i|.$$

Minimal finite element We denote the minimal finite entry of matrix $A \in \mathbb{R}_\varepsilon^{n \times m}$ as [10, Sec. 11.2]:

$$\|A\|_{\min} \stackrel{\text{def}}{=} \min_{i,j} \{ a_{i,j} \mid a_{i,j} \neq -\infty,\ i \in \underline{n},\ j \in \underline{m} \}$$

Maximal finite element We denote the maximal finite entry of matrix $A \in \mathbb{R}_\varepsilon^{n \times m}$ as [10, Sec. 11.2]:

$$\|A\|_{\max} \stackrel{\text{def}}{=} \max_{i,j} \{ a_{i,j} \mid a_{i,j} \neq +\infty,\ i \in \underline{n},\ j \in \underline{m} \}$$

2-1-5 Max-plus linear systems

DESs with a linear max-plus algebraic description are called max-plus linear (MPL) systems. These are systems that display synchronisation and no concurrency. The MPL-framework cannot capture the evolution of a system in which concurrency governed by the choice of an external user occurs. This framework has the following general deterministic form [12]:

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

where $A \in \mathbb{R}_\varepsilon^{n_x \times n_x}$, $B \in \mathbb{R}_\varepsilon^{n_x \times n_u}$ and $C \in \mathbb{R}_\varepsilon^{n_y \times n_x}$. It describes a time-invariant system, i.e., a system that exhibits constant behaviour over time. In this description, k is called the event counter, and the entries in $x(k)$ and $y(k)$ are event times. $u(k)$ represents an external delay

imposed on the system. The system matrices A, B and C contain processing times, similar to the delays in Eq. (2-1) on page 6.

An autonomous MPL system is one that evolves without external inputs and can thus be described by the simplified form

$$\begin{aligned} x(k+1) &= A \otimes x(k) \\ y(k) &= C \otimes x(k). \end{aligned}$$

For some systems, the values in the system matrices may change over time. Illustratively, this may happen for a railway network whose trains have a stricter speed limit during rainfall. In such a case, the system matrices of the uncertain MPL system are time-varying and the description changes to:

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

Note here that the dependence of the matrices on k does not explicitly allow time-driven changes, only event-driven changes. The description thus cannot account for unforeseen rainfall while the trains are on the move. However, it illustrates that the system dynamics can change from one event to the next.

Uncertainty in the dynamics due to, for example, rainfall, is modelled using a stochastic variable $e(k)$ [16]. The deterministic model of Eq. (2-2) is thus generalised to include this parametric uncertainty:

$$\begin{aligned} x(k+1) &= A(e(k)) \otimes x(k) \oplus B(e(k)) \otimes u(k) \\ y(k) &= C(e(k)) \otimes x(k) \end{aligned} \tag{2-3}$$

A system represented by Eq. (2-3) is called a max-plus linear parameter-varying (MP-LPV) system [17].

Explicit state-update equations, e.g., the one in Eq. (2-2), offer a direct way to calculate $x(k+1)$. However, in many systems, one finds that the updated state vector $x(k+1)$ is a function of itself and previous state vectors. In general, systems are described by the implicit recurrence relation [10]

$$x(k+1) = \bigoplus_{i=0}^{M} (A_i \otimes x(k+1-i)) \oplus B \otimes u(k) \tag{2-4}$$

where $M \geq 0$ represents the number of past cycles on which the updated state depends.

Implicit models are difficult to solve analytically or simulate. Often, repeated substitution is necessary to find the solution. The following lemma and theorem provide a basis for transforming the description into an explicit one.

Lemma 2.1. [10, Lemma 2.2] Let $A \in \mathbb{R}_{\varepsilon}^{n \times n}$ be such that any circuit in $\mathcal{G}(A)$ has an average circuit weight less than or equal to e. Then, it holds that

$$A^{+} = \bigoplus_{k=1}^{\infty} A^{\otimes^k} = A \oplus A^{\otimes^2} \oplus A^{\otimes^3} \oplus \ldots \oplus A^{\otimes^n} \in \mathbb{R}_{\varepsilon}^{n \times n}.$$

Here, $G(A)$ is the communication graph of matrix A. Section 2-3 on page 16 details the graph theory surrounding this concept.

Now, define A^* as

$$A^* \stackrel{\text{def}}{=} E \oplus A^+ = \bigoplus_{k \geq 0} A^{\otimes^k}.$$

Then, the following theorem holds:

Theorem 2.1. [10, Theorem 2.10] Let $A \in \mathbb{R}_\varepsilon^{n \times n}$ and $b \in \mathbb{R}_\varepsilon^n$. If the communication graph $\mathcal{G}(A)$ has maximal average circuit weight less than or equal to e, then the vector $x = A^* \otimes b$ solves the equation $x = (A \otimes x) \oplus b$. Moreover, if the circuit weights in $\mathcal{G}(A)$ are negative, then the solution is unique.

Heidergott et al. rewrite A_0 in Eq. (2-4) on page 11 according to Lemma 2.1 as [10, p. 82]

$$A_0^* = \bigoplus_{i=0}^{n-1} {A_0}^{\otimes^i},$$

given that A_0 has circuit weights less than or equal to zero.

Next, for

$$b(k) = \bigoplus_{i=1}^{M} A_i \otimes x(k+1-i),$$

Eq. (2-4) on page 11 reduces to

$$x(k+1) = A_0 \otimes x(k+1) \oplus b(k). \tag{2-5}$$

Theorem 2.1 dictates that Eq. (2-5) can be written as

$$\begin{aligned} x(k+1) &= A_0^* \otimes b(k) \\ &= A_0^* \otimes \bigoplus_{i=1}^{M} A_i \otimes x(k+1-i) \\ &= A_0^* \otimes A_1 \otimes x(k) \oplus \ldots \oplus A_0^* \otimes A_M \otimes x(k+1-M). \end{aligned} \tag{2-6}$$

These steps transform the implicit form of Eq. (2-4) on page 11 to the explicit form of Eq. (2-6). The M-th order recurrence relation of Eq. (2-6) can be reduced to a first-order relation if needed.

A more general way to model DESs with synchronisation but no concurrency is through switching max-plus linear (SMPL) systems, discussed in Chapter 3 on page 27. Also, the chapter introduces the modelling of uncertainty in the system's switching behaviour.

This chapter has omitted many aspects of the algebra for brevity. The referenced literature is a good starting point for further exploring the field and its capabilities. The following section discusses the acclaimed control algorithm model predictive control (MPC) and its application to dynamical systems represented in the MPA.

2-2 Model Predictive Control

During the 1970s, a need emerged for more powerful controllers in industrial applications. Until then, no systematic control algorithms existed that could respect hard constraints on the process to be controlled [18]. The fundamentals for a new control strategy, MPC, were laid in subsequent years. Cutler and Ramakers [19] and Richalet et al. [20] are recognised for their early influential work on MPC. In recent years, many industrial and academic applications have widely adopted the control method. Virtually every refinery and petrochemical plant exploits the algorithm [18]. Figure 2-1 suggests that the control structure is more popular than ever, and its popularity is still on the rise. Its data originate from Google's *Ngram viewer* [21], which uses their vast collection of books to investigate the percentage of 3-grams equal to 'model predictive control'. Here, a 3-gram is a contiguous set of three words. The tool also shows that MPC has overtaken 'pole placement' around 2000 and 'linear quadratic gaussian' or 'linear quadratic regulator' a few years earlier. This popularity analysis is hardly impeccable; however, it does show the impact of MPC on the industry.

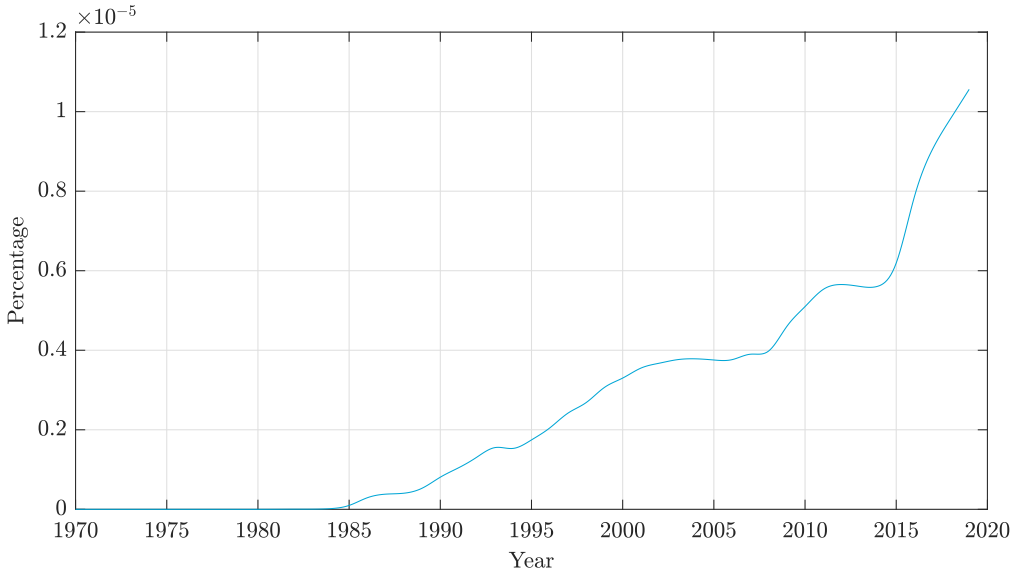

Figure 2-1: Percentage of 3-grams equal to 'model predictive control' in books of the English corpus of *Google books Ngram viewer* [21]. The data are smoothed using a smoothing spline.

The strength of MPC lies partially in its use of a system's mathematical model. Due to the incorporation of a model, the controller can optimise the system's performance within some predetermined bounds. The bounds may define the extremities of the system's output but can also represent limits on the actuator input. Within the MPC's optimisation problem, a cost function defines the optimality of a given solution. This section discusses the basic setup of the MPC algorithm and does not explore the field's borders, nor does it examine the state of the art. [22, 23, 24, 25] discuss MPC in more detail.

Consider the following state-space model describing a system with n_x states, n_u inputs and n_y outputs:

$$\begin{aligned} x(k+1) &= A\,x(k) + B\,u(k), \\ y(k) &= C\,x(k) + D\,u(k) \end{aligned} \tag{2-7}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ is the input vector, and $y \in \mathbb{R}^{n_y}$ is the output vector. A linear system modelled by Eq. (2-7) on page 13 is called a plus-times linear (PTL) system. It is the conventional counterpart to the MPL system.

In the MPC framework, the evolution of a model is captured over a prediction horizon N_p. The optimisation algorithm weighs the cost of outputs against the cost of inputs, often using a cost function of the form:

$$J(k) = J_\mathrm{out}(\tilde{u}) + \lambda\, J_\mathrm{in}(\tilde{u})$$

The function $J(k)$ represents the cost of applying a set of inputs $\tilde{u}$ over a period $[k,\ k+N_\mathrm{p}-1]$. The nonnegative scalar λ weighs the cost of output reference error against the actuator cost. The function representing the output cost, $J_\mathrm{out}(\tilde{u})$, is based on the predicted state evolution of the system over the prediction horizon. Generally, for PTL systems, the functions J_out and J_in are chosen in the following way [26]:

$$J_\mathrm{out} = \sum_{j=1}^{N_\mathrm{p}} ||\hat{y}(k+j \mid k) - r(k+j)||^2$$

$$J_\mathrm{in} = \sum_{j=1}^{N_\mathrm{p}} ||u(k+j-1)||^2$$

where $\hat{y}(k+j \mid k)$ is the output estimation of the system at time step $k+j$ given the information available at time step k. The signal $r(k+j)$ represents the reference output at time step $k+j$. The value of N_p is often chosen in such a way as to balance the processing load against prediction power. A longer prediction horizon results in a prediction of the system's behaviour over an extended period. Contrarily, it burdens the processor by increasing the number of optimisation variables. One could incorporate a control horizon N_c to limit the processing load while preserving prediction power over a longer period. Often, the input is assumed to be constant after time step $k+N_\mathrm{c}$: $u(k+j) = u(k+N_\mathrm{c}-1)$ for $j = N_\mathrm{c},\ \ldots,\ N_\mathrm{p}-1$ in order to decrease the number of optimisation variables.

At every time step, a cost function J is minimised over the control variables in the set $\tilde{u}$. Then, only the first element of the set of control actions is applied to the system and the two-step process repeats. This repetitive process justifies the alternative name of MPC; *receding horizon control*.

Figure 2-2 presents a schematic view of the workings of an MPC. The control loop in the figure is executed once every time step k. Note that all future inputs $\tilde{u}$ are used in the model for predicting the system's behaviour over N_p time steps. However, only the input of one time step is fed into the actual system. The figure is based on Figure 1.2 from [23].

As introduced, an advantage of MPC over control algorithms such as pole-placement, linear-quadratic-gaussian (LQG) control and robust algorithms such as H_∞, is the possible inclusion of constraints. Bounds on the input and output are typically considered as constraints. Additionally, the change of the input signal is often restricted:

$$\begin{aligned} u_\mathrm{min} \leq\ & u(k) && \leq u_\mathrm{max}, && \forall\, k \\ du_\mathrm{min} \leq\ & \Delta u(k) && \leq du_\mathrm{max}, && \forall\, k \\ y_\mathrm{min} \leq\ & \hat{y}(k) && \leq y_\mathrm{max}, && \forall\, k \end{aligned}$$

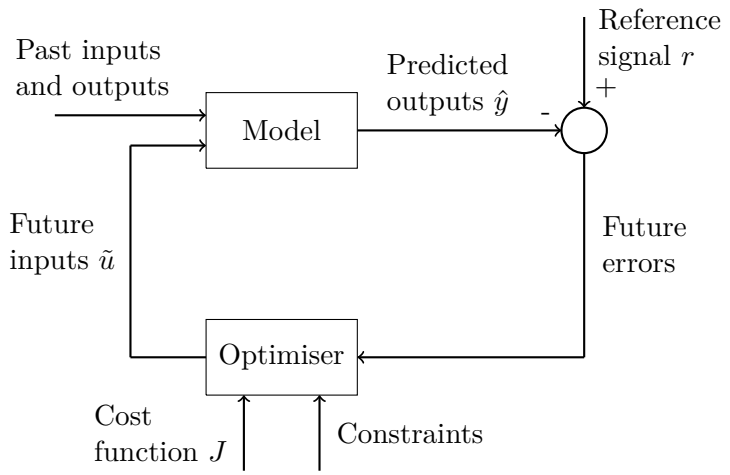

Figure 2-2: Schematic structure of an MPC, based on Figure 1.2 from [23]

where $\Delta u(k) = u(k) - u(k-1)$. MPC allows for the inclusion of any type of constraint. However, linear constraints are computationally the most efficient. They are modelled as:

$$E(k)\tilde{u}(k) + F(k)\tilde{y}(k) \leq h(k)$$

Here, $\tilde{y}(k)$ is a collection of all $\hat{y}$ over prediction horizon N_p and $\tilde{u}(k)$ as defined before:

$$\tilde{y}(k) = \begin{pmatrix} \hat{y}(k+1 \mid k) \\ \vdots \\ \hat{y}(k+N_\mathrm{p} \mid k) \end{pmatrix}, \quad \tilde{u}(k) = \begin{pmatrix} u(k) \\ \vdots \\ u(k+N_\mathrm{p}-1) \end{pmatrix}$$

The set $\tilde{y}(k)$ can be obtained as a function of $\tilde{u}(k)$ through repetitive substitution of both parts of Eq. (2-7) on page 13 [23]:

$$\tilde{y}(k) = H\tilde{u}(k) + g(k)$$

with

$$H = \begin{pmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_\mathrm{p}-1} & CA^{N_\mathrm{p}-2}B & \dots & CB \end{pmatrix} \quad \text{and} \quad g(k) = \begin{pmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_\mathrm{p}} \end{pmatrix} x(k)$$

The resulting optimisation problem is convex when using a quadratic cost function J and a linear set of constraints. Many efficient algorithms exist for solving convex problems, such as the cutting plane algorithm [27], the ellipsoid algorithm [27] and the interior-point algorithm [28]. Apart from describing the model and its constraints, the MPC tuning parameters are N_p, N_c and λ. N_p should be chosen such that the interval $(1,\, N_\mathrm{p})$ contains the system's crucial dynamics. The control horizon N_c is by definition at most equal to N_p and usually matches the system order. λ defines the trade-off between the output error and actuator input and is often selected as small as possible while avoiding a destabilising controller [26].

Many perks of MPC for PTL systems carry over to systems with an MPL description. The translation of MPC to these systems is detailed in Section 5-3 on page 72.

2-3 Graph Theory and Discrete Automata

Two additional concepts support the theory outlined in this thesis: graph theory and discrete automata. Many of the concepts within the DES theory mentioned before can be related to graph theory. In some cases, graph theory offers tools to do system analyses otherwise impossible. The section on graph theory is primarily based on the work by Heidergott et al. [10]. Furthermore, the framework of discrete automata offers the possibility of modelling discrete behaviour of systems. More specifically, they may aid in including constraints on the systems' mode sequences, as further illustrated in Section 3-3 on page 34. Both concepts are explored in this section.

2-3-1 General graph theory

A *directed graph* $\mathcal{G}(\mathcal{A})$ associated with matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$, also called the *communication graph* of A, consists of the pair $(V(A),\, E(A))$. Here, $V(A) = \underline{n}$ is the finite set of *vertices*, or nodes, of the graph, where $\underline{n}$ is the set of all positive integers up to n. The set of ordered pairs of vertices of the graph, called *arcs*, is denoted by $E(A)$.

If $[A]_{j,i} \neq \varepsilon$, then $\mathcal{G}(A)$ contains an arc $(i,\, j) \in E(A)$ from vertex i to vertex j. Specifically, the graph contains an *incoming arc* at j and an *outgoing arc* at i. The distinction between the arcs $(i,\, j)$ and $(j,\, i)$ justifies the name *directed* graph. Furthermore, a directed graph is *weighted* if a weight $w(i,\, j)$ is associated with each arc in the set $E(A)$. A sequence of arcs connecting any two vertices i and j is called a *path* from vertex i to vertex j. In the special case that $i = j$, the path is called a *circuit*.

A graph $\mathcal{G}(A)$ is called *strongly connected* if every vertex $j \in V(A)$ is reachable from any vertex $i \in V(A)$, i.e., if there is a path from i to j for all $i,\, j \in V(A)$. A matrix A associated with a strongly connected communication graph $\mathcal{G}(A)$ is called *irreducible*.

If a path exists from i to j, then vertex i *communicates with* vertex j. If a graph is not strongly connected, not all vertices in $V(A)$ communicate with each other. In that case, it is possible to subdivide $V(A)$ into a set with vertices that communicate with vertex i, and a set with vertices that do not. Then, contrary to the second subset, all vertices in the first subset must communicate with each other. Repeated subdivision of this set results in a partitioning $V_1(A) \cup V_2(A) \cup \cdots \cup V_r(A) = V(A)$ where each $V_q(A),\ q \in \underline{r}$, is a subset with vertices that communicate only with each other. These partitions are called *classes*.

The graph $\mathcal{G}$ can be partitioned correspondingly, resulting in $\mathcal{G} = (V_q(A),\, E_q(A)),\ q \in \underline{r}$, where $E_q(A)$ is the subset of $E(A)$ of arcs that have both the begin vertex and end vertex in $V_q(A)$. For every reducible matrix A, there exists a max-plus permutation matrix P such that its Frobenius normal form has the following structure [29]:

$$P \otimes A \otimes P^{\otimes^{-1}} = \tilde{A} = \begin{bmatrix} A_{11} & \varepsilon & \dots & \varepsilon \\ A_{21} & A_{22} & \dots & \varepsilon \\ \vdots & \vdots & \ddots & \vdots \\ A_{r1} & A_{r2} & \dots & A_{rr} \end{bmatrix}$$

where $A_{11},\, \dots,\, A_{rr}$ are irreducible submatrices of $\tilde{A}$. The partitions of $V(A)$ corresponding to the submatrices $A_{11},\, \dots,\, A_{rr}$ are $V_1(A),\, \dots,\, V_r(A)$. As a result of the permutation, an

arc from a vertex in the class $V_i(A)$ to a vertex in $V_j(A)$ exists only if $i \leq j$. The classes with no incoming arcs are called *initial classes*, and those without outgoing arcs are *final classes*. In the Frobenius normal form, $V_1(A)$ is an initial class, and $V_r(A)$ is a final class.

2-3-2 Discrete finite automata

As discussed, the framework of deterministic finite automata (DFAs) allows for modelling discrete system behaviour. To illustrate this, consider first their formal definition:

Definition 2.3. [30, Def. 1.4.2] An *automaton* is defined by the triple $\Sigma = (\mathcal{Q}, \mathcal{U}, \phi)$ with

- $\mathcal{Q}$ a finite or countable set of discrete states;
- $\mathcal{U}$ a finite or countable set of discrete inputs or the input alphabet;
- ϕ: $\mathcal{Q} \times \mathcal{U} \to P(\mathcal{Q})$ a partial *transition function*.

If $\mathcal{Q}$ and $\mathcal{U}$ are finite, we speak of a *finite automaton*.

Given a state q and an input u, the transition function $\phi(q, u)$ defines the set of possible next states. If this set is undefined or has zero or one element for each combination of (q, u), the finite automaton is a deterministic finite automaton (DFA). An automaton is often called a finite state machine (FSM), representing its discrete states.

The DFA can conveniently be regarded as a strongly connected, directed and labelled graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{Q}$ is the set of the automaton's states and $\mathcal{E} = \mathcal{U} \subset \mathcal{V} \times \mathcal{V}$ is the set of labelled edges [31]. The labelled edges in the graph imply the transition function. See Figure 2-3 for a deterministic finite automaton representing the switching behaviour of an exemplary bimodal system. It depicts a deterministic bimodal system that may not stay in mode 1 for more than one step but may always stay in mode 2. Figure 2-4 on page 18 shows a nondeterministic finite automaton (nDFA) for a bimodal system where one state-input pair leads to two possible outcomes, namely $\phi(1, 2) = \{1, 2\}$.

Often, an input to the automaton, i.e., a label on an edge, is called a *letter*. A sequence of inputs forms a *word*, and the set of all admissible words forms a *regular language*. A word is admissible if it can be formed by the succession of letters on a path in G.

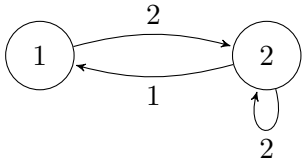

Figure 2-3: A DFA of an exemplary constrained bimodal system that is not allowed to stay in mode 1 for more than one step.

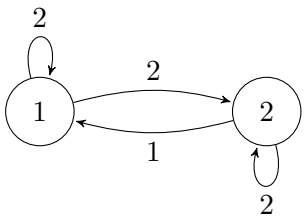

Figure 2-4: An nDFA of an exemplary constrained bimodal system that is undefined for the input '2' at state 1.

2-3-3 Discrete hybrid automata

First introduced by [32], a discrete hybrid automaton (DHA) originated from the connection between an FSM responsible for modelling the discrete states and a switched affine system (SAS) responsible for the continuous states. The two parts are connected through an event generator (EG) and a mode selector (MS); see Figure 2-5. Together, the elements and their interconnection provide a way of modelling various levels of system behaviour. They may combine the evolution of switched systems with additional discrete behaviour governed by DFAs.

Below, the four elements of a DHA are explained.

Switched affine system (SAS) In the DHA framework, a switched affine system (SAS) is a collection of linear affine systems

$$\begin{aligned} x_r(k+1) &= A_{i(k)}x_r(k) + B_{i(k)}u_r(k) + f_{i(k)} \\ y_r(k) &= C_{i(k)}x_r(k) + D_{i(k)}u_r(k) + g_{i(k)} \end{aligned}$$

where $x_r \in \mathcal{X}_r$ is the continuous state vector, $u_r \in \mathcal{U}_r$ is the exogenous continuous input vector, $y_r \in \mathcal{Y}_r$ is the continuous output vector, and the mode $i(k) \in \mathcal{I}$ is a discrete input signal. If necessary, reset maps may be added to the model to alter the system state after transitions.

Event generator (EG) The event generator (EG) generates a signal $\delta_e \in \mathcal{D} \subseteq \{0, 1\}^{n_c}$ according to an affine constraint

$$\delta_e(k) = f_{\text{EG}}(x_r(k),\, u_r(k),\, k)$$

where $f_{\text{EG}} \colon \mathcal{X}_r \times \mathcal{U}_r \times \mathbb{Z}_{\geq 0} \to \mathcal{D}$ is a vector of descriptive functions of a linear hyperplane, and $\mathbb{Z}_{\geq 0}$ is the set of nonnegative integers.

Finite state machine (FSM) A finite state machine (FSM), or automaton, differs from a hybrid automaton (HA) by the exclusion of the continuous state dynamics. Instead, its discrete state evolves according to the following state update function:

$$x_b(k+1) = f_{\text{FSM}}(x_b(k),\, u_b(k),\, \delta_e(k)) \tag{2-8}$$

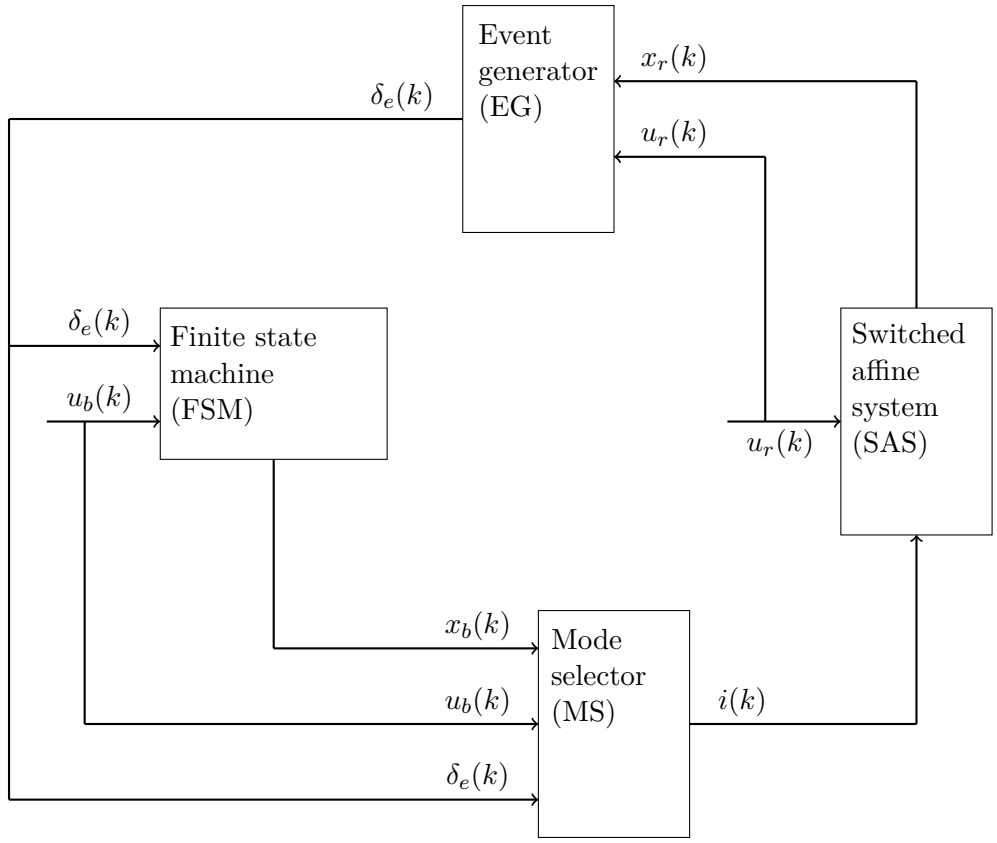

Figure 2-5: A discrete hybrid automaton that consists of an FSM and a SAS, connected through an EG and an MS. The output signals are omitted. Adopted from [32].

where $x_b \in \mathcal{X}_b \subseteq \{0, 1\}^{n_b}$ is the binary state vector denoting the active state of the FSM, $u_b \in \mathcal{U}_b \subseteq \{0, 1\}^{m_b}$ is an exogenous binary input and $f_{\text{FSM}} \colon \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{X}_b$ is a deterministic logic function. It may also have a Boolean output

$$y_b(k) = g_{\text{FSM}}(x_b(k), u_b(k), \delta_e(k))$$

where $y_b \in \mathcal{Y}_b \subseteq \{0, 1\}^{p_b}$ and $g_{\text{FSM}} \colon \mathcal{X}_r \times \mathcal{U}_r \times \mathcal{D} \to \mathcal{Y}_b$ is a logic output function.

Mode selector (MS) The mode selector (MS) converts the logic state $x_b(k)$, the Boolean inputs $u_b(k)$ and the events $\delta_e(k)$ to a dynamic mode $i(k)$ through a function $f_{\text{MS}} \colon \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \alpha(\mathcal{I})$, where α is the set of the binary codings of the elements of $\mathcal{I}$. The output of this function determines the active mode:

$$i(k) = f_{\text{MS}}(x_b(k), u_b(k), \delta_e(k))_{\text{MS}}$$

Note that mode switches can only occur at sampling instances. In the case of a switch at step k, $i(k) \neq i(k-1)$.

2-3-4 Discrete hybrid stochastic automata

Bemporad and Di Cairano extended the framework of DHAs to discrete hybrid stochastic automata (DHSAs) that take into account stochastic mode switching [33]. The framework

consists of the same three elements SAS, EG and MS, but with a stochastic version of the FSM, the stochastic finite state machine (sFSM). Specifically, Eq. (2-8) on page 18 is adjusted to reflect a probability of switching:

$$P\left[x_b(k+1) = \hat{x}_b\right] = f_{\text{sFSM}}(x_b(k),\, u_b(k),\, \delta_e(k),\, \hat{x}_b)$$

where $f_{\text{sFSM}} \colon 0, 1^{2n_b+m_b+n_e} \to [0,\, 1]$. Again, $x_b \in \{0,\, 1\}^{n_b}$ is the binary state vector denoting the active state of the sFSM, $u_b \in \{0,\, 1\}^{m_b}$ is a vector of exogenous binary inputs, and δ_e is the binary output of the EG.

2-4 Monte Carlo Randomised Algorithms

A randomised algorithm is one that employs some form of randomness in its logic. That is, for equal inputs, the algorithm may produce different results that may even be incorrect. The most common type of such algorithms is the MC randomised algorithm, formally defined as:

Definition 2.4. [34, Def. 10.1] A Monte Carlo (MC) randomised algorithm is a randomised algorithm that may produce an incorrect result, but the probability of such an incorrect result is bounded.

These MC randomised algorithms are often used for numerical integration, for instance, when calculating the expectation of random variables whose probability density functions (PDFs) are challenging to integrate analytically. The method was pioneered by Metropolis and Ulam [35], but a more recent book by Tempo et al. is the primary source of information for this section [34]. The next two sections provide an introduction to mean estimation of random variables using the MC method, and to quantifying the performance of random systems.

2-4-1 Monte Carlo method for estimating the expectation of random variables

Consider a random variable X and corresponding PDF $f(x)$, whose expected value $\mathbb{E}[X]$ is defined as

$$\mathbb{E}[X] = \int x f(x)\,\mathrm{d}x$$

Consider generating N samples of X according to the PDF $f(x)$ and storing them as:

$$X_{1\ldots N} \stackrel{\text{def}}{=} \{X_1, \ldots, X_N\}$$

The set $X_{1\ldots N}$ is called a *multisample* of X with cardinality N. Now, the *empirical mean* is defined as

$$\widehat{E}_N[X] \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{2-9}$$

The following two theorems support the evaluation of the empirical mean against the actual mean.

Theorem 2.2. [34, Theorem 7.2] For any $\epsilon > 0$, we have

$$\lim_{N\to\infty} P\left[\left|\mathbb{E}[X] - \widehat{E}_N[X]\right| > \epsilon\right] = 0.$$

Moreover, the empirical mean converges with probability one to the expected value, that is

$$\lim_{N\to\infty} \widehat{E}_N[X] = \mathbb{E}[X].$$

Theorem 2.3. [34, Theorem 7.3] If the variance $\mathrm{Var}(X)$ is finite, then for any $N \geq 1$, we have

$$\mathrm{Var}\left(\widehat{E}_N[X]\right) \stackrel{\text{def}}{=} \mathbb{E}\left[\left(\mathbb{E}[X] - \widehat{E}_N[X]\right)^2\right] = \frac{\mathrm{Var}(X)}{N}.$$

A consequence of Theorem 2.3, if the variance of the original random variable X is known a priori, we can compute the number of samples N needed to guarantee a maximum variance of the empirical mean $\mathrm{Var}(\widehat{E}_N[X])$:

$$N \geq \frac{\mathrm{Var}(X)}{\mathrm{Var}\left(\widehat{E}_N[X]\right)}$$

In that case, it is also possible to employ Chebychev's inequality to calculate the probability that the empirical mean is off from the actual mean by a value smaller than ϵ:

Theorem 2.4. (Chebychev inequality) Let x be a random variable with $\mathrm{Var}(x) < \infty$. Then, for any $\epsilon > 0$, we have

$$P\left[|x - \mathbb{E}[x]| \geq \varepsilon\right] \leq \frac{\mathrm{Var}(x)}{\epsilon^2}.$$

The formulation of the inequality is taken from [34]. Assuming no bias on the empirical mean defined in Eq. (2-9) such that $\mathbb{E}\left[\widehat{E}_N[X]\right] = \mathbb{E}[X]$, we obtain:

$$\begin{aligned} P\left[|\widehat{E}_N[X] - \mathbb{E}[X]| < \epsilon\right] &= 1 - P\left[|\widehat{E}_N[X] - \mathbb{E}[X]| \geq \epsilon\right] \\ &\geq 1 - \frac{\mathrm{Var}(\widehat{E}_N[X])}{\epsilon^2} \\ &\geq 1 - \frac{\mathrm{Var}([X])}{N\epsilon^2} \end{aligned}$$

The reader is referred to [36, Sec. 13.3] for a validation of this result and additional information regarding the topic.

Tempo et al. conclude that, unfortunately, the variance of the original function is often unknown and that Theorem 2.3 can only be used to conclude that the error is of the order $\mathcal{O}(N^{-1/2})$. In other words, the precision of the estimation will increase twofold when increasing the number of draws N by a factor of 4. Still, an essential characteristic of the MC method is that the mean square error (MSE) of the empirical mean is independent of the problem dimension, justifying the famous statement that the MC method *breaks the curse of dimensionality*. Section 2-4-2 discusses a method proposed in [34] to analyse robustness and quantify the performance of uncertain systems. The theory offers an insightful view on the application of MC methods.

2-4-2 Monte Carlo method for performance functions

In their work, Tempo et al. introduce a way of quantifying robustness for uncertain systems [34, Chap. 6]. They define for a system with uncertainties $\Delta \in \mathcal{B}_{\mathbb{D}}$ the performance function $J(\Delta)$:

$$J(\Delta)\colon \mathbb{D} \to \mathbb{R}$$

where $\mathbb{D}$ is the uncertainty structured set for which a definition is posed in [34, Sec. 3.6] and an associated performance level γ. This performance function $J(\Delta)$ may, for instance, be

defined to be equal to 0 if a system is stable for a particular uncertainty Δ and 1 otherwise. Then, they propose that checking for robust stability is equivalent to checking if

$$J(\Delta) \leq \gamma$$

for all $\Delta \in \mathcal{B}_{\mathbb{D}}$ [34, Problem 6.1]. In that case, the sets $\mathcal{B}_G \stackrel{\text{def}}{=} \{\Delta \in \mathcal{B}_{\mathbb{D}} \colon J(\Delta) \leq \gamma\}$ and $\mathcal{B}_{\mathbb{D}}$ coincide.

In contrast to robust stability, probabilistic robustness of a control system is the probability of a desired performance level being satisfied. Therefore, they aim to compute the following quantity [34, Sec. 6.3]:

$$p(\gamma) \stackrel{\text{def}}{=} P\left[J(\Delta) \leq \gamma\right] = \int_{\mathcal{B}_G} f_\Delta(\Delta)\,\mathrm{d}\Delta$$

Where the uncertainty matrix Δ has a PDF $f_\Delta(\Delta)$ and support $\mathcal{B}_{\mathbb{D}}$. In such a probabilistic setting, the measure of robustness is related to the ratio of the volumes $\mathrm{Vol}(\mathcal{B}_G)/\mathrm{Vol}(\mathcal{B}_{\mathbb{D}})$, which is, preferably, close to one. Tempo et al. summarise that the probability of performance $p(\gamma)$ measures the probability that a level of performance γ is achieved when Δ has distribution $f_\Delta(\Delta)$.

Since this probability is in general difficult to compute analytically, one may aim to compute the estimate $\widehat{p}_N(\gamma)$, instead [34, Sec. 7.1]:

$$\widehat{p}_N(\gamma) = \frac{N_G}{N}$$

where N_G is the number of samples Δ_i of multisample $\Delta_{1\ldots N}$ such that $J(\Delta_i) \leq \gamma$. To this end, one may define an indicator function $\mathbb{I}_{\mathcal{B}_G}(\Delta)$ to count these samples:

$$\mathbb{I}_{\mathcal{B}_G}(\Delta) = \begin{cases} 1 & \text{if } \Delta \in \mathcal{B}_G \\ 0 & \text{otherwise} \end{cases},$$

such that the estimate $\widehat{p}_N(\gamma)$ is equal to the empirical mean of this indicator function:

$$\widehat{p}_N(\gamma) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{\mathcal{B}_G}\left(\Delta_i\right).$$

Now, it is favourable to ensure with high probability that the estimation error is smaller than some value ϵ [34, Sec. 8.3]:

$$|\widehat{p}_N(\gamma) - p(\gamma)| < \epsilon.$$

That is, given accuracy $\epsilon \in (0,\, 1)$ and confidence $\delta \in (0,\, 1)$, one should know the minimum number of samples N required such that the following is satisfied:

$$P\left[|\widehat{p}_N(\gamma) - p(\gamma)| < \epsilon\right] > 1 - \delta. \tag{2-10}$$

To that end, consider Chebychev's inequality from Theorem 2.4 and the Hoeffding inequalities from the following theorems, taken from [34].

Theorem 2.5. (Hoeffding inequality) Let $x_1, \ldots, x_N$ be independent bounded random variables with $x_i \in [a_i, b_i]$ and define $s_N \stackrel{\text{def}}{=} \sum_{i=1}^{N} x_i$. Then, for any $\epsilon > 0$, we have

$$P\left[s_N - \mathbb{E}\left[s_N\right] \geq \epsilon\right] \leq e^{-2\epsilon^2 / \sum_{i=1}^{N}(b_i - a_i)^2}$$

and

$$P\left[s_N - \mathbb{E}\left[s_N\right] \leq -\epsilon\right] \leq e^{-2\epsilon^2 / \sum_{i=1}^{N}(b_i - a_i)^2}.$$

If the variables x_i are independent and bounded in the interval $[a, b]$, we acknowledge the following inequality:

Theorem 2.6. Two-sided Hoeffding inequality Let $x_1, \ldots, x_N$ be independent random variables such that $x_i \in [a, b]$. Then, for any $\epsilon > 0$, we have

$$P\left[\left|s_N - \mathbb{E}\left[s_N\right]\right| \geq \epsilon\right] \leq 2e^{-2\epsilon^2/(N(b-a)^2)}.$$

These three inequalities form the basis for the Bernoulli bound and the Chernoff bound that pose a minimum number of samples required to obtain a certain accuracy ϵ and confidence $1-\delta$ in Eq. (2-10). They require the definitions of $\mathbb{I}_{\mathcal{B}_G}(\Delta_i) = p_i$ and $s_N = \sum_i^N p_i$. The Bernoulli bound, found in 1713, is an elegant lower bound on the number of samples required [37]:

Theorem 2.7. (Bernoulli bound) For any $\epsilon \in (0, 1)$ and $\delta \in (0, 1)$, if

$$N \geq \frac{1}{4\epsilon^2 \delta},$$

then, with probability greater than $1 - \delta$, we have $|\widehat{P}_N(\gamma) - p(\gamma)| < \epsilon$.

However, in 1952, Chernoff found a significant improvement to this bound that is based on the Hoeffding Inequality [38]:

Theorem 2.8. (Chernoff bound) For any $\epsilon \in (0, 1)$ and $\delta \in (0, 1)$, if

$$N \geq \frac{1}{2\epsilon^2} \log \frac{2}{\delta},$$

then, with probability greater than $1 - \delta$, we have $|\widehat{P}_N(\gamma) - p(\gamma)| < \epsilon$.

The minimum number of samples required, as described by the Bernoulli bound or the Chernoff bound, can be computed a priori. Tempo et al. present a table with typical values for ϵ and δ and highlight the differences between the Bernoulli and Chernoff bounds [34, Table 8.1]. A replica of that table is shown as Table 2-1.

As opposed to the bounds discussed above, it is possible to compute the classical confidence intervals a posteriori. In order to calculate the lower and upper confidence intervals p_L and p_U such that

$$P\left[p_L \leq p(\gamma) \leq p_U\right] > 1 - \delta,$$

Table 2-1: Comparison of the minimum number of samples required to obtain a certain precision ϵ and confidence $1-\delta$, obtained from [34, Table 8.1]

ϵ	$1-\delta$	Bernoulli	Chernoff
0.05	0.95	2000	738
	0.99	1.00×10^4	1060
	0.995	2.00×10^4	1199
	0.999	1.00×10^5	1521
0.01	0.95	5.00×10^4	1.84×10^4
	0.99	2.50×10^5	2.65×10^4
	0.995	5.00×10^5	3.00×10^4
	0.999	2.50×10^6	3.80×10^4
0.005	0.95	2.00×10^5	7.38×10^4
	0.99	1.00×10^6	1.06×10^5
	0.995	2.00×10^6	1.20×10^5
	0.999	1.00×10^7	1.52×10^5
0.001	0.95	5.00×10^6	1.84×10^6
	0.99	2.50×10^7	2.65×10^6
	0.995	5.00×10^7	3.00×10^6
	0.999	2.50×10^8	3.80×10^6

one evaluates a posteriori the equations

$$\sum_{k=N_G}^{N} \binom{N}{k} p_L^k (1-p_L)^{N-k} = \delta_L;$$
$$\sum_{k=0}^{N_G} \binom{N}{k} p_U^k (1-p_U)^{N-k} = \delta_U;$$

with $\delta_L + \delta_U = \delta$ and the binomial coefficient $\binom{N}{k}$. An explicit solution to these equations is not available, so one should resort to standard tables or numerical methods. The reader is advised to consult [34, Sec. 8.3] and [39] for more information.

Chapter 3

Switching Max-Plus Linear Systems

This chapter discusses the modelling of switching max-plus linear (SMPL) systems used throughout this thesis. The concept was introduced in 2006 to account for systems with multiple modes of operation [40]. Each mode describes the system with a max-plus linear (MPL) state equation and an MPL output equation. The switching may be deterministic, as a function of only the inputs and the states, or stochastic. The interpretation of SMPL systems in the event-driven domain is analogous to piecewise-affine (PWA) systems in the time-driven domain. Similarly, the properties of the individual MPL systems provide insight into the behaviour of the underlying SMPL system.

This chapter details two equivalent stochastic descriptions of SMPL systems. First, a deterministic model of an SMPL system is introduced in Section 3-1 on page 28. Section 3-2 on page 31 introduces stochastic switching, and Section 3-3 on page 34 discusses constraints on the mode and state sequences of SMPL systems.

3-1 Deterministic Modelling

A switching max-plus linear (SMPL) system description differs from an MPL representation by its variable system matrices [40]. In each mode $\ell \in \mathcal{L} = \underline{n_L}$, where n_L is the number of modes in the set $\mathcal{L}$, the system is described by an MPL state and output equation. Additionally, at each step k, the mode $\ell(k)$ is the output of a switching function $\phi(\cdot)$, leading to the following general description:

$$\begin{aligned} x(k) &= A^{(\ell(k))} \otimes x(k-1) \oplus B^{(\ell(k))} \otimes u(k) \\ y(k) &= C^{(\ell(k))} \otimes x(k) \\ \ell(k) &= \phi\left(\ell(k-1),\, x(k-1),\, v(k),\, u(k)\right) \\ \ell(k) &\in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N} \end{aligned} \tag{3-1}$$

Here, $A^{(\ell(k))} \in \mathbb{R}_\varepsilon^{n_x \times n_x}$, $B^{(\ell(k))} \in \mathbb{R}_\varepsilon^{n_x \times n_u}$ and $C^{(\ell(k))} \in \mathbb{R}_\varepsilon^{n_y \times n_x}$ are the system matrices for the ℓ-th mode. Furthermore, the discrete control signal $v(k)$ with $|v(k)| = n_v$ is used to influence the switching behaviour of the system. By changing the mode $\ell(k)$, the system matrices and thus its dynamics are altered. In this way, structural system changes such as, for example, the order of events can be modelled. The function $\phi(\cdot)$ that dictates the switching may be deterministically or stochastically dependent on its input parameters. This section discusses the deterministic case, while Section 3-2 on page 31 details the stochastic one. Additionally, Section 3-3 on page 34 discusses the case where the set of mode sequences is constrained.

In their early work [40], Van den Boom and De Schutter ascribe deterministic switching to a switching variable $z(k)$:

$$z(k) = \Phi(\ell(k-1),\, x(k-1),\, u(k),\, v(k)) \quad \in \mathbb{R}_\varepsilon^{n_z},\ \forall k \in \mathbb{N}.$$

If $z(k)$ is in the partition $\mathscr{L}^{(i)}$ of $\mathbb{R}_\varepsilon^{n_z}$, the system's mode is $\ell(k) = i$. In the above formulation and as introduced before, $v(k)$ is an additional discrete control input that influences switching behaviour. It may, for example, exclusively determine the next mode $\ell(k)$, or it may influence the switching stochastics, as explained further in Section 3-2 on page 31.

An autonomous SMPL system description is analogous to an autonomous MPL description, and has the following form:

$$x(k) = A^{(\ell(k))} \otimes x(k-1) \tag{3-2a}$$
$$\ell(k) = \phi(\ell(k-1),\, x(k-1)) \tag{3-2b}$$
$$y(k) = C^{(\ell(k))} \otimes x(k) \tag{3-2c}$$

Again, mode switching is determined through the switching function $\phi(\cdot)$ in Eq. (3-2b).

We classify SMPL systems based on various characteristics related to control. As a first observation, an autonomous system is one under no control. A *nonautonomous* system is controlled through *discrete*, *continuous* or *hybrid* control. Eq. (3-1) and Eq. (3-2) are examples of a description of *open-loop* SMPL systems. That is, control inputs $u(k)$ and $v(k)$ are unspecified in the system description. Conversely, a *controlled* SMPL system is one whose description also includes a controller. These various forms of SMPL systems are defined in this report as follows.

First, we distinguish between autonomy levels:

Autonomous system: A system uninfluenced by external inputs.

Nonautonomous system: A system influenced through discrete, continuous or hybrid control.

Then, we denote whether a system description is in open-loop:

Open-loop system: A system whose description does not specify control inputs $u(k)$ and $v(k)$.

Controlled system: A system whose description includes a controller.

Lastly, the notions of discrete, continuous and hybrid control are specified as:

Discrete control: Control using the discrete signal $v(k)$ and no continuous signal $u(k)$.

Continuous control: Control using the continuous signal $u(k)$ and no discrete signal $v(k)$.

Hybrid control: Control using both the discrete signal $v(k)$ and the continuous control signal $u(k)$.

For simplicity, we combine the vectors on which the switching depends into the vector $w(k)$:

$$w(k) = \begin{bmatrix} \ell(k-1) & x^\top(k-1) & u^\top(k) & v^\top(k) \end{bmatrix}^\top \in \mathbb{R}^{n_w}$$

where $n_w = 1 + n_x + n_u + n_v$.

Since the switching is assumed to be deterministic, the following holds:

$$P\left[L(k) = \ell(k) \mid w(k)\right] \in \{0,\, 1\}$$

and

$$\sum_{\ell(k)=1}^{n_L} P\left[L(k) = \ell(k) \mid w(k)\right] = 1$$

Here, $L(k)$ is a stochastic variable denoting the system's mode at event step k and $\ell(k)$ is its value.

Consider Example 3.1, which shows a case of deterministic switching for an SMPL system with three modes.

Example 3.1. Consider an unspecified SMPL system with three modes that is in mode 1 at event step k. Let $v(k)$ be a control variable determining the switching sequence. This influence is achieved by defining the probability functions

$$P\left[L(k) = 1 \mid 1,\, x(k-1),\, u(k),\, v(k)\right] = \begin{cases} 1 \text{ for } v(k) = 1,\ \forall k \\ 0 \text{ for } v(k) \neq 1,\ \forall k \end{cases}$$

$$P\left[L(k) = 2 \mid 1,\, x(k-1),\, u(k),\, v(k)\right] = \begin{cases} 1 \text{ for } v(k) = 2,\ \forall k \\ 0 \text{ for } v(k) \neq 2,\ \forall k \end{cases}$$

$$P\left[L(k) = 3 \mid 1,\, x(k-1),\, u(k),\, v(k)\right] = \begin{cases} 1 \text{ for } v(k) = 3,\ \forall k \\ 0 \text{ for } v(k) \neq 3,\ \forall k \end{cases}$$

Evidently, for $\ell(k) = 1$, the control variable $v(k)$ solely and deterministically controls the next mode $\ell(k+1)$. The previous state $x(k-1)$ and continuous control signal $u(k)$ do not influence the switching behaviour. Often, as is the case for this example, the probability functions for deterministic switching are piecewise constant with values of either 0 or 1 [4].

Finally, we define three structural properties of SMPL systems. For a given integer N, let the set $\mathcal{L}_N = \left\{ [\ell_1 \ \dots \ \ell_N]^\top \mid \ell_m \in \{1, \dots, n_L\},\ m = 1, \dots, N \right\}$. Here, $A^\top$ denotes the transpose of matrix A.

Definition 3.1. [12] Let $\alpha \in \mathbb{R}$ be given. Define the matrices $A_\alpha^{(\ell)}$ with $\left[A_\alpha^{(\ell)}\right]_{i,j} = \left[A^{(\ell)}\right] - \alpha$. An SMPL system is *structurally controllable* if there exists a finite positive integer N such that for all $\tilde{\ell} = [\ell_1 \ \dots \ \ell_N]^\top \in \mathcal{L}_N$ the matrices

$$\Gamma_\alpha^N(\tilde{\ell}) = \begin{bmatrix} A_\alpha^{(\ell_N)} \otimes \cdots \otimes A_\alpha^{(\ell_2)} \otimes B^{(\ell_1)} & A_\alpha^{(\ell_N)} \otimes A_\alpha^{(\ell_{N-1})} \otimes B^{(\ell_{N-2})} & A_\alpha^{(\ell_N)} \otimes B^{(\ell_{N-1})} & B^{(\ell_N)} \end{bmatrix}$$

are row-finite, i.e., in each row, there is at least one entry different from ε.

Definition 3.2. [6, Def. 4.5] Let $\alpha \in \mathbb{R}$ be given. Define the matrices $A_\alpha^{(\ell)}$ with $\left[A_\alpha^{(\ell)}\right]_{i,j} = \left[A^{(\ell)}\right] - \alpha$. An SMPL system is *weakly structurally controllable* if for a finite positive integer $k \leq n$ such that there exists a mode sequence $\tilde{\ell} = [\ell_1 \ \dots \ \ell_k]$ such that the matrices

$$\Gamma_\alpha^k(\tilde{\ell}) = \begin{bmatrix} A_\alpha^{(\ell_N)} \otimes \cdots \otimes A_\alpha^{(\ell_2)} \otimes B^{(\ell_1)} & A_\alpha^{(\ell_N)} \otimes A_\alpha^{(\ell_{N-1})} \otimes B^{(\ell_{N-2})} & A_\alpha^{(\ell_N)} \otimes B^{(\ell_{N-1})} & B^{(\ell_N)} \end{bmatrix}$$

are row-finite, i.e., in each row, there is at least one entry different from ε.

Definition 3.3. [12] Let $\alpha \in \mathbb{R}$ be given. Define the matrices $A_\alpha^{(\ell)}$ with $\left[A_\alpha^{(\ell)}\right]_{i,j} = \left[A^{(\ell)}\right]_{i,j} - \alpha$. An SMPL system is *structurally observable* if there exists a finite positive integer M such that for all $\tilde{\ell} = [\ell_1 \ \dots \ \ell_M]^\top \in \mathcal{L}_M$ the matrices

$$O_\alpha^M(\tilde{\ell}) = \begin{bmatrix} C_\alpha^{(\ell_M)} \otimes A_\alpha^{(\ell_M)} \otimes \cdots \otimes A_\alpha^{(\ell_2)} \\ \vdots \\ C_\alpha^{(\ell_M)} \otimes A_\alpha^{(\ell_M)} \otimes A_\alpha^{(\ell_{N-1})} \\ C_\alpha^{(\ell_M)} \otimes A_\alpha^{(\ell_M)} \\ C_\alpha^{(\ell_M)} \end{bmatrix}$$

are column-finite, i.e., in each column, there is at least one entry different from ε.

The wording of Definition 3.1 and Definition 3.3 has been taken from work by Van den Boom and De Schutter [4].

3-2 Stochastic Modelling

The switching in SMPL systems may also depend on nondeterministic influences. The travel time between two stations in a train network may unpredictably increase due to a faulty train. In some cases, this uncertainty may be modelled as parametric uncertainty, as in Section 2-1 on page 6. Other times, the uncertainty can only be captured in the switching behaviour. This happens when, for example, a connection between two stations is lost due to a broken track and the structure of the corresponding system matrices changes. This section deals with unpredictable mode switching and reviews two equivalent uncertain SMPL descriptions: type-1 and type-2. First, Section 3-2-1 introduces lesser-used randomly switching max-plus linear (RSMPL) systems.

3-2-1 Randomly switching max-plus-linear systems

Introduced by Van den Boom and De Schutter in 2007 [41], RSMPL systems switch modes based on a stochastic sequence. For a system described by the following state-update equations, the probability of switching from a mode i to a mode j is assumed to be given by $P_{\mathrm{s}}(i,\, j)$, $\forall i = 1,\, \ldots,\, n_L$, $j = 1,\, \ldots,\, n_L$:

$$\begin{aligned} x(k) &= A^{(\ell(k))} \otimes x(k-1) \oplus B^{(\ell(k))} \otimes u(k) \\ y(k) &= C^{(\ell(k))} \otimes x(k) \\ \ell(k) &\in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N}, \end{aligned}$$

where

$$P\left[L(k) = j \mid \ell(k-1) = i\right] = P_{\mathrm{s}}(i,\, j).$$

In other words, the probability of switching to a certain mode is entirely determined by the system's previous mode. No internal or external signal may further alter the probability. Such a system description is an exciting subject for throughput analyses under continuous control. It, however, offers no possibility for discrete or hybrid control.

Often, both deterministic and stochastic variables determine the switching of an SMPL system [42, 4]. In that case, the switching depends on the previous mode $\ell(k-1)$, the previous state $x(k-1)$, the input variable $u(k)$ and an additional control signal $v(k)$. The system is then categorised as a type-1 SMPL system, introduced in the following section.

3-2-2 Type-1 and type-2 switching max-plus-linear systems

Van den Boom and De Schutter distinguish between the equivalent type-1 and type-2 stochastic SMPL systems, where the first are defined as follows.

Definition 3.4. (Type-1 SMPL system) [4] Consider a system described by Eq. (3-1) on page 28 and n_L possible modes and let the probability of switching to mode $\ell(k)$ given $\ell(k-1)$, $x(k-1)$, $u(k)$, $v(k)$ be denoted by $P[L(k) = \ell(k) \mid \ell(k-1),\, x(k-1),\, u(k),\, v(k)]$. Then, the system is a type-1 SMPL system if for any given $\ell(k) \in \{1,\, \ldots,\, n_L\}$, $P[L(k) = \ell(k) \mid \cdot,\, \cdot,\, \cdot,\, \cdot]$ is a probability function that is piecewise affine on polyhedral partition of the space of the variables $\ell(k-1)$, $x(k-1)$, $u(k)$, $v(k)$.

Here, a polyhedral partition is defined as:

Definition 3.5. [4] A polyhedral partition $\{\Gamma_i\}_{i=1,\dots,n_s}$ of the space $\mathbb{R}^{n_w}$ is defined as the partitioning of the space $\mathbb{R}^{n_w}$ into non-overlapping polyhedra Γ_i, $i = 1, \dots, n_s$ of the form

$$\Gamma_i = \{w(k) \mid S_i w(k) \preceq_i s_i\}, \quad \text{for } i = 1, \dots, n_s,$$

for some matrices $S_i \in \mathbb{R}^{q \times n_w}$ and vectors $s_i \in \mathbb{R}^q$ and with $\preceq_i$ a vector operator where the entries stand for either $\leq$ or $<$ and there holds

$$\bigcup_{i=1}^{n_s} \Gamma_i = \mathbb{R}^{n_w} \quad \text{and} \quad \Gamma_i \cap \Gamma_j = \emptyset \quad \text{for } i \neq j.$$

Through these definitions, we derive that for any type-1 SMPL system, there exist a polyhedral partition $\{\Gamma_i\}_{i=1,\dots,n_s}$ of $\mathbb{R}^{n_w}$, vectors $\alpha_{m,i}$ and scalars $\beta_{m,i}$ for $i = 1, \dots, n_s$ and $m \in \{1, \dots, n_L\}$ such that the probability P can be written as

$$P[L(k) = \ell(k) \mid w(k)] = \alpha_{\ell(k),i}^\top w(k) + \beta_{\ell(k),i}, \quad \text{if } w(k) \in \Gamma_i$$

for $w(k) = \begin{bmatrix} \ell(k-1) & x^\top(k-1) & u^\top(k) & v^\top(k) \end{bmatrix}^\top \in \mathbb{R}^{n_w}$.

In general, since P is a probability, the following conditions hold for n_L possible modes:

$$0 \leq P[L(k) = \ell(k) \mid w(k)] \leq 1$$

and

$$\sum_{\ell(k)=1}^{n_L} P[L(k) = \ell(k) \mid w(k)] = 1.$$

Because of its form, a type-1 SMPL model offers intuitive insight into the system's dynamics. [43] proposes a type-2 SMPL system that is more easily translated into other hybrid system descriptions, such as a PWA system or a max-min-plus-scaling (MMPS) system [44]. The following definition is taken from later work by Van den Boom and De Schutter.

Definition 3.6. (Type-2 SMPL system) [4] A type-2 SMPL system is defined as follows: Consider the system described by Eq. (3-1) on page 28 with n_L possible modes. The mode $\ell(k) = m$ if

$$z(k) = \begin{bmatrix} \ell(k-1) & x^\top(k-1) & u^\top(k) & v^\top(k) & d(k) \end{bmatrix}^\top \in \Omega_m \subset \mathbb{R}^{n_z}$$

for $m = 1, \dots, n_L$, where $d(k) \in [0, 1]$ is a uniformly distributed stochastic scalar signal, and where Ω_m is a union of polyhedra, so $\Omega_m = \cup_{j=1}^{n_m} \Omega_{m,j}$ in which $\{\Omega_{m,j}\}_{m=1,\dots,n_L;\, j=1,\dots,n_m}$ is a polyhedral partition.

The polyhedral partition $\{\Omega_{m,j}\}_{m=1,\dots,n_L,\, j=1,\dots,n_m}$ can be parameterised by

$$\Omega_{m,j} = \{z(k) \mid R_{m,j} z(k) \preceq_{m,j} r_{m,j}\}, \quad \text{for } j = 1, \dots, n_m.$$

Van den Boom and De Schutter prove that the classes of type-1 SMPL systems and type-2 SMPL systems are equivalent in the sense of input-state-output-mode behaviour [4]. The reader is referred to their work for the proof. They also propose and prove the following:

Proposition 3.1. [4] A type-2 SMPL system can always be rewritten in the form:

$$x(k) = \bar{A}^{(\kappa(k))} \otimes x(k-1) \oplus \bar{B}^{(\kappa(k))} \otimes u(k)$$
$$y(k) = \bar{C}^{(\kappa(k))} \otimes x(k)$$

such that the mode $\kappa(k) = m$ if

$$z(k) = \begin{bmatrix} \kappa(k-1) & x^\top(k-1) & u^\top(k) & v^\top(k) & d(k) \end{bmatrix}^\top \in \bar{\Omega}_m$$

where $d(k) \in [0,\, 1]$ is a uniformly distributed stochastic scalar signal, and where $\{\bar{\Omega}_m\}_{m=1,\dots,n_\kappa}$ is a polyhedral partition and $\bar{\Omega}_m$ can be written as

$$\bar{\Omega}_m = \{z(k) \mid R_m z(k) \preceq^r_m r_m\}.$$

Here,

$$R_{m,i} = \begin{bmatrix} \bar{\alpha}^\top_{m-1,i} & -1 \\ -\bar{\alpha}^\top_{m,i} & 1 \\ S_i & 0 \end{bmatrix}, \quad r_{m,i} = \begin{bmatrix} \bar{\beta}_{m-1,i} \\ -\bar{\beta}_{m,i} \\ -s_i \end{bmatrix}, \quad \preceq^r_{m,i} = \begin{bmatrix} < \\ \leq \\ \preceq^s_i \end{bmatrix}$$

They also introduce the notion of structural finiteness for SMPL systems in the following way:

Definition 3.7. [4] an SMPL system is structurally finite if for any finite $(x(k-1),\, u(k))$ we have that $x(k)$ and $y(k)$ are finite for all $\ell(k-1) \in \{1,\, \dots,\, n_L\}$ and any $d(k) \in [0,\, 1]$.

This notion is captured in their lemma:

Lemma 3.1. [4] an SMPL system is structurally finite if and only if the matrix

$$H^{(\ell)} = \begin{bmatrix} A^{(\ell)} & B^{(\ell)} & \varepsilon \\ \varepsilon & \varepsilon & C^{(\ell)} \end{bmatrix}$$

is row-finite for all $\ell = 1,\, \dots,\, n_L$.

The authors note that physical systems are typically structurally finite. This characteristic is necessary to convert between type-2 SMPL systems and type-d PWA systems of the form

$$\begin{aligned} x(k) &= A_i x(k-1) + B_i u(k) + f_i \\ y(k) &= C_i x(k) + D_i u(k) + g_i \end{aligned} \quad \text{for} \quad \begin{bmatrix} x(k-1) \\ u(k) \\ d(k) \end{bmatrix} \in \Omega_i,$$

where $f_i \in \mathbb{R}^n$, $g_i \in \mathbb{R}^l$, $A_i \in \mathbb{R}^{n \times n}$, $B_i \in \mathbb{R}^{n \times m}$, $C_i \in \mathbb{R}^{l \times n}$ and $D_i \in \mathbb{R}^{l \times m}$ for $i = 1,\, \dots,\, N$. The signal $d(k) \in [0,\, 1]$ in the description above is a uniformly distributed stochastic scalar signal and $\{\Omega_i\}_{i=1,\dots,N}$ is a polyhedral partition of $\mathbb{R}^{n+m+1}$. This form, introduced by [4], is an extension to the PWA system description described by Sontag [45].

In summary, whereas type-1 systems offer distinct physical insight into their behaviour, type-2 systems are more easily translated into other frameworks. For this reason, we opt to use a type-1 SMPL description for the remainder of this report. The work by Van den Boom and De Schutter contains additional information on the equivalence between the frameworks [4].

3-3 Automaton-Based Mode Constraints

In many practical situations, the mode sequence of an SMPL system may be subject to constraints. For instance, a production system may not be able to indefinitely produce the same part since there is a limited supply of subparts. Also, trains sometimes need to divert to another route because of regular maintenance on a part of the network.

The probability of switching to a certain mode $\ell(k)$ may depend in a state-invariant Markovian fashion on the previous mode $\ell(k-1)$ [46, 47], as discussed in Section 3-2 on page 31. Additionally, the set of admissible switching sequences may be governed by an automaton; a concept explained in Section 2-3 on page 16 [48]. The latter is the main point of discussion in this section since it allows for straightforward restriction of the length of mode sequences.

Consider again the nonautonomous SMPL description of Eq. (3-1) on page 28 with n_L modes in event cycle k. Denote a finite length switching sequence as $\sigma_k = \{\ell(1), \ell(2), \ldots, \ell(k)\}$, where $|\sigma_k| = k \geq 1$. This section discusses the constraining of the set of admissible mode sequences by an automaton. A switching sequence is admissible if its corresponding word is in the regular language.

Next to these constraints, the switching behaviour may be deterministic or stochastic. A deterministic finite automaton (DFA), introduced in Section 2-3-3 on page 18, is suitable for modelling systems that switch in a deterministic but constrained way. Stochastically switching systems may be modelled using discrete hybrid stochastic automata (DHSAs), introduced below.

A DHSA, first introduced in [33], is composed of four elements:

- Switched affine system (SAS)
- Event generator (EG)
- Stochastic finite state machine (sFSM)
- Mode selector (MS)

The interconnection of these four elements is similar to those of a discrete hybrid automaton (DHA) introduced in Section 2-3-3 on page 18 and is visualised in Figure 3-1. The main difference between a DHSA and a DHA is the replacement of the finite state machine (FSM) with an sFSM. Now, the state update equation of the sFSM is described as:

$$P[x_b(k+1) = \hat{x}_b] = f_{\text{sFSM}}(x_b(k),\, u_b(k),\, \delta_e(k),\, \hat{x}_b)$$

where only the probability distribution of $x_b(k+1) = \hat{x}_b$ is known. Generally, the state vector $x_b(k)$ is a zero vector with entry i being a 1 if the i-th discrete sFSM state is active.

Now, we can introduce mode switching constraints that are a function of the discrete sFSM state $x_b(k)$. If switching is stochastic, we enforce these constraints on the discrete control signal $v(k)$ that influences mode switching rather than on the mode $\ell(k)$ itself. Due to the switching uncertainty, this choice is not equivalent to enforcing a hard constraint on the mode sequence, but rather, it acts as a soft constraint.

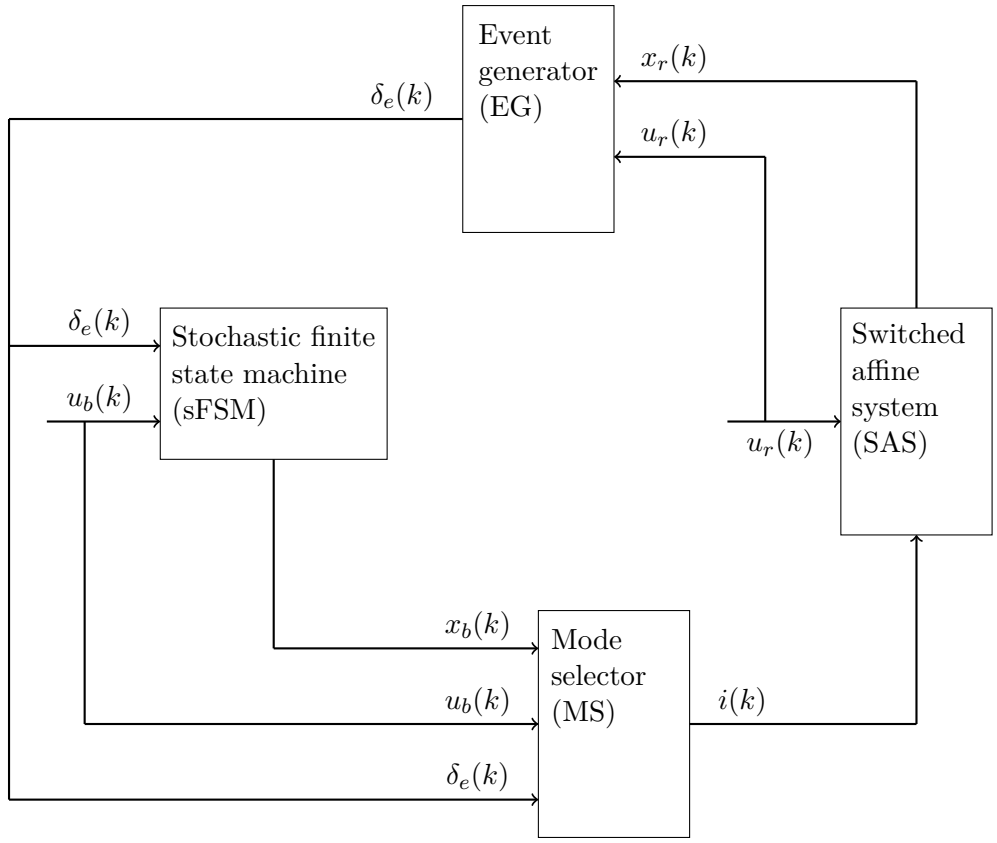

Figure 3-1: A discrete hybrid stochastic automaton (DHSA) that consists of an sFSM and a SAS, connected through an EG and an MS. The output signals are omitted. The schematic is adopted from [32, Fig. 1].

To this end, we introduce the constraint vector C_{x_b} that has the set of admissible discrete control inputs as a function of $x_b(k)$ as its entries:

$$C_{x_b} = \begin{bmatrix} \tilde{v}_{x_b,1} \\ \tilde{v}_{x_b,2} \\ \vdots \\ \tilde{v}_{x_b,|x_b|} \end{bmatrix}$$

where $\tilde{v}_{x_b,i}$ denotes the admissible set of discrete control inputs if the i-th discrete sFSM state is active as denoted by $x_b(k)$. In the case of deterministic switching, we often define $P\left[L(k) = \ell(k) \mid v(k) = \ell(k)\right] = 1$, such that the entries of C_{x_b} may equivalently relate to admissible succeeding modes.

Example 3.2. For the example of Figure 3-2 on page 36, we define the discrete state vector as such:

$$x_b(k) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{if state 1 is active at event step } k \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if state 2 is active at event step } k \end{cases}$$

The admissible control inputs as visualised by the arrows in the figure are captured by C_{x_b}:

$$C_{x_b} = \begin{bmatrix} \tilde{v}_{x_b,1} \\ \tilde{v}_{x_b,2} \end{bmatrix} = \begin{bmatrix} \{2\} \\ \{1,\, 2\} \end{bmatrix}$$

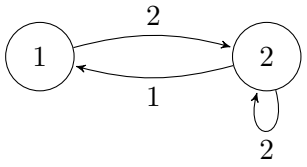

Figure 3-2: Automaton of a constrained bimodal system

The framework can be used to capture and limit the number of equal modes in a row. For this, denote with the discrete sFSM state Q_{ij} the j-th occurrence of mode $\ell = i$ in a row. Figure 3-3 depicts this for a bimodal system for sequences up to length n of the same mode. Its paths with corresponding labels relate to the set of admissible succeeding modes. These states form in a specific order the vector $\tilde{Q}$, such that $\tilde{Q}^\top x_b(k) = Q_{ij}$ if Q_{ij} is the active state at event step k. Consider now a system in which the maximum length of a sequence of equal modes is constrained to equal $m < n$. Then, the set of admissible discrete control sequences at $Q_{im} \dots Q_{in}$ is $\underline{n_L} \backslash \{i\}$. Here, the backslash operation $\backslash$ denotes the relative complement, i.e., $A \backslash B$ is the set of objects that belong to A and not B.

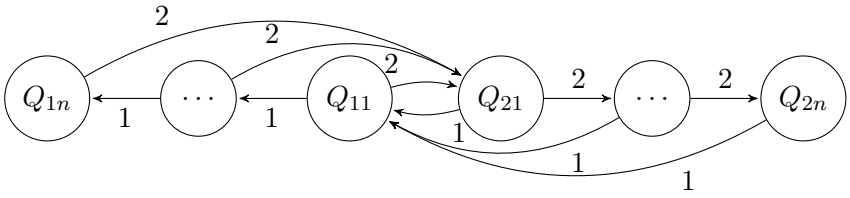

Figure 3-3: Example of capturing number of equal modes in a row for a bimodal system that allows for sequences up to length n of the same mode.

Example 3.3. As an example, consider the constraint matrix C_{x_b} and corresponding vector $\tilde{Q}$ of a trimodal system where no mode may occur twice in a row:

$$C_{x_b} = \begin{bmatrix} \{2,\, 3\} \\ \{1,\, 3\} \\ \{1,\, 2\} \end{bmatrix}, \quad \tilde{Q} = \begin{bmatrix} Q_{11} \\ Q_{21} \\ Q_{31} \end{bmatrix}$$

where $x_b(k)$ is such that $\tilde{Q}^\top x_b(k) = Q_{ij}$ if Q_{ij} is the active sFSM state, i.e., $\ell(k-j) = \ell(k-j+1) = \ldots = \ell(k-1) = i$, and $\ell(k-j-1) \neq i$. These correspond to the following automaton:

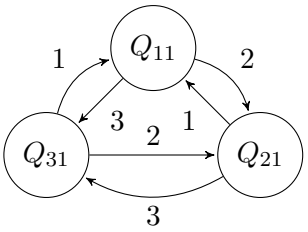

Figure 3-4: Example of an automaton enforcing the constraint that no mode may occur twice in a row for the trimodal system of Example 3.3.

Example 3.4. As a second example, consider the constraint matrix C_{x_b} and corresponding vector $\tilde{Q}$ of a bimodal system where no mode may occur three times in a row:

$$C_{x_b} = \begin{bmatrix} \{1,\,2\} \\ \{2\} \\ \{1,\,2\} \\ \{1\} \end{bmatrix}, \quad \tilde{Q} = \begin{bmatrix} Q_{11} \\ Q_{12} \\ Q_{21} \\ Q_{22} \end{bmatrix}$$

where $\tilde{Q}$ and $x_b\tilde{(k)}$ are defined as before. The following automaton may represent them:

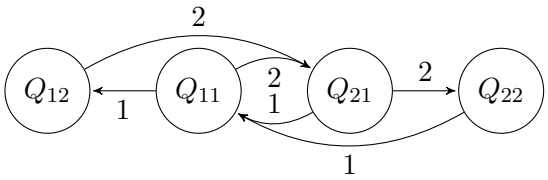

Figure 3-5: Example of an automaton enforcing the constraint that no mode may occur three times in a row for the bimodal system of Example 3.4.

To conclude, we state that for any deterministic SMPL system whose mode sequence can be captured as a discrete state of an automaton, its mode constraints can be captured by appropriate $\tilde{Q}$ and C_{x_b}. These constraints are on the control sequence $\tilde{v}(k)$ for stochastic systems. A discrete control sequence $\tilde{v}$ or mode sequence $\tilde{\ell}(k)$ of a certain length is admissible if its corresponding word is in the regular language of the sFSM.

In contrast to the conventional definition of DHSAs as introduced before, we opt to include switching uncertainty in the MS instead. This choice allows us to employ the sFSM solely for representing hard constraints while effortlessly incorporating Markovian uncertainty in the MS. The discrete state of the sFSM is then determined by the set of past modes of the SAS.

Chapter 4

Growth Rate of SMPL Systems

It is essential to have a notion of system performance in mind when optimising the system's future behaviour through control. For conventional systems, performance often pertains to their deviation from a stable point in the state-space. Such a definition is impractical for discrete-event systems (DESs), whose states often grow unboundedly. This chapter explores the notion of growth rate that offers a promising alternative on which to assess a system's stability. Section 4-1 on page 40 introduces the concept for deterministic systems, after which Section 4-2 on page 46 translates to stochastically switching systems. Lastly, Section 4-3 on page 53 assesses the accuracy of various approximations posed in the chapter.

4-1 Growth Rate of Deterministic SMPL Systems

Stability for conventional systems pertains to the boundedness of the state vector. For a switching max-plus linear (SMPL) system, its state is expected to grow unboundedly as time passes. Thus, the notion of stability for these systems is adjusted to relate to the boundedness of the state growth. More precisely, it is often associated with the boundedness of the system's buffer levels, of which the concept was introduced in Section 2-1 on page 6 [49]. Asymptotic stability implies that these levels remain constant, which can only be achieved when the average *growth rate* of all states becomes equal [5]. Chapter 5 on page 65 discusses the stability of SMPL systems in more detail. This chapter examines the growth rate of such systems.

4-1-1 Asymptotic growth rate

Define the cycle time vector $\xi \in \mathbb{R}^n$ for a DES in the max-plus algebra (MPA) with state vector $x(k)$ as [10]:

$$\xi_i \stackrel{\text{def}}{=} \lim_{k\to\infty} \frac{x_i(k)}{k}. \tag{4-1}$$

If the limit exists, $\xi_i \in \mathbb{R}$ represents the asymptotic average time between event occurrences, or, *growth rate*. The largest value $\overline{\xi}$ in ξ determines the system's performance [50, Theorem 7]. Vice versa, the system's *throughput*, often denoted by $1/\overline{\xi}$, is inversely proportional to this maximum value [12]. Often, a smaller state growth is desirable over a larger one since it increases throughput. For a train network, a smaller state growth signifies shorter times between departures.

Gupta et al. defined the notion of *synchronised system evolution* to explore the behaviour of SMPL systems further:

Definition 4.1. [6] System evolution is *synchronised* if the asymptotic average growth rates of the state trajectories attain a common value in Eq. (4-1):

$$\xi_i = \xi_j \stackrel{\text{def}}{=} \rho_{\text{s}}, \quad \forall i, j \in \underline{n}$$

Here, we define ρ_{s} as the *state growth rate*. Naturally, the state growth rate is the growth rate of all individual states in the case of synchronised system evolution.

In some cases, it is helpful to look at another definition of growth rate, the *output growth rate* ρ_{o}:

$$\rho_{\text{o}} \stackrel{\text{def}}{=} \lim_{k\to\infty} \frac{C \otimes x(k)}{k}$$

Note that $\otimes$ is not a linear operator, thus ρ_{o} is not by definition equal to $C \otimes \xi$. In many cases, however, the values of ρ_{s} and ρ_{o} differ only by a constant value related to the C-matrix, shown by the following relation. Consider, without loss of generality, a single-output C-matrix. Then, assuming synchronised state evolution, i.e., $x_i(k) = x_j(k)$, $\forall i, j \in \underline{n}$, ρ_{o} can be written as

$$\begin{aligned}\rho_{\text{o}} = \lim_{k\to\infty} \frac{\max_i \left(x_i(k) + C_i\right)}{k} &= \max_i \left(\lim_{k\to\infty} \left[\frac{x_i(k)}{k} + \frac{C_i}{k} \right] \right) \\ &= \max_i \left(\rho_{\text{s}} + \lim_{k\to\infty} \frac{C_i}{k} \right) = \rho_{\text{s}} + \lim_{k\to\infty} \frac{\max_i \left(C_i\right)}{k}\end{aligned} \tag{4-2}$$

Therefore, as long as the system's state evolution is synchronised and $\max_i (C_i) = 0$, i.e., $C \otimes \mathbb{1}_n = 0$, the two values ρ_s and ρ_o coincide. Here, $\mathbb{1}_n$ is a max-plus algebraic n-dimensional unity vector of all zeros. In the case of asynchronous system evolution, the output growth rate relates to the system's states as:

$$\rho_\mathrm{o} = \max_i \left(\lim_{k\to\infty} \left[\frac{x_i(k)}{k} + \frac{C_i}{k} \right] \right).$$

Clearly, the maximum deviation of ρ_o with respect to the growth rate of the individual states is $\|C\|_{\max} + \frac{1}{k}\|x(k)\|_{\mathbb{P}}$. In all cases, the influence of the finitely-valued entries of C will diminish as $k \to \infty$.

In summary, while these two definitions of growth rate are not equal in all cases, they allow for similar analyses of their bounds and expectations. In the remainder of this report, we will consider either state growth rate ρ_s or output growth rate ρ_o with a fixed C-matrix that has entries all equal to $e = 0$. The definitions used in the concepts proposed in this report will be clearly stated if that is of relevance.

The example below provides a helpful visualisation of the growth rate of a max-plus linear (MPL) system.

Example 4.1. Consider a one-dimensional MPL system of the form

$$x(k+1) = \rho \otimes x(k)$$

with a growth rate equal to the eigenvalue of its state-update matrix, in this case, ρ. Figure 4-1 on page 42 shows a visual representation in the form of a line of the growth rate of such a system. Evidently, at every event step, the system's state grows with a value of ρ, a measure of the slope of the growth rate line. A flatter line relates to higher system throughput and is, therefore, often desirable.

Let us now consider the growth rate ρ_s of systems with a synchronised state evolution in greater detail. The cycle time vector ξ for a deterministic autonomous MPL system is unique and determined by the system's A-matrix.

To show this, consider a deterministic autonomous MPL system with irreducible matrix A that has a unique eigenvalue λ with associated eigenvector v. Then, for $x(0) = v$ [10]:

$$\begin{aligned} x(k) &= A^{\otimes^k} \otimes x(0) \\ &= \lambda^{\otimes^k} \otimes v \end{aligned}$$

for all $k \geq 0$, such that:

$$\lim_{k\to\infty} \frac{x_i(k)}{k} = \lambda, \quad \forall i \in \underline{n}$$

This result is independent of the initial condition $x(0)$ [10, Lemma 3.12]. The source also states that if a generalised eigenmode of an MPL system with a reducible matrix exists, the cycle-time vector exists and is still unique [10, Sec. 3.2.3]. It is apparent that, in most cases, MPL systems exhibit state evolution with a fixed cycle-time vector. This rate can, of course, be increased by applying a continuous control input $u(k)$ [12, §6].

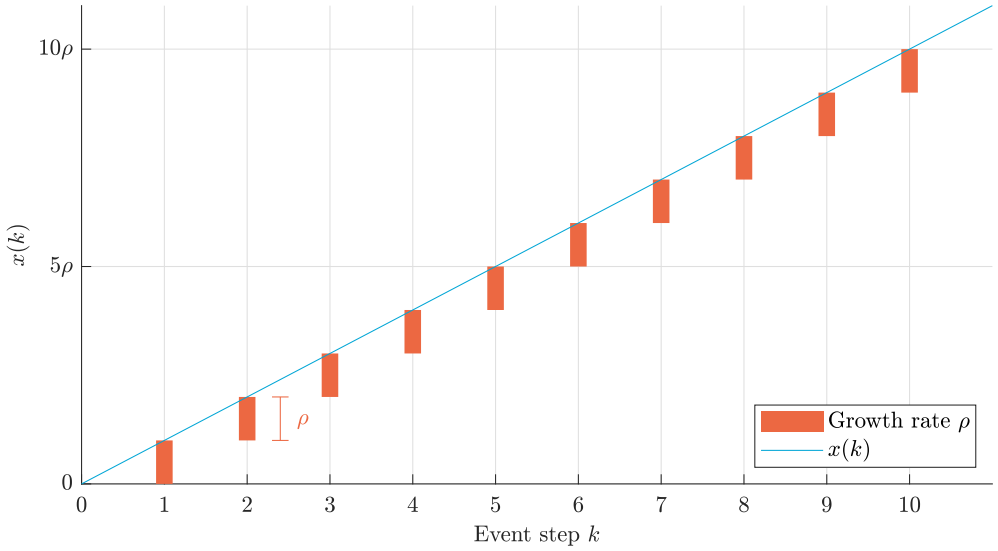

Figure 4-1: Visual representation of the state growth ρ of a one-dimensional MPL system $x(k+1) = \rho \otimes x(k)$. The orange rectangles represent the inter-event duration ρ. The growth rate is visualised using a line with slope ρ.

Contrarily, the cycle time vector for an SMPL system is generally dependent on the system's mode sequence, and is thus susceptible to discrete control. In some cases, discrete control can even reduce the growth rates of the SMPL system's states below their value for the individual MPL systems [6]. The range of achievable growth rates through discrete control can be upper and lower bounded. Without distinguishing between ρ_s and ρ_o, these bounds are denoted by $\overline{\rho}$ (pronounced 'rho bar') and $\underline{\rho}$ (pronounced 'rho underbar'), respectively. The following section shows that the upper bound $\overline{\rho}$ can be calculated using spectral theory, which is not true for the lower bound $\underline{\rho}$.

4-1-2 Maximum growth rate: $\overline{\rho}$

Contrary to MPL systems, switching systems may have a range of possible growth rates instead of a fixed one. In the case of continuous control of structurally controllable SMPL systems, it is straightforward to prove stabilisability for growth rates above $\overline{\rho}$, see Section 5-2 on page 68. When constraining the system to operate above $\overline{\rho}$, the value represents an upper bound on throughput. In the case of discrete control, the maximum growth rate serves as a lower bound on achievable throughput. Therefore, it is insightful to investigate the maximum growth rate $\overline{\rho}$ of such systems as a bound on their throughput.

To that end, let us denote the largest max-plus algebraic eigenvalue of a matrix $A \in \mathbb{R}_{\varepsilon}^{n \times n}$ as $\rho(A)$. This value is called the *spectral radius* of A. If A is irreducible, $\rho(A)$ is its unique eigenvalue. By definition, all associated eigenvectors v of A satisfy $v > 0$. That is, they have at least one finite element. Now, the maximum growth rate $\overline{\rho}$ of an autonomous MPL system with a reducible system matrix A is given by the spectral radius of its A-matrix [10, Sec. 3.2]. For any initial state that is a corresponding eigenvector, all states will grow with the rate $\rho(A)$.

Similarly, the joint spectral radius (JSR) measures the maximum growth rate of an autonomous switching system with multiple A-matrices, such as SMPL systems. Rota and Strang defined the JSR for the set of A-matrices $\mathcal{M} = \{A^{(1)}, \ldots, A^{(m)}\}$ of a system as [51]:

$$\rho(\mathcal{M}) \stackrel{\text{def}}{=} \limsup_{t\to\infty} \rho_t(\mathcal{M}),$$

where, for $\mathcal{M}$ expressed in the MPA, $\rho_t(\mathcal{M}) \stackrel{\text{def}}{=} \sup\{\rho(A)^{\otimes^{1/t}} : A \in \mathcal{M}^{\otimes^t}\}$. This computation is known to be NP-hard[1] [52, Theorem 1].

However, Gaubert found in 1995 that the JSR for max-plus algebraic systems is defined by only one matrix $S(\mathcal{M})$ [53]:

$$S(\mathcal{M}) \stackrel{\text{def}}{=} \bigoplus_{A^{(\ell)}\in\mathcal{M}} A^{(\ell)}$$

Then

$$\rho(\mathcal{M}) = \rho(S(\mathcal{M})). \tag{4-3}$$

In other words, the maximum growth rate $\overline{\rho} = \rho(\mathcal{M})$ is easily computed and is no larger than the largest entry in all matrices $A^{(\ell)}$, i.e., $\overline{\rho} \leq \max_{i,j,\ell}[A^{(\ell)}]_{ij}$. This quantification gives the growth rate of the slowest mode sequence possible as an upper bound to the growth rate of the autonomous SMPL system with system matrices in the set $\mathcal{M}$. The value may be larger than the largest eigenvalue of any individual system matrix $A^{(\ell)}$.

Van den Boom and De Schutter verify this result by proposing a dual solution using scaling via max-plus diagonal matrices. They arrive at the following definition:

Definition 4.2. [41, Def. 1] Consider a randomly switching max-plus linear (RSMPL) system with matrices $A_\alpha^{(\ell)}$ with $[A_\alpha^{(\ell)}]_{ij} = [A^{(\ell)}]_{ij} - \alpha$. The maximum growth rate $\overline{\rho}$ of the RSMPL system is the smallest α for which there exists a max-plus diagonal matrix $S = \text{diag}_\oplus(s_1, \ldots, s_n)$ with finite diagonal elements s_i, such that

$$[S \otimes A_\alpha^{(\ell)} \otimes S^{\otimes^{-1}}]_{ij} \leq 0, \quad \forall i, j, \ell$$

Defining $\rho' = \max_{i,j,\ell}[A^{(\ell)}]_{ij}$ yields:

$$[S \otimes A_{\rho'}^{(\ell)} \otimes S^{\otimes^{-1}}]_{ij} = [A_{\rho'}^{(\ell)}]_{ij} = [A^{(\ell)}]_{ij} - \rho' \leq 0, \quad \forall i, j, \ell.$$

The maximum growth rate $\overline{\rho}$ can then be easily computed by solving a linear programming problem.

As explained by Gupta et al., the maximum growth rate $\overline{\rho}$, or, equivalently, the JSR of the set of system matrices $A^{(\ell)}$, is equal to the smallest β that satisfies the following equation for all trajectories [6, Def. 3.2]:

$$\begin{aligned} &\exists \alpha, \beta \in \mathbb{R} \text{ with } \alpha \leq \beta \text{ s.t.} \\ &\alpha \otimes \mathbb{1}_n \leq \Delta x(k) \leq \beta \otimes \mathbb{1}_n, \quad \forall k \in \mathbb{N} \end{aligned} \tag{4-4}$$

[1] A problem A is NP-hard if it is at least as hard as some NP-complete problem B, in that B can be reduced to A in polynomial time. NP is short for "nondeterministic polynomial-time".

where $\mathbb{1}_n$ is an n-dimensional vector of all zeros and $x(k)$ is the system's state vector at event step k.

In summary, the JSR for SMPL systems is unique, easily computed, and may serve as both a lower and upper bound on their throughput. We denote the value by the maximum growth rate $\overline{\rho}$. For further information on the JSR, the reader is referred to [54, 55]. The following section discusses the concept of the minimum growth rate $\underline{\rho}$, related to the lesser-known lower spectral radius (LSR).

4-1-3 Minimum growth rate: $\underline{\rho}$

As introduced before, the minimum growth rate $\underline{\rho}$ is the lowest possible growth rate of a system. Following the example of Gupta, $\underline{\rho}$ is equal to the largest α for which Eq. (4-4) on page 43 is true. Essentially, this value is a strict lower bound on the growth rate achievable through control. It may be lower than the growth rate of the slowest mode.

The minimum growth rate $\underline{\rho}$ is equal to the lower spectral radius (LSR) of the set of matrices $\mathcal{M}$:

$$\underline{\rho} = \rho^*(\mathcal{M})$$

Contrary to the max-plus algebraic JSR and to the best of the author's knowledge, there is no scheme that calculates the max-plus algebraic LSR to any arbitrary precision. It is not the aim of this report to do so. Instead, we discuss an upper bound to the LSR in this section, finite-horizon estimates in Section 4-1-4 and stochastic growth rate definitions in Section 4-2 on page 46.

In recent years, Gupta et al. formalised an upper bound $\rho^*(\mathcal{M})$ of the max-plus algebraic LSR [6, Sec. 3.3]:

$$\rho^*(\mathcal{M}) = \min_{\ell \in \mathcal{L}} \min_{j \in \underline{r_\ell}} \{\lambda(A_{jj}^{(\ell)}) \mid V_j \text{ is an initial class}\}. \tag{4-5}$$

They assumed that there are r_ℓ classes for matrices $A^{(\ell)}$ in $\mathcal{M}$. V_j represents the vertex set $V(A^{(\ell)})$ corresponding to the submatrix $A_{jj}^{(\ell)}$ of $A^{(\ell)}$. The result is based on the supereigenvector theory by Butkovič [56, Section 5]. The reader is advised to consult Section 2-3 on page 16 for information on graph-theoretical concepts such as classes.

In summary, the minimum growth rate $\underline{\rho}$ is equal to the LSR of the system matrices of SMPL systems, but is not easily computed. Instead, we will propose an approximation of the value in the next section and evaluate them in Section 4-3 on page 53.

4-1-4 Finite-horizon approximations of the maximum and minimum growth rate

In some cases, it is helpful to consider a finite-horizon estimate of the growth rate. Such an approximation of the infinite-horizon counterparts may offer computational benefits, especially when calculating the minimum growth rate $\underline{\rho}$. It may also prove helpful when employing receding horizon control strategies as an estimation of growth rate over a certain horizon.

Thus, we introduce the definition of the output growth rate $\rho_{N_\mathrm{p}}(k)$ at event step k over a certain prediction horizon N_p:

$$\rho_{N_\mathrm{p}}(k) \stackrel{\text{def}}{=} \frac{C \otimes x(k+N_\mathrm{p}) - C \otimes x(k+N_\mathrm{t})}{N_\mathrm{p} - N_\mathrm{t}} \tag{4-6}$$

where $0 \leq N_\mathrm{t} < N_\mathrm{p}$, and N_t is chosen such that the influence of both transient behaviour as well as switching stochastics on the growth rate are minimised, keeping the following consequences in mind:

Larger N_t	$\rightarrow$	decreased influence from transients
Smaller N_t	$\rightarrow$	decreased influence from individual mode transitions

Although the definition in Eq. (4-6) is sensitive to transients and other small-scale behaviour, it is insusceptible to the synchronicity of the state evolution. Note that in the case of non-synchronised state evolution, the finite-horizon growth rate $\rho_{N_\mathrm{p}}(k)$ may be entirely driven by a single state and be oblivious to the dynamics of the others.

Now, define an upper bound $\overline{\rho}_{N_\mathrm{p}}(k)$ and lower bound $\underline{\rho}_{N_\mathrm{p}}(k)$ on the finite-horizon output growth rate:

$$\begin{aligned} \overline{\rho}_{N_\mathrm{p}}(k) &\stackrel{\text{def}}{=} \max_{x(k),\ldots,x(k+N_\mathrm{p})} \rho_{N_\mathrm{p}}(k) \\ \underline{\rho}_{N_\mathrm{p}}(k) &\stackrel{\text{def}}{=} \min_{x(k),\ldots,x(k+N_\mathrm{p})} \rho_{N_\mathrm{p}}(k) \end{aligned} \tag{4-7}$$

where $\tilde{v}_{N_\mathrm{p}}$ is a discrete control sequence of length N_p. For increasing N_p and finite N_t, these values converge to their asymptotic values:

$$\begin{aligned} \lim_{N_\mathrm{p}\to\infty} \overline{\rho}_{N_\mathrm{p}}(k) &= \overline{\rho} \\ \lim_{N_\mathrm{p}\to\infty} \underline{\rho}_{N_\mathrm{p}}(k) &= \underline{\rho} \end{aligned}$$

Note that in the case of implementing a control horizon N_c, the limit should be $\lim_{N_\mathrm{c}\to\infty}$. Furthermore, the minimisation over the variables $x(k), \ldots, x(k+N_\mathrm{p})$ in Eq. (4-7) may not lead to a violation of the system dynamics describing state evolution.

In the remainder of this chapter, we will extend the growth rate definitions and approximations toward stochastic systems and evaluate all approximations using statistical reasoning. These values will form the basis for stabilisability analysis in Chapter 5 on page 65.

4-2 Growth Rate of Stochastic SMPL Systems

The SMPL systems discussed in this chapter have not yet shown stochastic switching behaviour. However, one can imagine it is helpful to quantify the expected state evolution of a system, rather than to place conservative bounds on its behaviour. While stochastics do not influence the bounds $\overline{\rho}$ and $\underline{\rho}$, they do allow for the definition of expected growth rates. This section proposes mathematical definitions and finite-horizon approximations of these values.

4-2-1 Asymptotic (minimum) expected growth rate: $\overline{\overline{\rho}}$ and $\underline{\underline{\rho}}$

Two additional measures offer insight into these growth rates: the *expected growth rate*, and *minimum expected growth rate*. The expected growth rate of an autonomous stochastic SMPL system is denoted by $\overline{\overline{\rho}}$ (pronounced 'rho double bar'). The minimum expected growth rate of a stochastic SMPL system under discrete or hybrid control is represented by $\underline{\underline{\rho}}$ (pronounced 'rho double underbar'). To sum up, these four quantities measure the growth rate of stochastic SMPL systems:

- $\overline{\rho}$ Maximum growth rate under autonomous evolution
- $\overline{\overline{\rho}}$ Expected growth rate based on controlled stochastic evolution , given a discrete control sequence
- $\underline{\underline{\rho}}$ Minimum expected growth rate based on controlled stochastic evolution
- $\underline{\rho}$ Best case minimum growth rate based on controlled deterministic switching

By definition, the newly introduced stochastic growth rates are bounded by the previously considered bounds in the following way:

$$\underline{\underline{\rho}} \leq \overline{\overline{\rho}} \leq \overline{\rho}, \qquad \underline{\rho} \leq \underline{\underline{\rho}} \leq \overline{\overline{\rho}}$$

In the case of all system modes being equal, the SMPL system reduces to an MPL system and the four quantities coincide. The aforementioned ordinality is visualised in Figure 4-2. The growth rate lines in the figure correspond to a single one-dimensional system with growth rates defined as above but may be generalised to multi-dimensional SMPL systems. The sole aim of the visualisation is to convey the ordinality of the growth rate definitions for any SMPL system. The interpretation of the figure is similar to the one in Example 4.1 on page 41. It is essential to realise that these values and corresponding ordinality relate to the growth rate without continuous control. Depending on the system's B-matrices, the growth rate may be increased arbitrarily by employing a delay through $u(k)$.

As introduced before, the expected asymptotic output growth rate $\overline{\overline{\rho}}$ is related to the expected asymptotic average time between event occurrences. Since the switching stochastics often depend on a discrete control sequence $\tilde{v}_k$ of length k, we extend Eq. (4-1) on page 40 in the following way:

$$\mathbb{E}\left[\xi_i\right] = \mathbb{E}\left[\xi_i \mid \tilde{v}_k\right] \stackrel{\text{def}}{=} \lim_{k\to\infty} \frac{\mathbb{E}\left[x_i(k) \mid \tilde{v}_k\right]}{k}. \tag{4-8}$$

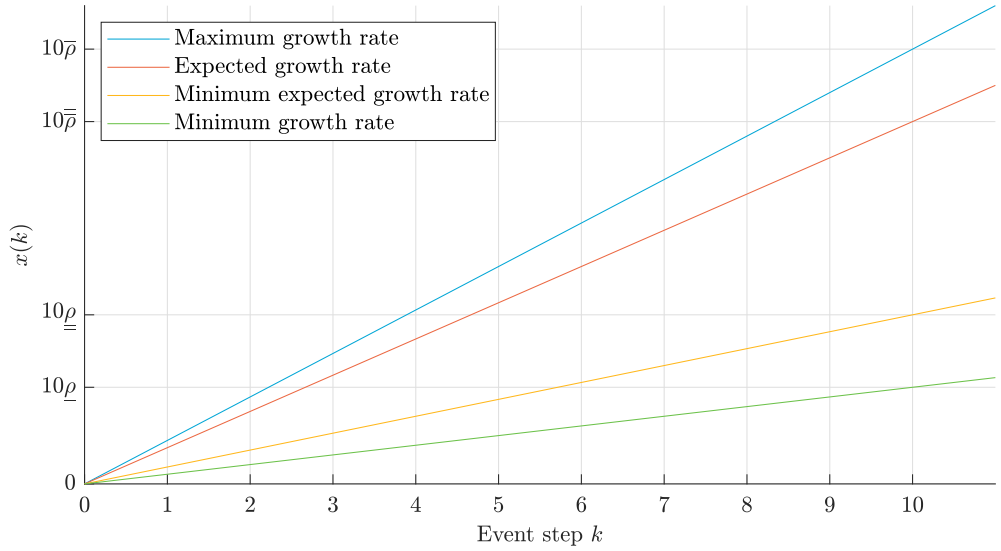

Figure 4-2: Four different state growth quantities $\overline{\rho}$, $\overline{\overline{\rho}}$, $\underline{\underline{\rho}}$ and $\underline{\rho}$ with $\underline{\rho} \leq \underline{\underline{\rho}} \leq \overline{\overline{\rho}} \leq \overline{\rho}$ for a single one-dimensional SMPL system. The figure aids in visualising the ordinality of the growth rate values.

Now, as an extension to Definition 4.1 on page 40, we notice that system evolution synchronicity also relates to the equality of the expectation of the asymptotic average growth rates of the state trajectories in Eq. (4-8):

$$\mathbb{E}\left[\xi_i\right] = \mathbb{E}\left[\xi_j\right] \stackrel{\text{def}}{=} \rho_{\text{s}}(\tilde{v}_k) = \rho_{\text{s}}, \quad \forall i, j, \in \underline{n}$$

Note that we again explicitly define the state growth rate as a function of a discrete control sequence $\tilde{v}_k$ of length k, since we assume that the switching stochastics may be altered by the control signal $v(k)$. For brevity, we may omit the dependency and write ρ_{s}, instead.

In the same manner, the output growth rate is defined as:

$$\rho_{\text{o}} = \rho_{\text{o}}(\tilde{v}_k) \stackrel{\text{def}}{=} \lim_{k\to\infty} \frac{\mathbb{E}\left[C \otimes x(k) \mid \tilde{v}_k\right]}{k}$$

Notice that the relation of Eq. (4-2) on page 40 holds true:

$$\begin{aligned}\rho_{\text{o}} = \lim_{k\to\infty} \frac{1}{k} \max_i \left(\mathbb{E}\left[x_i(k) \mid \tilde{v}_k\right] + C_i\right) &= \max_i \left(\lim_{k\to\infty} \frac{\mathbb{E}\left[x_i(k) \mid \tilde{v}_k\right]}{k} + C_i\right) \\ &= \max_i \left(\mathbb{E}\left[\xi_i \mid \tilde{v}_k\right] + C_i\right) = \max_i \left(\rho_{\text{s}}(\tilde{v}_k) + C_i\right) = \rho_{\text{s}}(\tilde{v}_k) + \max_i \left(C_i\right)\end{aligned}$$

Thus, if the system's state evolution is synchronised and $\max_i \left(C_i\right) = 0$, the values $\rho_{\text{s}}(\tilde{v}_k)$ and $\rho_{\text{o}}(\tilde{v}_k)$ still coincide. In case of asynchronous evolution, we derive that

$$\rho_{\text{o}} = \max_i \left(\lim_{k\to\infty} \frac{\mathbb{E}\left[x_i(k) \mid \tilde{v}_k\right]}{k} + C_i\right)$$

such that the difference of ρ_{o} with respect to the individual states is maximally $\|C\|_{\max} + \frac{1}{k}\|\mathbb{E}\left[x_i(k) \mid \tilde{v}_k\right]\|_{\mathbb{P}}$.

In this report, we define the expected growth rate and minimum expected growth rate as:

$$\begin{aligned} \overline{\overline{\rho}}(\tilde{v}_k) &\stackrel{\text{def}}{=} \rho_{\text{s}}(\tilde{v}_k) \\ \underline{\underline{\rho}} &\stackrel{\text{def}}{=} \min_{\tilde{v}_k} \rho_{\text{s}}(\tilde{v}_k) \end{aligned} \tag{4-9}$$

For brevity, we will often omit the dependence on $\tilde{v}$ and write $\overline{\overline{\rho}}(\tilde{v})$ as $\overline{\overline{\rho}}$.

To the best of the author's knowledge, no work has been done on evaluating or approximating the growth rates of stochastic SMPL systems. This report's aim is not to evaluate these infinite-horizon values to any arbitrary accuracy. However, we introduce an infinite-horizon approximation of the expected growth rate $\overline{\overline{\rho}}$ in Section 4-2-2 and finite-horizon approximations of $\overline{\overline{\rho}}$ and $\underline{\underline{\rho}}$ in Section 4-2-3 on page 50. The finite-horizon approximations are validated in Section 4-3 on page 53.

4-2-2 Infinite-horizon approximation of $\overline{\overline{\rho}}$

As a proposed algorithm for approximating the expected growth rate $\overline{\overline{\rho}}$ and based on unpublished work by Van den Boom, we extend Definition 4.2. The extension is based on finding the expected upper bound on the following max-plus matrix product:

$$\Gamma(N) \stackrel{\text{def}}{=} \bigotimes_{i=1}^{N} A^{(\ell_i)}$$

where N is the prediction horizon and $A^{(\ell_i)}$ is the system matrix for the ℓ-th mode at event step i, such that the system evolution according to a mode sequence $\tilde{\ell}$ is calculated as

$$x(k+N) = \Gamma(N) \otimes x(k), \quad \tilde{\ell} = \{\ell_1, \ldots, \ell_N\}$$

and, with a $x(k)$ and the switching stochastics as a function of the discrete control sequence $\tilde{v}$ known,

$$\mathbb{E}\left[x(k+N) \mid \tilde{v}\right] = \mathbb{E}\left[\Gamma(N) \mid \tilde{v}\right] \otimes x(k)$$

Thus, $\lim_{N\to\infty} \mathbb{E}\left[\Gamma(N) \mid \tilde{v}\right]$ measures the expected state growth rate. Assuming nonsynchronicity of the system's states, we further specify the measure to relate to the maximum element of this matrix, as a way of capturing individual state behaviour. Considering the dependence on $\tilde{v}$, we deduce that

$$\overline{\overline{\rho}} = \mathbb{E}\left[\overline{\Gamma(N)} \mid \tilde{v}\right] = \mathbb{E}\left[\overline{\bigotimes_{i=1}^{N} A^{(\ell_i)}} \mid \tilde{v}\right]$$

where $\overline{\Gamma(N)} = \max_{i,j} \left[\Gamma(N)\right]_{ij}$. Denote the probability $\mathbb{P}$ of switching to mode $m \in \underline{n_L}$ by $\mathbb{P}[\ell(k) = m \mid v(k)] = p_m$, where $\sum_{m\in\underline{n_L}} p_m = 1$.

Now, define

$$\alpha(\ell, S) \stackrel{\text{def}}{=} \overline{S^{-} A^{(\ell)} S}, \quad \text{for } \ell \in \underline{n_L}$$

and

$$\mathbb{E}\left[\alpha(\ell_r,\, S) \mid v\right] = \sum_{i=1}^{n_L} \alpha(i,\, S) \mathbb{P}\left[\ell_r = i \mid v\right] = \sum_{i=1}^{n_L} \alpha(i,\, S) p_i$$

where p_i has a dependence on the discrete control signal v. Then,

$$\begin{aligned}
\overline{\Gamma(N)} &= \overline{\bigotimes_{i=1}^{N} A^{(\ell_i)}} \\
&= \max_{i,j} \left[A^{(\ell_1)} A^{(\ell_2)} \cdots A^{(\ell_N)} \right]_{ij} \\
&= \max_{i,j} \left[SS^- A^{(\ell_1)} SS^- A^{(\ell_2)} \cdots SS^- A^{(\ell_N)} SS^- \right]_{ij} \\
&\leq \max_{i,j} \left[S \left(\max_{ij} \left[S^- A^{(\ell_1)} S \right] E \right) \left(\max_{ij} \left[S^- A^{(\ell_2)} S \right] E \right) \cdots \left(\max_{i,j} \left[S^- A^{(\ell_N)} S \right] E \right) S^- \right]_{ij} \\
&\leq \overline{SES^-} \alpha(\ell_1,\, S) \alpha(\ell_2,\, S) \cdots \alpha(\ell_N,\, S) \\
&\leq \overline{SES^-} + \alpha(\ell_1,\, S) + \alpha(\ell_2,\, S) + \ldots + \alpha(\ell_N,\, S)
\end{aligned}$$

And the expectation of $\overline{\Gamma(N)}$ is

$$\begin{aligned}
\mathbb{E}\left[\overline{\Gamma(N)} \mid \tilde{v} \right] &\leq \overline{SES^-} + \mathbb{E}\left[\alpha(\ell_1,\, S) \mid \tilde{v} \right] + \mathbb{E}\left[\alpha(\ell_2,\, S) \mid \tilde{v} \right] + \ldots + \mathbb{E}\left[\alpha(\ell_N,\, S) \mid \tilde{v} \right] \\
&= \overline{SES^-} + N \sum_{i=1}^{n_L} \alpha(i,\, S) p_i
\end{aligned}$$

Now, look for the matrix S that minimises the expectation of the state evolution matrix $\Gamma(N)$:

$$\min_S \overline{SES^-} + N \sum_{i=1}^{n_L} \alpha(i,\, S) p_i \tag{4-10}$$

A linear programming (LP) setup is used to find this upper bound by defining a vector $s \in \mathbb{R}^n$:

$$s_i = [S]_{i,i}$$

and

$$\begin{aligned}
\sigma_0 &\geq s_j - s_k, \quad \forall\, j, k \\
\sigma_i &\geq -s_j + [A^{(i)}]_{j,k} + s_k, \quad \forall\, i, j, k
\end{aligned}$$

Such that Eq. (4-10) can be recast as the following LP problem:

$$\begin{aligned}
\min_s \quad & \sigma_0 + N \sum_{i=1}^{n_L} \sigma_i p_i \\
\text{s.t.} \quad & \sigma_0 \geq s_j - \sigma_k, && \forall\, j, k \\
& \sigma_i \geq -s_j + [A^{(i)}]_{j,k} + s_k, && \forall\, i, j, k
\end{aligned}$$

Now, the result of this optimisation scheme is the smallest upper bound on the expected growth rate $\overline{\overline{\rho}}$ of the system.

The result above has been established as a function of a specific discrete control sequence $\tilde{v}$. However, this sequence can often be determined by a controller in order to, for instance, optimise throughput. In such a case, the goal becomes to minimise the expected growth rate, and we state:

$$\underline{\underline{\rho}} = \min_{\tilde{v}} \mathbb{E}\left[\overline{\Gamma}(N) \mid \tilde{v} \right]$$

The presented method of finding the upper bound on $\overline{\overline{\rho}}$ does not, in this form, allow for minimisation over $\tilde{v}$. It is recommended to investigate this addition in future work. Furthermore, although this analytic bound on $\overline{\overline{\rho}}$ is of value when a strict upper bound is desired, we will recommend its evaluation and validation for future research and place it outside the scope of this report. Instead, we will focus on additional finite-horizon approximations in the next section.

4-2-3 Finite-horizon approximation of $\overline{\overline{\rho}}$ and $\underline{\underline{\rho}}$

This section introduces finite-horizon approximations of the expected and minimum expected growth rates $\overline{\overline{\rho}}$ and $\underline{\underline{\rho}}$. These approximations offer a computationally efficient way of predicting uncertain system behaviour over a predefined horizon. We extend the definition of the finite-horizon growth rate approximation introduced in Eq. (4-6) on page 45 as:

$$\rho_{N_\mathrm{p}}(k,\, \tilde{v}_{N_\mathrm{p}}) \stackrel{\mathrm{def}}{=} \frac{\mathbb{E}\left[C \otimes x(k+N_\mathrm{p}) - C \otimes x(k+N_\mathrm{t}) \mid \tilde{v}_{N_\mathrm{p}}\right]}{N_\mathrm{p} - N_\mathrm{t}}.$$

This value is a representation of the expected growth rate over horizon N_p, whereas the minimisation of this value over $\tilde{v}_{N_\mathrm{p}}$ represents the case of highest throughput, and we define

$$\begin{aligned} \overline{\overline{\rho}}_{N_\mathrm{p}}(k,\, \tilde{v}_{N_\mathrm{p}}) &\stackrel{\mathrm{def}}{=} \rho_{N_\mathrm{p}}(k,\, \tilde{v}_{N_\mathrm{p}}) \\ \underline{\underline{\rho}}_{N_\mathrm{p}}(k) &\stackrel{\mathrm{def}}{=} \min_{\tilde{v}_{N_\mathrm{p}}} \rho_{N_\mathrm{p}}(k,\, \tilde{v}_{N_\mathrm{p}}) \end{aligned}$$

where $\overline{\overline{\rho}}_{N_\mathrm{p}}(k,\, \tilde{v}_{N_\mathrm{p}})$ is the expected growth rate for a certain discrete control sequence $\tilde{v}_{N_\mathrm{p}}$, and $\underline{\underline{\rho}}_{N_\mathrm{p}}(k)$ is the minimum achievable expected growth rate over horizon N_p. Here, too, the values converge to their asymptotic counterparts:

$$\lim_{N_\mathrm{p}\to\infty} \overline{\overline{\rho}}_{N_\mathrm{p}}(k,\, \tilde{v}_{N_\mathrm{p}}) = \overline{\overline{\rho}}(\tilde{v}_{N_\mathrm{p}})$$

$$\lim_{N_\mathrm{p}\to\infty} \underline{\underline{\rho}}_{N_\mathrm{p}}(k) = \underline{\underline{\rho}}$$

Note that $\overline{\overline{\rho}}$, too, has a dependency on $\tilde{v}_{N_\mathrm{p}}$, as shown in Eq. (4-9) on page 48.

The values of $\overline{\overline{\rho}}_{N_\mathrm{p}}$ and $\underline{\underline{\rho}}_{N_\mathrm{p}}$ can be calculated using a tree-search algorithm, in which every discrete control sequence is evaluated and weighted according to its probability of occurrence. Such an algorithm is shown in Algorithm 4.1 and implemented in the MATLAB script `growthRate.m` of Appendix A-2-10 on page 165. For improved efficiency, one can investigate evaluating only the sequences with a probability of occurrence higher than a certain threshold.

Algorithm 4.1. (Calculate (minimum) expected growth rate)
For a given N_p, denote with $\tilde{V}_{N_\mathrm{p}}$ the set of all control sequences $\tilde{v}^j_{N_\mathrm{p}}$ of length N_p and with $\tilde{L}_{N_\mathrm{p}}$ the set of all mode sequences $\tilde{\ell}^i_{N_\mathrm{p}}$ of length N_p. Predefine the parameter N_t, initial state $x(0)$ and initial mode $\ell(0)$, and do:

1. **For** each $\tilde{\ell}^i_{N_\mathrm{p}}$ in $\tilde{L}_{N_\mathrm{p}}$, **do**

1-1. Simulate the system with a predefined initial state $x(0)$ for N_p simulation steps using predefined mode sequence $\tilde{\ell}^i_{N_\mathrm{p}}$

1-2. Calculate the simulated system's growth rate based on Eq. (4-6) on page 45 and predefined N_t

1-3. Store growth rate as ρ_i

1-4. **For** each $\tilde{v}^j_{N_\mathrm{p}}$ in $\tilde{V}_{N_\mathrm{p}}$, **do**

1-4-1. Calculate the probability of occurrence of mode sequence $\tilde{\ell}^i_{N_\mathrm{p}}$ given discrete control sequence $\tilde{v}^j_{N_\mathrm{p}}$ and $\ell(0)$ and store as p_{ij}

1-4-2. Calculate weighted growth rate $\rho_i \cdot p_{ij}$ and store as $\tilde{\rho}_{ij}$

1-5. **End**

2. **End**

3. Calculate the expected growth rate for a given control sequence $\overline{\overline{\rho}}_{N_\mathrm{p}}(\tilde{v}^j_{N_\mathrm{p}}) = \sum_i \tilde{\rho}_{ij}$

4. Calculate the minimum expected growth rate as $\underline{\overline{\rho}}_{N_\mathrm{p}} = \min_j \overline{\overline{\rho}}_{N_\mathrm{p}}(\tilde{v}^j_{N_\mathrm{p}})$

5. Calculate the minimum growth rate as $\underline{\rho}_{N_\mathrm{p}} = \min_i \rho_i$

Section 4-3 on page 53 discusses the accuracy of the approximations found through Algorithm 4.1, and proposes methods to optimise it. The following section discusses an extension to account for the automaton-based mode constraints in the algorithm.

4-2-4 Extension to mode-constrained systems

Constraints on the mode sequence of a system, as introduced in Section 3-3 on page 34, influence the expected growth rate and its bounds. The consideration of these constraints may offer tighter bounds on the achievable growth rates and may change the expected and minimum expected growth rates. When denoting growth rate as a function of the constraint matrix C_{x_b} as $\rho(C_{x_b})$, we can say that:

$$\overline{\rho}(C_{x_b}) \leq \overline{\rho}$$
$$\underline{\rho}(C_{x_b}) \geq \underline{\rho}$$

These relationships are easily verified by realising that the set of possible mode sequences subject to constraints is a subset of all mode sequences. Thus, the minima and maxima of this subset cannot exceed the ones from the complete set. The same relationship, however, cannot be established for the finite-horizon approximations of these quantities since they do not represent bounds on the asymptotic values.

The value of $\overline{\rho}(C_{x_b})$ is represented by the constrained joint spectral radius (CJSR) $\hat{\rho}$ in literature as an extension to the JSR. The authors of [57] offer arbitrarily accurate approximation schemes for estimating the CJSR for systems with switching sequences constrained by an automaton. Given a relative accuracy $r > 0$, the algorithms compute an estimate of $\hat{\rho}$ within the range $[\hat{\rho},\, (1+r)\hat{\rho}]$.

Again, we redefine the finite-horizon growth rate approximations in the following way, as to incorporate both stochastics and constraints:

$$\begin{aligned}
\overline{\rho}_{N_\mathrm{p}}(k,\, C_{x_b}) &\stackrel{\text{def}}{=} \max_{x(k),\ldots,x(k+N_\mathrm{p})} \frac{[C \otimes x(k+N_\mathrm{p}) - C \otimes x(k+N_\mathrm{t}) \mid C_{x_b}]}{N_\mathrm{p} - N_\mathrm{t}} \\
\overline{\overline{\rho}}_{N_\mathrm{p}}(k,\, \tilde{v}_{N_\mathrm{p}},\, C_{x_b}) &\stackrel{\text{def}}{=} \frac{\mathbb{E}\left[C \otimes x(k+N_\mathrm{p}) - C \otimes x(k+N_\mathrm{t}) \mid \tilde{v}_{N_\mathrm{p}},\, C_{x_b}\right]}{N_\mathrm{p} - N_\mathrm{t}} \\
\underline{\underline{\rho}}_{N_\mathrm{p}}(k,\, C_{x_b}) &\stackrel{\text{def}}{=} \min_{\tilde{v}_{N_\mathrm{p}}} \frac{\mathbb{E}\left[C \otimes x(k+N_\mathrm{p}) - C \otimes x(k+N_\mathrm{t}) \mid \tilde{v}_{N_\mathrm{p}},\, C_{x_b}\right]}{N_\mathrm{p} - N_\mathrm{t}} \\
\underline{\rho}_{N_\mathrm{p}}(k,\, C_{x_b}) &\stackrel{\text{def}}{=} \min_{x(k),\ldots,x(k+N_\mathrm{p})} \frac{[C \otimes x(k+N_\mathrm{p}) - C \otimes x(k+N_\mathrm{t}) \mid C_{x_b}]}{N_\mathrm{p} - N_\mathrm{t}}
\end{aligned} \tag{4-11}$$

These constraints can be accounted for in Algorithm 4.1 on page 50 by excluding mode sequences or control sequences that violate them. That is to say; one should consider subsets $\tilde{V}^{\text{con}}_{N_\mathrm{p}}$ of $\tilde{V}_{N_\mathrm{p}}$ and $\tilde{L}^{\text{con}}_{N_\mathrm{p}}$ of $\tilde{L}_{N_\mathrm{p}}$, excluding the sequences in violation. The finite-horizon approximations in Eq. (4-11) are used throughout this report since they incorporate both constraints and stochastics. Additionally, they do not require synchronised state evolution.

4-3 Accuracy of the Finite-Horizon Approximations

The finite-horizon approximations proposed earlier in this chapter offer computationally efficient ways of estimating the infinite-horizon growth rates of constrained stochastic SMPL systems. The accuracy of these approximations is explored in more detail in this section. Specifically, the following sections offer insight into the influence of the parameters N_t and N_p in Eq. (4-11) and propose a framework for validating the approximations.

4-3-1 The influence of N_t on the approximation accuracy

The parameter N_t in Eq. (4-11) mitigates the effect of the initial state x_0 on the approximation of the growth rate. Naturally, the infinite-horizon definitions are independent of this initial value. Therefore, through this N_t, we aim to redefine the starting condition at a point where the state vector x has converged to a 'normal' value. 'Normal', in this case, refers to the condition that $\|x\|_\mathbb{P}$ has converged to a value within certain bounds such that the influence of x_0 has subsided.

When employing a single-output C-matrix with finite entries C_i, the output is a function of all states. As a result, there cannot be a hidden state, i.e., one that does not influence the output, larger than the rest by at least $\|C\|_\mathbb{P}$. Assuming this hidden largest state is x_i, and the smallest state that solely determines the output is x_j, then we have:

$$x_i + C_i < x_j + C_j$$

The positive difference $x_i - x_j$ is:

$$x_i - x_j < C_j - C_i \leq \max(C) - \min(C) = \|C\|_\mathbb{P}$$

If at one event step, the output $y(k)$ is equal to $x_j + C_j$ and the largest hidden state was x_i, then the output at the next event step $y(k+1)$ is upper bounded by:

$$y(k+1) \leq \max_j \left(\max_i \left(x_i + a_{ji} \right) + c_j \right) \leq x_i + \|A\|_{\max} + \|C\|_{\max}$$

And

$$\begin{aligned} y(k+1) - y(k) &\leq (x_i + \|A\|_{\max} + \|C\|_{\max}) - (x_j + C_j) \\ &= \|A\|_{\max} + \|C\|_{\max} + \|C\|_\mathbb{P} - C_j \\ &\leq \|A\|_{\max} + \|C\|_{\max} + \|C\|_\mathbb{P} - \|C\|_{\min} \\ &= \|A\|_{\max} + 2 \cdot \|C\|_\mathbb{P} \\ &\stackrel{\text{def}}{=} \overline{\Delta y} \end{aligned}$$

This upper bound on the difference between two system outputs without continuous control can be used as a guide to select N_t. Evidently, the initial output $y(k+N_\mathrm{t})$ cannot differ by more than $\overline{\Delta y}$ from $y(k+N_\mathrm{t}-1)$ and $y(k+N_\mathrm{t}+1)$. For systems with a C-matrix consisting only of zeros, such as the ones regarded in this report, the upper bound simplifies to $y(k+1) - y(k) \leq \|A\|_{\max}$. Note that this is not the system's growth rate; it is merely a measure of the maximum one-step influence of hidden states on the system's output.

Through switching, however, hidden states may stay hidden for any number of event steps. If this hidden state influences other states and the output only at certain modes that occur infrequently, one needs a large value of N_t to suppress the influence of x_0. The frequency of occurrence of such modes may be used as an inverse of a lower bound on N_t.

As a general guide, a designer may select N_t based on the prediction horizon N_p, the upper bound on the influence of hidden states as discussed above, and the general structure of the system matrices across all modes. For structurally observable systems with few infinite entries in their A-matrices, one can often get away with values not larger than the system's dimension.

4-3-2 The influence of N_p on the approximation accuracy

From their definitions, it is clear that for $N_\mathrm{p} \to \infty$, the finite-horizon approximations converge to their asymptotic values. One may conclude that, *in general*, a larger N_p leads to a closer approximation. However, it is not uncommon for SMPL systems to have non-constant first-order differences of their output signal for input signals with a constant rate of change. The discontinuous nature of the maximum operator that determines a system's states may cause the output signal to grow at different rates at each step. Figure 4-3 from Example 4.2 shows the difference between an exemplary tow-dimensional SMPL system's states $x_i(k)$ and their mean value $\bar{x}(k)$, visualising the effect that results in a jerky output signal. Due to this jerky nature, a lower value for N_p may result in a more accurate approximation of the system's growth rate than a higher one. In general, however, a larger N_t results in a better estimation since the state differences as percentages of the state values decrease.

Example 4.2. Consider a two-dimensional autonomous SMPL system with

$$A = \begin{bmatrix} \varepsilon & 1 \\ 0.5 & e \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad \text{and} \quad x(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

in its state-space representation

$$\begin{aligned} x(k) &= A \otimes x(k-1) \\ y(k) &= C \otimes x(k). \end{aligned}$$

It is apparent that

$$\begin{aligned} x_1(k) &= x_2(k-1) + 1 \\ x_2(k) &= \max\left(x_1(k-1) + 0.5,\, x_2(k-1)\right) \\ y(k) &= \max\left(x_1(k),\, x_2(k)\right), \end{aligned}$$

causing the two states to alternate in maximum value, as visualised in Figure 4-3. The mean value $\bar{x}(k)$ grows with a constant rate of 0.75, which is the unique max-plus eigenvalue of matrix A:

$$\begin{bmatrix} \varepsilon & 1 \\ 0.5 & e \end{bmatrix} \otimes \begin{bmatrix} 0.25 + \delta \\ \delta \end{bmatrix} = 0.75 \otimes \begin{bmatrix} 0.25 + \delta \\ \delta \end{bmatrix}.$$

As long as the initial state $x(0)$ is not a max-plus eigenvector corresponding to the system's max-plus eigenvalue, i.e., it is not of the form $\begin{bmatrix} 0.25 + \delta & \delta \end{bmatrix}^\top$ for any $\delta \in \mathbb{R}$, the system's states demonstrate jerky behaviour.

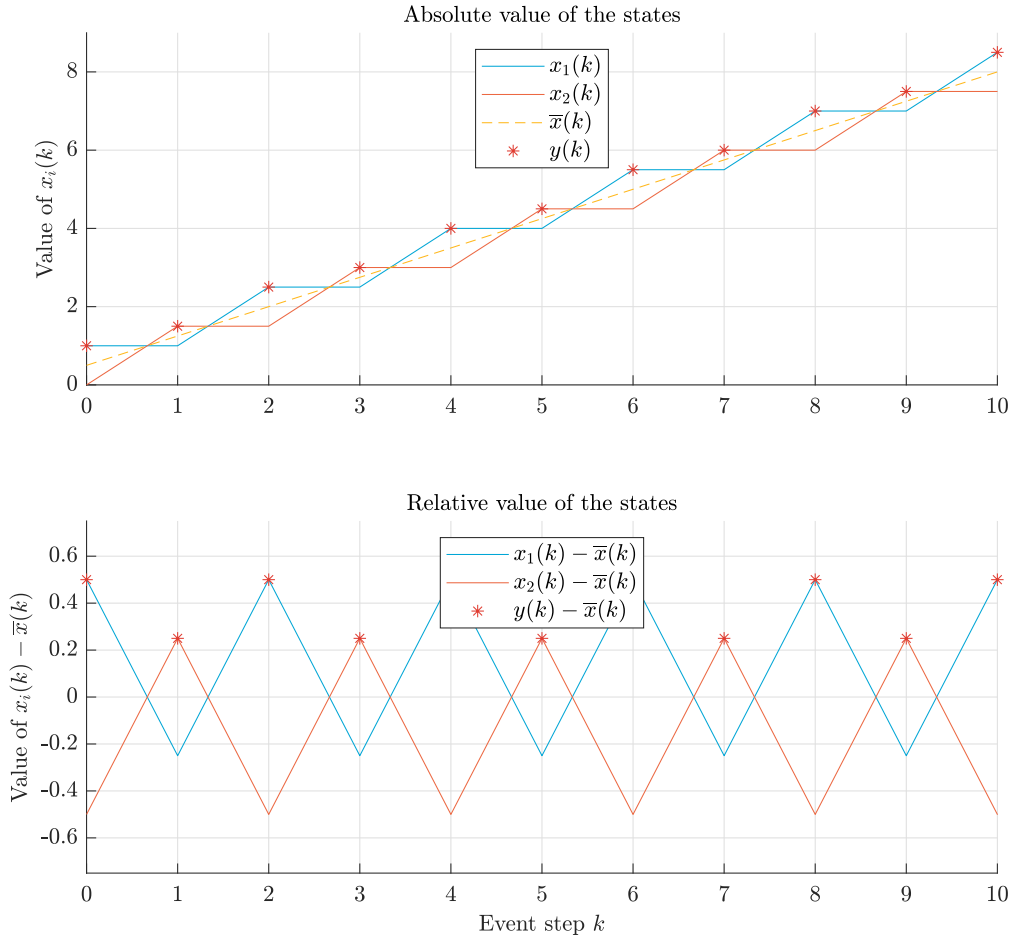

Figure 4-3: ABsolute and relative value of an example system's individual states $x_i(k)$, introduced in Example 4.2.

Importantly, there is a trade-off to be made between the accuracy of the approximation and computational efficiency—it is the reason for the existence of these approximations. Since calculating the expected and minimum growth rates of a system with n_L modes require evaluation of the probability of occurrence of all $n_L^{N_\mathrm{p}}$ possible mode sequences $\tilde{\ell}$ over the horizon N_p, it is often impractical to select horizons larger than $N_\mathrm{p} = 8$. If used in combination with a model predictive control (MPC) algorithm, it may be beneficial to choose the same horizon for the growth rate calculation as for the control algorithm.

In order to further isolate the growth rate as a result of a certain mode sequence with length N_p from the influence of the initial state and the jerky output signal, we employ another technique. While still considering only mode sequences with length N_p, we now evaluate the system's growth rate using repetitions of these sequences. Therefore, we obtain cyclic mode sequences of arbitrary total length but with a period of N_p. It is essential to acknowledge the discrepancy between this method and selecting longer values for the prediction horizon. Namely, we only search the subset of control sequences with a period of N_p, which would

result in finding a local growth rate minimum instead of a global one. Therefore, one can expect lower minimum growth rate values when extending N_p, but not when extending the number of repetitions of control sequences. Furthermore, the expected (minimum) growth rates are also skewed. Namely, when considering the hypothetical scenario of $N_\mathrm{p} = 1$, the algorithm evaluates only the probabilities of the n_L mode sequences consisting of the same modes:

$$\tilde{\ell}(k) = \{\ell(k) = i,\, \ell(k+1) = i,\, \dots\}\,, \quad \forall i \in \underline{n_L}$$

Other mode sequences are not evaluated, inevitably skewing the stochastic approximations. This way of scaling the simulation may be implemented in step 1-1 and step 1-4-1 of Algorithm 4.1 on page 50 by considering repetitions of the original $\tilde{\ell}_{N_\mathrm{p}}$ and $\tilde{v}_{N_\mathrm{p}}$ sequences.

As a guide, an engineer may select the largest value of N_p that still results in acceptable computation times. Alternatively, when paired with an MPC algorithm, they may choose to set equal the prediction horizon of the controller and the approximation algorithm. Furthermore, it is often wise to simulate the system using a repetition of the periodic mode sequences in order to isolate the system's growth rate as a function of its mode sequences.

4-3-3 Compare approximation to the empirical mean using a Monte Carlo algorithm

This section discusses the accuracy and correct interpretation of the finite-horizon growth rate approximations as introduced in this chapter, focusing on the minimum expected growth rate $\underline{\rho}_{N_\mathrm{p}}$. This value is deemed most important since it yields an upper bound of expected system throughput. A few characteristics complicate this investigation. Firstly, we have no way of analytically calculating the exact infinite-horizon value $\underline{\underline{\rho}}$ and its corresponding discrete control sequence $\tilde{v}$. The finite-horizon growth rate of a stochastic SMPL system follows an unknown distribution that we can only empirically sample from. Secondly, it is unknown how system parameters such as eigenvalues or structural characteristics such as the ε-structure influence the distribution. Thirdly, there is no accurate way to predict whether a certain control algorithm will succeed in steering the system along this optimal track since it may require infinite-horizon predictive capacity. Therefore, the investigation in this section is inherently limited. Section 7-2 on page 121 proposes valuable additions to the investigation for future research.

Using the theory presented in Section 2-4 on page 21 and considering the limitations described above, we will tackle the following problem:

Problem 4.1. For a given structurally observable system, find the finite-horizon approximation of the minimum expected growth rate $\underline{\rho}_{N_\mathrm{p}}$ as introduced in Section 4-2-3 on page 50 and its corresponding control sequence $\tilde{v}_{N_\mathrm{p}}$. Define $p(\gamma)$ with $\gamma > 0$ as the probability that the infinite-horizon value $\underline{\underline{\rho}}$ is not larger than $\underline{\rho}_{N_\mathrm{p}} + o$ with given offset o. Then, determine the number of samples N needed so that we can bound the error on this probability $p(\gamma)$ with value ϵ and confidence $1 - \delta$. Using a multisample of $\underline{\underline{\rho}}$ with cardinality N, find the offset o such that we can say with confidence $1 - \delta$ that $p(\gamma)$ is within an error ϵ of a desired value.

To that end, consider the finite-horizon approximation of the minimum expected growth rate $\underline{\rho}_{N_\mathrm{p}}$ achieved through a control sequence that is a repetition of the sequence $\tilde{v}_{N_\mathrm{p}}$. Then,

through simulation, approximate the actual growth rate ρ_{o} of the system under the same control sequence by considering the estimator $\rho_{N_{\mathrm{sim}}}$ that makes use of a much larger simulation horizon N_{sim}:

$$\rho_{N_{\mathrm{sim}}} = \frac{y(N_{\mathrm{sim}}) - y(1 + N_{\mathrm{t}})}{N_{\mathrm{sim}} - N_{\mathrm{t}} - 1}. \tag{4-12}$$

We assume that the stochastic variable $\rho_{N_{\mathrm{sim}}}$ is an unbiased estimator of the actual growth rate ρ_{o}, a claim substantiated through simulation. The following example elaborates on the validation of the claim.

Example 4.3. We consider a randomly generated trimodal three-dimensional structurally observable SMPL system with the following A and C matrices:

$$A^{(1)} = \begin{bmatrix} \varepsilon & \varepsilon & 0.2 \\ 1.1 & \varepsilon & 5.5 \\ 0.6 & 2.6 & \varepsilon \end{bmatrix},$$
$$A^{(2)} = \begin{bmatrix} \varepsilon & 1.0 & 3.9 \\ 2.5 & \varepsilon & 0.0 \\ \varepsilon & 2.6 & \varepsilon \end{bmatrix},$$
$$A^{(3)} = \begin{bmatrix} \varepsilon & \varepsilon & 1.4 \\ 4.2 & \varepsilon & \varepsilon \\ 1.1 & 3.3 & \varepsilon \end{bmatrix},$$
$$C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}.$$

The entries of the matrices used for the simulations have been rounded to one decimal such that the representations above are exact.

The system has been generated by randomly generating an ε-structure for the A-matrix of each mode such that, on average, half of the entries are ε. If the resulting matrix is irreducible, the remaining finite elements are generated from a uniform [0, 1] distribution and scaled such that the sum of all elements is equal to 10. If instead, the matrix is reducible, a new ε-structure is generated until the resulting matrix is irreducible. This way of generating systems is used throughout the research and ensures a similar scaling across all systems and across all modes while allowing for structural differences and individual element scalings. The system generation is implemented in MATLAB using `generateSystem.m` in Appendix A-2-9 on page 164 and supporting files `generateSupport.m` in Appendix A-2-8 on page 163 and `generateSemigroup.m` in Appendix A-2-7 on page 162.

The switching stochastics of the system used in this example are such that

$$P\left[L(k) = \ell(k) \mid v(k)\right] = \begin{cases} 0.8 & \text{for } \ell(k) = v(k) \\ 0.1 & \text{otherwise.} \end{cases}$$

Using repetitions of the randomly generated discrete control sequence $\tilde{v} = \{1, 2, 2, 2, 3\}$, we simulated the system 1000 times for ten different simulation horizons N_{sim}, logarithmically spaced in the range $[10^1, 10^4]$. The mean μ and standard deviation σ of distribution of the growth rate $\rho_{N_{\mathrm{sim}}}$ for the ten different simulation horizons using $N_{\mathrm{t}} = 3$ are visualised in Figure 4-4 on page 58. The experiment has been conducted for about 20 other randomly generated systems. We conclude that the mean of the distribution, unlike its variance, is

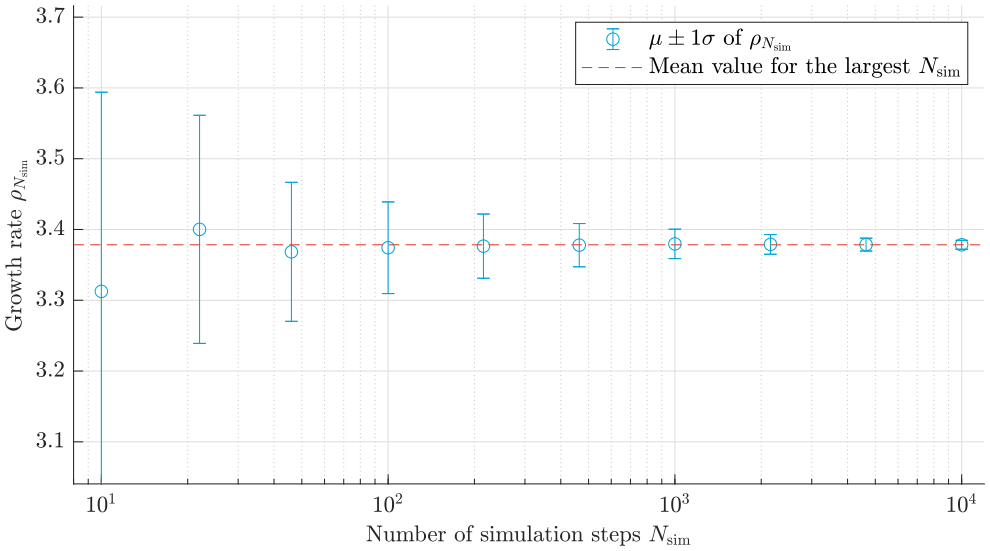

Figure 4-4: Mean and error bar with a length of 2σ of the growth rate $\rho_{N_{\text{sim}}}$ of a system evaluated after various numbers of simulation steps N_{sim}, using $N_{\text{t}} = 3$ and each averaged over 1000 runs. The system is described in Example 4.3 on page 57.

independent of the horizon N_{sim}, a finding that is true for all simulated systems. In other words, the estimator in Eq. (4-12) on page 57 appears to be an unbiased one.

Next, let us define the performance function $J_{\text{p}}(\rho_{N_{\text{sim}}})$ that is zero whenever the actual growth rate $\rho_{N_{\text{sim}}}$ of the system under discrete control according to the sequence $\tilde{v}$ is larger than the finite-horizon approximation of the minimum expected growth rate $\underset{=N_{\text{p}}}{\rho}$ minus some offset o:

$$J_{\text{p}}(\rho_{N_{\text{sim}}}) = \begin{cases} 0 & \text{if } \rho_{N_{\text{sim}}} \leq \underset{=N_{\text{p}}}{\rho} + o \\ 1 & \text{otherwise} \end{cases} . \tag{4-13}$$

Now, we wish to estimate the probability of performance $p(\gamma) = P\left[J_{\text{p}}(\rho_{N_{\text{sim}}}) \leq \gamma\right]$ given a performance level γ by finding $\widehat{p}_N(\gamma)$:

$$\widehat{p}_N(\gamma) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{\mathcal{B}_G}(\rho_{N_{\text{sim}}}^{(i)}),$$

where $\rho_{N_{\text{sim}}}^{(i)}$ is a single sample of the growth rate using a simulation horizon N_{sim}. The indicator function $\mathbb{I}_{\mathcal{B}_G}$ takes the form of

$$\mathbb{I}_{\mathcal{B}_G}(\rho_{N_{\text{sim}}}^{(i)}) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } \rho_{N_{\text{sim}}}^{(i)} \in \mathcal{B}_G \\ 0 & \text{otherwise} \end{cases}$$

and $\mathcal{B}_G = \left\{\rho_{N_{\text{sim}}}^{(i)} \in \mathcal{B}_{\mathbb{D}} \colon J_{\text{p}}(\rho_{N_{\text{sim}}}^{(i)}) \leq \gamma\right\}$. The unspecified set $\mathcal{B}_{\mathbb{D}}$ represents the bounds on the uncertainty of the system that is a result of the stochasticity in the mode switching. Since

the variance $\mathrm{Var}\left(\rho_{N_{\mathrm{sim}}}\right)$ depends on N_{sim} and the empirical mean $\mathbb{E}\left[\rho_{N_{\mathrm{sim}}}\right]$ inherently has a nonzero offset with respect to its finite-horizon approximation, the probability $p(\gamma)$ depends on N_{sim}, too. Therefore, a suitable N_{sim} should be selected a priori. It is recommended to select a value similar to the simulation horizon used in a control application, or, in general, any value for which you need to have probability of performance guarantees.

As a result of the definition in Eq. (4-13), any nonnegative performance level smaller than one, i.e., $\gamma \in [0,\, 1)$, will result in the same value of $p(\gamma)$, since $J_{\mathrm{p}}(\rho_{N_{\mathrm{sim}}})$ is either zero or one.

Next, define $p_i = \mathbb{I}_{\mathcal{B}_G}(\rho^{(i)}_{N_{\mathrm{sim}}})$ and $s_N = \sum_{i=1}^{N} p_i$. If we apply Theorem 2.6 on page 24 and observe that $p_i \in [0,\, 1]$, we obtain

$$P\left[|s_N - \mathbb{E}\left[s_N\right]| \geq \epsilon\right] \leq 2e^{-2\epsilon^2/N}.$$

If we now let $\widehat{p}_N(\gamma) = s_N/N$ such that $\mathbb{E}\left[p(\gamma)\right] = \mathbb{E}\left[\widehat{p}_N(\gamma)\right]$, we have

$$P\left[|\widehat{p}_N - p(\gamma)| \geq \epsilon\right] \leq 2e^{-2N\epsilon^2}. \tag{4-14}$$

Then, we may employ the Chernoff bound from Theorem 2.8 on page 24 to equate the right-hand side of Eq. (4-14) to δ and find the minimum number of samples for a given accuracy ϵ and confidence $1-\delta$:

$$N \geq \frac{1}{2}\epsilon^2 \log\frac{2}{\delta}$$

With this value of N, we can guarantee that the following holds:

$$P\left[|\widehat{p}_N(\gamma) - p(\gamma)| < \epsilon\right] > 1 - \delta$$

We know from Table 2-1 on page 25 that we need a minimum of 3.00×10^4 samples for an accuracy of $\epsilon = 0.01$ and confidence $1 - \delta = 0.995$. We will, in Example 4.4 on page 60, investigate for a single randomly generated system for which offset o we may say with precision $\epsilon = 0.01$ and certainty $1-\delta$ that for a fraction of 0.95 of all simulations, $\underline{\underline{\rho}} \leq \underline{\underline{\rho}}_{N_{\mathrm{p}}} + o$.

In the rest of this section, we will assume that $\mathbb{E}\left[\rho_{N_{\mathrm{sim}}}\right]$ obtained with the same control sequence used to find $\underline{\underline{\rho}}_{N_{\mathrm{p}}}$ is equal to $\underline{\underline{\rho}}$. In theory, there may be a control sequence that results in an even lower expected value. We will not consider that case since there exists no algorithm to obtain that sequence. Therefore, we will regard $\underline{\underline{\rho}}$, found through Algorithm 4.2, as an approximate upper bound to the infinite-horizon $\underline{\underline{\rho}}$.

Algorithm 4.2. (Performance verification and offset selection)
Given $\epsilon,\, \delta \in (0,\, 1)$, this algorithm returns a minimum offset o on the finite-horizon approximation of the minimum expected growth rate such that any probability of performance can be met with accuracy ϵ and confidence $1-\delta$, such that:

$$P\left[|\widehat{p}_N(\gamma) - p(\gamma)| < \epsilon\right] > 1 - \delta$$

1. Calculate finite-horizon approximation $\underline{\underline{\rho}}_{N_{\mathrm{p}}}$ of the minimum expected growth rate $\underline{\underline{\rho}}$ and find the corresponding discrete control sequence $\tilde{v}$.
2. Select N using Table 2-1 on page 25 or Eq. (2.8) on page 24.

3. Select suitable N_{sim} and N_{t} based on heuristics discussed in this section.
4. Select range of offset o based on $\underline{\rho}_{=N_{\text{p}}}$.
5. Draw N samples $\rho^{(i)}_{N_{\text{sim}}}$ by through simulation with a repetition of control sequence $\tilde{v}$.
6. Find for all values of o the empirical probability

$$\widehat{p}_N(0.5) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{\mathcal{B}_G}\left(\rho^{(i)}_{N_{\text{sim}}}\right)$$

   where $\mathbb{I}_{\mathcal{B}_G}$ is the indicator function of the set $\mathcal{B}_G = \{\rho_{N_{\text{sim}}} \colon J_{\text{p}}(\rho_{N_{\text{sim}}}) \leq 0.5\}$.
7. Select based on $\widehat{p}_N(0.5)$ as a function of the offset o the minimum offset necessary for obtaining a probability of performance within accuracy ϵ and confidence $1-\delta$.

The algorithm is implemented in the MATLAB script of Appendix A-2-6 on page 159.

Example 4.4. (Selection of offset o)
Consider a randomly generated three-dimensional trimodal structurally observable stochastic SMPL system with the following approximate A and C matrices:

$$A^{(1)} \approx \begin{bmatrix} \varepsilon & \varepsilon & 1.5126 \\ 1.9395 & \varepsilon & 1.0671 \\ 1.9174 & 2.2776 & 1.2858 \end{bmatrix},$$
$$A^{(2)} \approx \begin{bmatrix} 0.4229 & 0.5541 & 1.9906 \\ \varepsilon & 2.0655 & 1.7282 \\ 0.7906 & 1.0565 & 1.3915 \end{bmatrix},$$
$$A^{(3)} \approx \begin{bmatrix} \varepsilon & \varepsilon & 3.6583 \\ 1.9722 & \varepsilon & 3.4227 \\ \varepsilon & 0.9468 & \varepsilon \end{bmatrix}$$
$$C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}.$$

The system generation is according to the procedure given in Example 4.3 on page 57. Again, the switching stochastics are such that

$$P\left[L(k) = \ell(k) \mid v(k)\right] = \begin{cases} 0.8 & \text{for } \ell(k) = v(k) \\ 0.1 & \text{otherwise.} \end{cases}$$

The finite-horizon approximation of the system's minimum expected growth rate found using Algorithm 4.1 on page 50 with $N_{\text{p}} = 8$ and $N_{\text{t}} = 10$ is $\underline{\rho}_{=N_{\text{p}}} \approx 2.07$ with corresponding control sequence used to obtain the value denoted as $\tilde{v}$. We simulate the system using a repetition of $\tilde{v}$ a total of 3.00×10^4 times using four simulation horizons of $N_{\text{sim}} \in \{100, 200, 10^3, 10^4\}$ and calculate the growth rate according to Eq. (4-12) on page 57 using $N_{\text{t}} = 10$. Then, for 1000 linearly spaced offset values $o \in [-0.2, 0.2]$, we checked what fraction of the 3.00×10^4 runs resulted in a growth rate smaller than $\underline{\rho}_{=N_{\text{p}}} + o$, by selecting $\gamma = 0.5$.

Figure 4-5 on page 62 and Figure 4-6 on page 63 show the investigation results. The first figure visualises the value of $\widehat{p}_N(0.5)$ as a function of the offset o for four different values of N_{sim}. The latter shows the distribution of the sampled $\rho^{(i)}_{N_{\text{sim}}}$ values. As expected, an offset o of zero results in a probability of performance of around 0.5. Naturally, for an unbiased estimator, about half of the draws will be on one side of the actual value. Furthermore, the mean of the distributions shown in Figure 4-6 remains relatively unchanged while the variance decreases. This decrease is apparent in the first figure, where the necessary offset to obtain a certain probability of performance decreases as N_{sim} increases. The results may be interpreted as follows:

> When simulating the system for 100 steps, we can say with confidence 0.995 that in 95 % of the cases, with an accuracy of 1 %, this particular system under hybrid control will be able to follow a growth rate of $\underline{\underline{\rho}}_{N_{\text{p}}} + o$ with offset $o = 0.068$. For simulation horizons of 200, 1000 and 10,000, an offset of 0.048, 0.022 or 0.007, respectively, suffices. For illustration, the respective offsets are 3.3 %, 2.3 %, 1.1 % and 0.3 % of the growth rate $\underline{\underline{\rho}}_{N_{\text{p}}}$. These values cannot be expected to hold for other systems. Furthermore, this conclusion does not consider whether a control algorithm will be able to select the appropriate discrete control inputs. Lastly, it does not consider any discrete control sequences that would result in growth rates lower than $\underline{\underline{\rho}}_{N_{\text{p}}}$.

The algorithm performed by MATLAB version R2022a takes, respectively and on average, 61 s, 69 s, 135 s and 869 s to complete for the four values of N_{sim} on an HP ZBook Studio G5 with an Intel Core i7-8750H CPU with a base clock speed of 2.20 GHz and 16.0 GB of RAM, using parallelisation of all six CPU cores. The MATLAB script used to perform the calculations is included in Appendix A-2-6 on page 159.

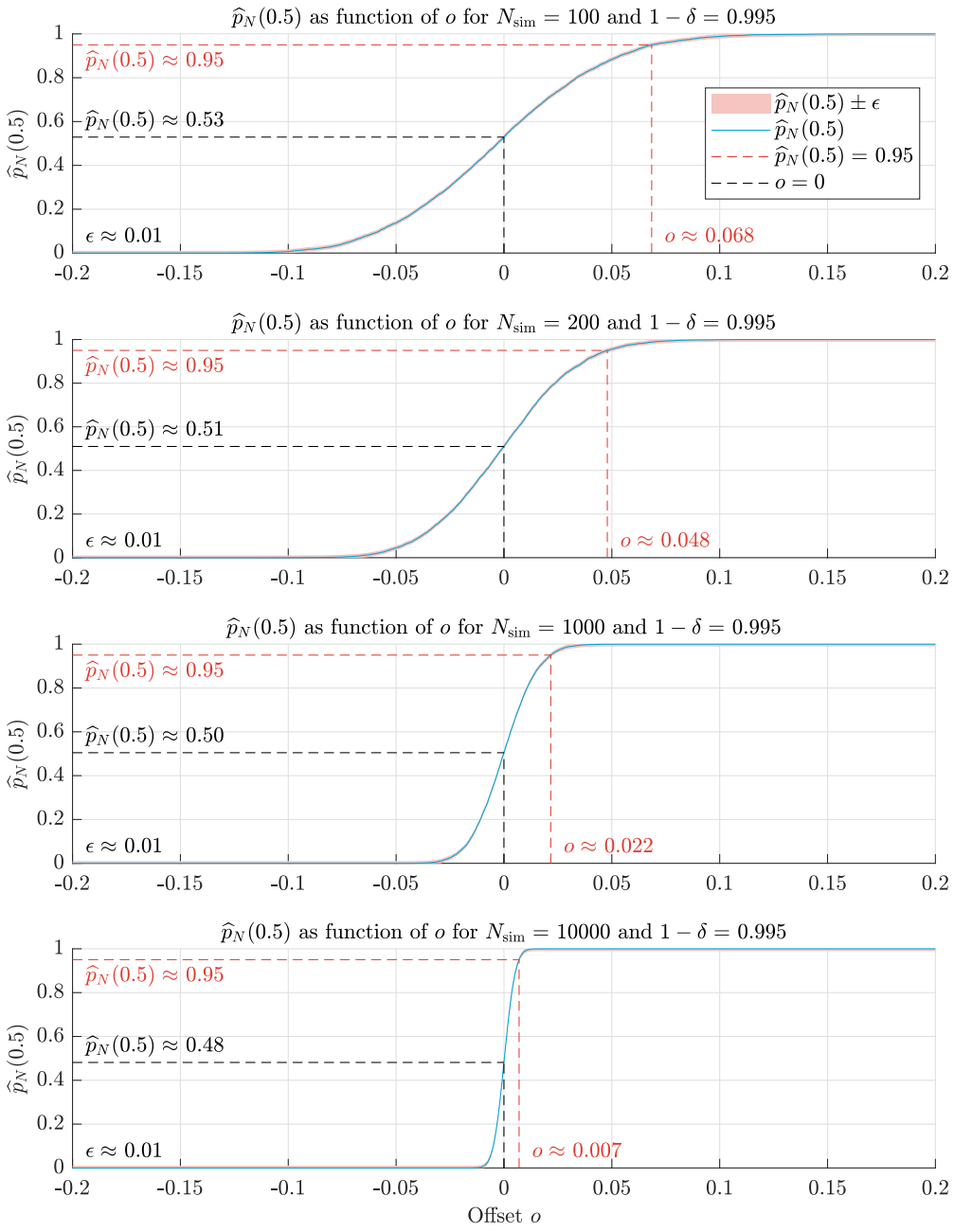

Figure 4-5: The value of $\widehat{p}_N(0.5)$ as a function of the offset o and simulation horizon N_{sim} as calculated using Algorithm 4.2 on page 59 for the system of Example 4.4 on page 60. It shows the value of $\widehat{p}_N(0.5)$ for a zero offset, and the minimum offset necessary for obtaining $\widehat{p}_N(0.5) = 0.95$. Furthermore, the red bounds around the plot represent the uncertainty $\epsilon = 0.01$ with confidence $1 - \delta = 0.995$. The reader is referred to Example 4.4 on page 60 for an elaborate interpretation of the figure.

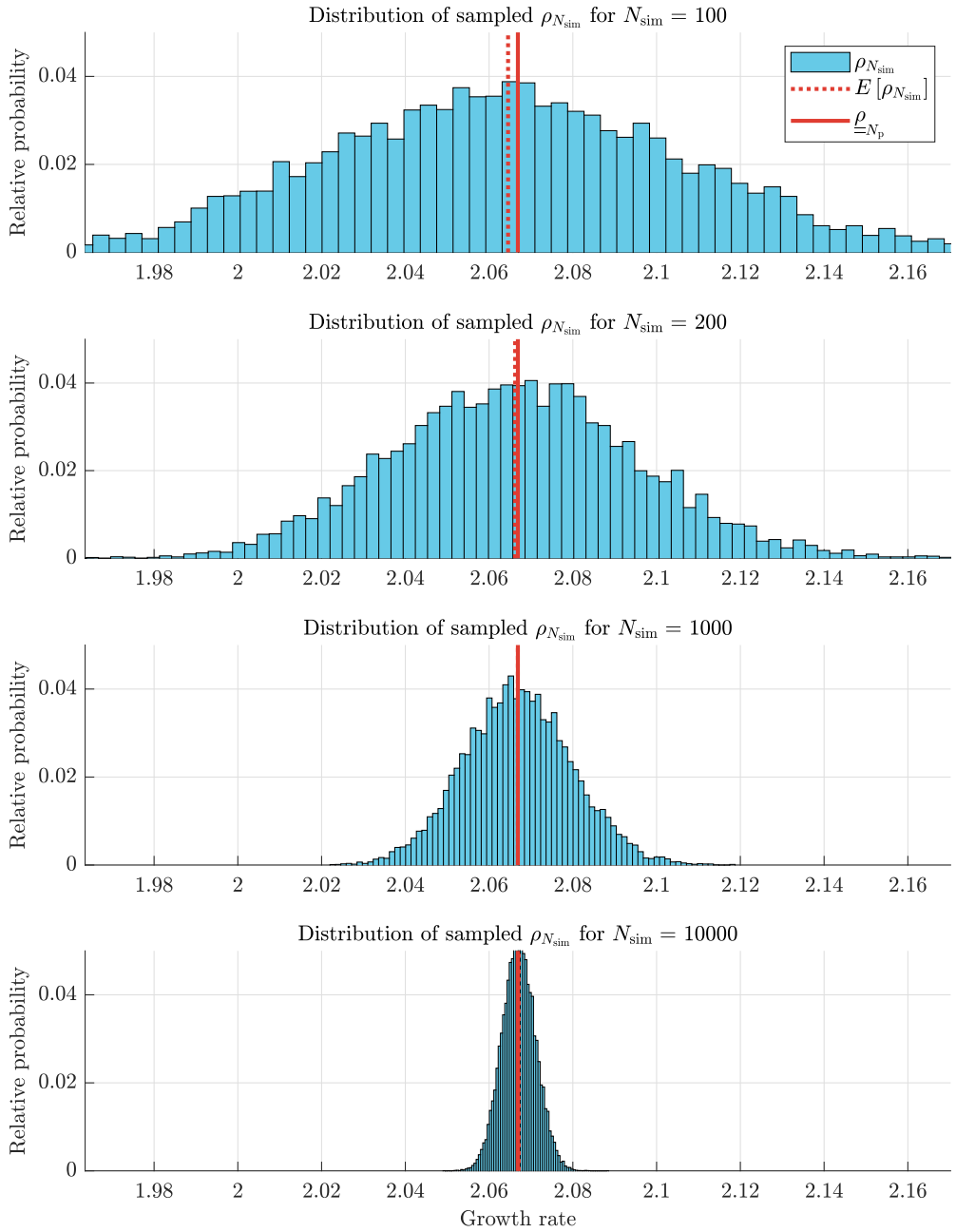

Figure 4-6: The distribution of $\rho_{N_{\text{sim}}}$, its mean value $\mathbb{E}[\rho_{N_{\text{sim}}}]$ and the finite-horizon approximation of the minimum expected growth rate $\underline{\underline{\rho}}_{N_{\text{p}}}$. The distribution is shown for four different values of N_{sim}.

Chapter 5

MPC Stabilisability of SMPL Systems

Based on the notion of growth rate, this chapter explores the conditions for stability and stabilisability of switching max-plus linear (SMPL) systems. It proposes a control framework that considers these notions and that can be applied to mode-constrained systems with uncertain switching behaviour. Section 5-1 on page 66 introduces the concept of stability for SMPL systems, Section 5-2 on page 68 discusses stabilisability, and Section 5-3 on page 72 proposes a stabilising model predictive controller.

5-1 Stability of SMPL systems

As discussed in Section 4-1 on page 40, the notion of stability for discrete-event system (DES) systems relates to the boundedness of their buffer levels or, alternatively, their growth rate. This relation to growth rate results in a straightforward stability analysis for systems with a unique eigenvalue but presents difficulties when dealing with other systems, such as SMPL systems. In recent years, Gupta et al. presented multiple frameworks for analysing the stability of autonomous SMPL systems. They introduced the notions of *(asymptotic) max-plus weakly bounded-buffer stability* and *(asymptotic) max-plus strongly bounded-buffer stability* that are related to the boundedness of the buffer levels within the same event cycle [5, Def. 4.1–4.2]. Conversely, they propose the notion of *(asymptotic) max-plus Lipschitz stability* that considers the buffer levels associated to time delays in consecutive event cycles [5, Def. 4.3]. The latter concept does not imply the bounded-buffer stability of the first two concepts. The paragraphs below introduce their definitions of the stability criteria.

(Asymptotic) max-plus weakly bounded-buffer stability This form of stability is posed by Gupta et al. and relates to the boundedness of the state trajectories within the same event cycle:

Definition 5.1. [5, Def. 4.1] An autonomous DES is said to be max-plus weakly bounded-buffer stable if for every $x_0 \in \mathbb{R}^n_\varepsilon$, there exists a bound $M_{\mathrm{x}}(x_0) \in \mathbb{R}$ such that the states are bounded in the projective norm, defined in Section 2-1-4 on page 10:

$$\|x(k;x_0)\|_{\mathbb{P}} \leq M_{\mathrm{x}}(x_0), \quad \forall k \in \mathbb{N} \tag{5-1}$$

Asymptotic max-plus weakly bounded-buffer stability is achieved if the norm in Eq. (5-1) attains a constant value:

$$\lim_{k\to\infty} \frac{1}{k}\|x(k;x_0)\|_{\mathbb{P}} = 0. \tag{5-2}$$

(Asymptotic) max-plus strongly bounded-buffer stability The same work proposes a definition of stability suitable for when the time delays between the states are also constrained:

Definition 5.2. [5, Def. 4.3] An autonomous DES is said to be max-plus strongly bounded-buffer stable if the state are bounded with respect to each other in the supremum norm, as defined in Section 2-1-4 on page 10. Equivalently, the state trajectory is contained in a finitely generated set $\mathcal{K} \subseteq \mathbb{R}^n$:

$$\begin{aligned} &\exists Q \in \mathbb{R}^{n\times n}, \quad [Q]_{i,j} = q_{i,j} \text{ and} \\ &\exists \mathcal{K} = \{x \in \mathbb{R}^n \mid x_i - x_j \geq q_{j,i},\ \forall i,j \in \underline{n}\} \neq \emptyset \end{aligned} \tag{5-3}$$

such that $x(k;x_0) \in \mathcal{K}, \quad \forall k \in \mathbb{N}$. The max-plus cone $\mathcal{K}$ is finitely generated if Q is an irreducible matrix [58]. Asymptotic max-plus strongly bounded-buffer stability is achieved if the component-wise bounds attain a constant value:

$$\lim_{k\to\infty} \frac{1}{k}(x_i(k) - x_j(k)) = 0, \quad \forall i,j \in \underline{n}$$

The set $\mathcal{K}$ in Eq. (5-3) can be represented as a max-plus cone:

$$\mathcal{K} = \{x \in \mathbb{R}^n \mid Q \otimes x \leq x\}.$$

(Asymptotic) max-plus Lipschitz stability The authors propose a different form of stability that relates the state trajectories of consecutive event cycles:

Definition 5.3. [6, Def. 3.2] An autonomous DES is said to be max-plus Lipschitz stable if there exist upper and lower bounds ($\alpha, \beta \in \mathbb{R}$) on the first-order difference of the state trajectories:

$$\alpha \otimes \mathbb{1}_n \leq \Delta x(k) \leq \beta \otimes \mathbb{1}_n, \forall k \in \mathbb{N}$$

These inequalities suggest minimum and maximum duration requirements between two consecutive events. These conditions hold for max-plus linear (MPL) systems over finite state evolution if the system matrix A is regular [59].

Stability for systems with a due date As discussed, synchronised system evolution implies adherence to the condition in Eq. (5-2) for asymptotic max-plus weakly bounded-buffer stability. This boundedness, however, only applies to the states within the same event cycle. It does not dictate the system's behaviour over a certain period. In many cases, SMPL systems are subject to a reference signal that follows a desired throughput $\frac{1}{\mu}$, i.e., systems with a due date. Therefore, we discuss the condition for stability that the system's output should remain bounded with respect to the reference signal. It is known that all buffer levels of structurally observable systems, as introduced in Definition 3.3 on page 30, are bounded if their dwelling times, i.e., the time differences between $u(k)$ and $y(k)$, remain bounded [60]. Therefore, we consider a SISO structurally observable SMPL system stable if there exist finite constants k_0, M_{yr}, M_{yx} and M_{xu} such that

$$|y(k) - r(k)| \leq M_{yr}, \tag{5-4a}$$
$$|y(k) - x_i(k)| \leq M_{yx}, \quad \forall i \tag{5-4b}$$
$$|x_j(k) - u(k)| \leq M_{xu}, \quad \forall j \tag{5-4c}$$

for all $k \geq k_0$. The condition in Eq. (5-4a) bounds the difference between the reference signal and the system output. It does not specify whether a system should be quicker or slower than this signal. The conditions in Eq. (5-4b) and Eq. (5-4c) bound the internal dwelling times of the system.

Note that a system that meets these conditions is max-plus weakly bounded-buffer stable and max-plus Lipschitz stable. Furthermore, recognise that stability is not an intrinsic feature of SMPL systems but depends on the reference signal $r(k)$. The following section explores the stabilisability of SMPL systems as a function of this signal.

5-2 Stabilisability of SMPL Systems

This section aims to define conditions for which structurally controllable and structurally observable SMPL systems are stabilisable. For the same setting, Gupta et al. define the following [6]:

Definition 5.4. [6, Def. 3.1] A DES is *stabilisable* if there exist control inputs such that the system evolution becomes synchronised:

$$\begin{aligned} &\exists\, \mu \in \mathbb{R} \quad \text{s.t.} \\ &\xi_i = \xi_j = \mu, \quad \forall\, i,\, j \in \underline{n}. \end{aligned}$$

Recall that this relates to the concept of synchronised system evolution explained in Definition 4.1 on page 40 and that μ is the reciprocal of the system throughput.

This definition, however, does not directly relate to the stability criteria posed in Eq. (5-4) on page 67. To form stabilisability conditions for systems with a due date, we consider a strictly increasing one-dimensional reference signal $r(k)$ of the form:

$$r(k) = \rho_{\mathrm{r}}\, k + c(k), \quad \text{where } |c(k)| \leq c_{\max} \text{ and } \rho_{\mathrm{r}} > 0 \tag{5-5}$$

where $\rho_{\mathrm{r}} \in \mathbb{R}$ is the desired growth rate and $c(k) > c(k-1) - \rho_{\mathrm{r}}$.

Now, Van den Boom and De Schutter state conditions and an input signal $u(k)$ that will stabilise an SMPL system. The reader is advised to consult their work for the proof.

Theorem 5.1. [4, Theorem 1] Consider a structurally observable and structurally controllable SMPL system with maximum growth rate ρ_{s} and consider the reference signal in Eq. (5-5) with growth rate ρ_{r}. Now, if $\rho_{\mathrm{r}} > \overline{\rho}$, then any input signal

$$u(k) = \rho_{\mathrm{r}}\, k + \nu(k), \quad \text{where } |\nu(k)| \leq \nu_{\max}, \quad \forall k \tag{5-6}$$

will stabilise the SMPL system for a finite value $\nu_{\max}$ according to the criteria in Eq. (5-4) on page 67.

The theorem is powerful in that it ensures stability for the rather lenient conditions of structurally observability and controllability and all reference signals with a slope larger than the system's maximum growth rate. This last condition, however, is a restrictive one. The condition is a logical result of the application of continuous control, but it is constraining for hybrid or discrete control. Similarly, the authors of Definition 5.4 prove that under continuous control, a max-plus Lipschitz stable DES is stabilisable only for $\mu \geq \overline{\rho}$ [6, Theorem 4.2]. However, as discussed in Chapter 4 on page 39, even under continuous control, SMPL systems may exhibit growth rates lower than this value $\overline{\rho}$. With the application of discrete or hybrid control, one may even aim to predictably stabilise the system with a growth rate below the maximum growth rate $\overline{\rho}$.

This report investigates the conditions for stabilisability regarding reference signals with a growth rate $\rho_{\mathrm{r}} \leq \overline{\rho}$. It considers the effects of the different forms of control, the inclusion of mode constraints as introduced in Section 3-3 on page 34 and uncertainty in the system's switching behaviour as discussed in Section 3-2 on page 31. Furthermore, it explores the additional requirement of synchronised evolution.

We distinguish between the following three problems:

Problem I: Formulate conditions for stabilisability of a mode-constrained deterministic SMPL system under discrete control for a reference signal with a growth rate $\rho_r \leq \overline{\rho}$.

Problem II: Formulate conditions for stabilisability of a mode-constrained deterministic SMPL system under hybrid control for a reference signal with a growth rate $\rho_r \leq \overline{\rho}$.

Problem III: Formulate conditions for stabilisability of a mode-constrained stochastic SMPL system under hybrid control for a reference signal with a growth rate $\rho_r \leq \overline{\rho}$.

These problems are discussed in Section 5-2-1, Section 5-2-2 on page 70 and Section 5-2-3 on page 70, respectively. Furthermore, they are investigated in the case studies of Chapter 6 on page 87.

5-2-1 Stabilisability of a mode-constrained deterministic system under discrete control

In Problem I, we explore the conditions for stabilisability of a deterministic SMPL system under discrete control subject to a reference signal of the form in Eq. (5-5) with a growth rate ρ_r smaller than the maximum growth rate $\overline{\rho}$.

The theorem below provides sufficient conditions for synchronised evolution through discrete control:

Theorem 5.2. [6, Theorem 4.1] A max-plus Lipschitz stable autonomous SMPL system can be synchronised by discrete control for *some* $\rho_r \in [\rho^*(\mathcal{M}, \overline{\rho}]$ if the semigroup generated by the matrices in $\mathcal{M}$ is irreducible. Here, $\rho^*(\mathcal{M})$ is the upper bound of the max-plus algebraic lower spectral radius (LSR) in Eq. (4-5) on page 44.

A semigroup is a set combined with an associative binary operation. For example, a set $\mathcal{A}$ of regular square matrices in the max-plus algebra (MPA) of dimension n forms a multiplicative semigroup [6]:

$$\Psi(A) \stackrel{\text{def}}{=} \left\{ A^{(i_1)} \otimes \ldots \otimes A^{(i_k)} \mid A^{(i_j)} \in \mathcal{A},\ j \in \underline{k},\ k \in \mathbb{N} \right\}$$

They note that the theorem provides only a sufficient condition; it may also work for reducible semigroups. Due to multiple irreducible matrices in the semigroup $\Psi(A)$, it is hard to define the set of achievable growth rates. Future research may find stricter bounds on the set.

The theorem assumes SMPL systems with finite periodic mode sequences since those can be reduced to MPL systems that achieve asymptotic max-plus Lipschitz stability [6]. Under this assumption, the theorem illustrates that stabilisability can only be predictably achieved for reference signals with some growth rates in the proposed range.

We continue on this theorem by proposing the following hypothesis:

Hypothesis 5.1. Any max-plus Lipschitz stable and structurally observable SMPL system can be stabilised by discrete control for *some* $\rho_r \in [\underline{\rho}, \overline{\rho}]$ including $\underline{\rho}$ but not necessarily $\overline{\rho}$.

This hypothesis is believed to hold true since Theorem 5.2 implies synchronised evolution, i.e., bounded buffer levels, for some growth rates in this range, whereas the addition of structural observability implies adherence to the stability conditions in Eq. (5-4) on page 67. Note that the lower limit of the range has been extended to the minimum growth rate $\underline{\rho}$ since we no longer assume periodic mode sequences of finite length. However, it is not a given that the system is able to follow the maximum growth rate $\overline{\rho}$ that acts as a conservative upper bound.

The inclusion of mode-constraints represented by the constraint matrix C_{x_b} tightens the proposed range to $[\underline{\rho}(C_{x_b}), \overline{\rho}(C_{x_b})]$ and may reduce the number of growth rates for which the system is stabilisable. The theory presented in Section 4-3 on page 53 sheds light on approximations of the minimum growth rate $\underline{\rho}$ and their interpretation for individual systems.

5-2-2 Stabilisability of a mode-constrained deterministic system under hybrid control

In this paragraph, we explore the conditions for stabilisability of a deterministic SMPL system under hybrid control subject to a reference signal of the form in Eq. (5-5) on page 68 with a growth rate ρ_{r} smaller than the maximum growth rate $\overline{\rho}$.

Theorem 5.3. [6, Theorem 4.3] A max-plus Lipschitz stable SMPL system can be synchronised by hybrid control for *some* $\rho_{\mathrm{r}} \in [\rho^*(\mathcal{M}), \overline{\rho}]$ if the system is weakly structurally controllable.

Again, the authors assume a periodic mode sequence of finite length and an input signal with a growth rate $\rho_u \leq \overline{\rho}$.

We propose the following hypothesis:

Hypothesis 5.2. Any max-plus Lipschitz stable, structurally observable and structurally controllable SMPL system can be stabilised by hybrid control for *all* $\rho_{\mathrm{r}} \geq \underline{\rho}$.

Recall from Definition 3.1 on page 30 and [6, Theorem 4.2] that structural controllability for max-plus Lipschitz stable SMPL systems under continuous control implies that the growth rate of a system can be steered towards any value larger than $\overline{\rho}$. Then, if Hypothesis 5.1 dictates that the system can be stabilised for $\rho_{\mathrm{r}} = \underline{\rho}$, then such a system under hybrid control can be stabilised for all $\rho_{\mathrm{r}} \geq \underline{\rho}$. The addition of structural observability ensures that the system meets the conditions of Eq. (5-4) on page 67.

Again, the inclusion of mode-constraints represented by C_{x_b} may increase the lower bound $\underline{\rho}$ to the set of growth rates for which the system is stabilisable to $\underline{\rho}(C_{x_b})$. Furthermore, the reader is referred to Section 4-3 on page 53 for details on growth rate approximations and their validity.

5-2-3 Stabilisability of a mode-constrained stochastic system under hybrid control

Lastly, for Problem III, we explore the conditions for stabilisability of a stochastic SMPL system under hybrid control subject to a reference signal of the form in Eq. (5-5) on page 68 with a growth rate ρ_{r} smaller than the maximum growth rate $\overline{\rho}$.

Since Theorem 5.3 presumes the case where the mode sequence is deterministically controlled via v, it does not apply to systems with uncertainty in their switching behaviour. Due to this uncertainty, in general, no upper limit other than $\overline{\rho}$ can be put on the growth rate of such systems. We can state that, *on average*, the growth rate of systems will converge for infinite horizon to $\overline{\overline{\rho}}(\tilde{v})$ as a function of a specific control sequence $\tilde{v}$ and without continuous control. Additionally, these systems will, *on average*, not be able to achieve growth rates smaller than $\underline{\underline{\rho}}$ for infinite horizon.

Conversely, if we assume that, given a discrete control sequence $\tilde{v}$ and no continuous control, a system's growth rate will always converge to $\overline{\overline{\rho}}(\tilde{v})$, and subsequently can always converge to $\underline{\underline{\rho}}$ with suitable discrete control, we can better predict its behaviour. We will assume such a *stochastically predictable* system in the following hypothesis.

For illustration, we give the following example of a system that does not follow such behaviour:

Example 5.1. Consider a one-dimensional autonomous trimodal stochastic SMPL system with the following system matrices:

$$A^{(1)} = \begin{bmatrix} 1 \end{bmatrix},$$
$$A^{(2)} = \begin{bmatrix} 2 \end{bmatrix},$$
$$A^{(3)} = \begin{bmatrix} 3 \end{bmatrix},$$
$$C = \begin{bmatrix} 0 \end{bmatrix},$$

and with Markovian switching probabilities:

$$P\left[L(k) = j \mid \ell(k-1) = j\right] = 1, \quad \text{for } j \in \{1, 3\}$$
$$P\left[L(k) = j \mid \ell(k-1) = 2\right] = 0.5, \quad \text{for } j \in \{1, 3\}.$$

Evidently, with $\ell(0) = 2$, the system will either stay in mode 1 or mode 3, indefinitely. Mode 1 and 3 are considered *sinks* in an underlying graph representing the modes of the switching system, while mode 2 is considered a *source*. As a result of this behaviour, the system will have a growth rate of 1 or 3 at every individual simulation instance while having an expected growth rate $\overline{\overline{\rho}}$ of 2. We classify such a system as *stochastically unpredictable*. This is in contrast to a stochastically predictable system, whose growth rate will converge to $\overline{\overline{\rho}}$ for all simulation instances.

Hypothesis 5.3. Any stochastically predictable, max-plus Lipschitz stable, structurally observable and structurally controllable SMPL system can be stabilised by hybrid control for *all* $\rho_{\mathrm{r}} \geq \underline{\underline{\rho}}$.

The justification of this hypothesis is similar to Hypothesis 5.2. The only difference is that the lower bound considers the stochastics in the switching behaviour. The inclusion of mode-constrained represented by C_{x_b} may again increase this bound to $\underline{\underline{\rho}}(C_{x_b})$. Section 4-3 on page 53 shows approximations of the expected growth rate metrics and their interpretations.

5-3 A Stabilising Model Predictive Controller

This section discusses the application of model predictive control (MPC) to mode-constrained SMPL systems. De Schutter and Van den Boom pioneered an MPC approach for MPL systems in 2000 [26]. They showed in 2002 that in many cases, MPL-MPC results in a linear programming problem, which can be solved efficiently. Later, they extended the algorithm to deterministic and stochastic SMPL systems [40, 41, 42]. In this report, we apply their findings to mode-constrained systems subject to hybrid control.

Section 5-3-1 first poses a general introduction to SMPL-MPC. Then, in Section 5-3-2 on page 77, Section 5-3-3 on page 80 and Section 5-3-4 on page 83, respectively, the following three cases are discussed:

Case 1: Discrete control of deterministic mode-constrained SMPL systems

Case 2: Hybrid control of deterministic mode-constrained SMPL systems

Case 3: Hybrid control of stochastic mode-constrained SMPL systems

These three cases correspond to the ones discussed in Section 5-2 on page 68. Afterwards, Chapter 6 on page 87 applies this section's theory to an exemplary SMPL system. The reader is referred to Section 2-2 on page 13 for a general introduction to MPC.

5-3-1 Model predictive control for general SMPL systems

The theory in this section is primarily based on [4] and other work of the same authors, unless stated otherwise.

Consider a system in the discrete hybrid stochastic automaton (DHSA) framework, where the switched affine system (SAS) and the mode selector (MS) are represented by a type-1 SMPL system of the form in Eq. (3-1) on page 28:

$$\begin{aligned} x(k) &= A^{(\ell(k))} \otimes x(k-1) \oplus B^{(\ell(k))} \otimes u(k) \\ y(k) &= C^{(\ell(k))} \otimes x(k) \\ \ell(k) &= \phi(\ell(k-1),\, x(k-1),\, v(k),\, u(k)) \\ \ell(k) &\in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N} \end{aligned}$$

where the switching function $\phi(\cdot)$ is unspecified other than its possible dependence on $\ell(k-1)$, $x(k-1)$, $v(k)$ and $u(k)$. The reader is referred to Section 3-3 on page 34 for the theory on DHSA modelling.

Furthermore, consider mode-constraints in the form of admissible discrete control inputs $\tilde{v}$ captured by a constraint matrix C_{x_b}. These arise from the unspecified underlying stochastic finite state machine (sFSM) in the DHSA framework.

Now, define the prediction vectors at event step k over horizon N_p:

$$\tilde{y}(k) \stackrel{\text{def}}{=} \begin{bmatrix} \hat{y}(k \mid k) \\ \vdots \\ \hat{y}(k+N_\mathrm{p}-2 \mid k) \\ \hat{y}(k+N_\mathrm{p}-1 \mid k) \end{bmatrix}, \quad \tilde{u}(k) \stackrel{\text{def}}{=} \begin{bmatrix} u(k) \\ \vdots \\ u(k+N_\mathrm{p}-2) \\ u(k+N_\mathrm{p}-1) \end{bmatrix},$$

$$\tilde{\ell}(k) \stackrel{\text{def}}{=} \begin{bmatrix} \ell(k \mid k) \\ \vdots \\ \ell(k+N_\mathrm{p}-2 \mid k) \\ \ell(k+N_\mathrm{p}-1 \mid k) \end{bmatrix}, \quad \tilde{v}(k) \stackrel{\text{def}}{=} \begin{bmatrix} v(k) \\ \vdots \\ v(k+N_\mathrm{p}-2) \\ v(k+N_\mathrm{p}-1) \end{bmatrix}, \tag{5-7}$$

$$\tilde{r}(k) \stackrel{\text{def}}{=} \begin{bmatrix} r(k) \\ \vdots \\ r(k+N_\mathrm{p}-2) \\ r(k+N_\mathrm{p}-1) \end{bmatrix}$$

where $\hat{y}(k+j \mid k)$ denotes the prediction of $y(k+j)$ based on knowledge at event step k, $u(k+j)$ denotes the continuous input at event step $k+j$, $\ell(k+j)$ represents the mode at event step $k+j$, $v(k+j)$ is the discrete input at event step $k+j$ and $r(k+j)$ denotes the reference at event step $k+j$. This reference signal is assumed to have the form of Eq. (5-5) on page 68:

$$r(k) = \rho_\mathrm{r} k + c(k), \quad \text{where } |c(k)| \leq c_{\max} \text{ and } \rho_\mathrm{r} > 0$$

where ρ_r is the desired growth rate and $c(k) > c(k-1) - \rho_\mathrm{r}$.

In conjunction to the aforementioned prediction vectors, define the following prediction matrices:

$$\begin{aligned} \tilde{A}_m(\tilde{\ell}(k)) &\stackrel{\text{def}}{=} A^{(\ell(k+m-1|k)} \otimes \cdots \otimes A^{(\ell(k|k)} \\ \tilde{B}_{m,n}(\tilde{\ell}(k)) &\stackrel{\text{def}}{=} \begin{cases} A^{(\ell(k+m-1|k))} \otimes \cdots \otimes A^{(\ell(k+n|k))} \otimes B^{(\ell(k+n-1|k))} & \text{if } m > n \\ B^{(\ell(k+m-1|k))} & \text{if } m = n \\ \varepsilon & \text{if } m < n \end{cases} \end{aligned} \tag{5-8}$$

and

$$\begin{aligned} \tilde{C}_m(\tilde{\ell}(k)) &\stackrel{\text{def}}{=} C^{(\ell(k+m-1|k))} \otimes \tilde{A}_m(\tilde{\ell}(k)) \\ \tilde{D}_{m,n}(\tilde{\ell}(k)) &\stackrel{\text{def}}{=} C^{(\ell(k+m-1|k))} \otimes \tilde{B}_{m,n}(\tilde{\ell}(k)), \end{aligned} \tag{5-9}$$

such that the prediction model over horizon $j \in \{0, 1, \ldots, N_\mathrm{p}\}$ is given by

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

in which

$$\begin{aligned}
\bar{B}_j(\tilde{\ell}(k)) &\stackrel{\text{def}}{=} \begin{bmatrix} \tilde{B}_{j,1}(\tilde{\ell}(k)) & \dots & \tilde{B}_{j,N_\text{p}}(\tilde{\ell}(k)) \end{bmatrix}, \\
\tilde{C}(\tilde{\ell}(k)) &\stackrel{\text{def}}{=} \begin{bmatrix} \tilde{C}_1(\tilde{\ell}(k)) \\ \vdots \\ \tilde{C}_{N_\text{p}}(\tilde{\ell}(k)) \end{bmatrix}, \\
\tilde{D}(\tilde{\ell}(k)) &\stackrel{\text{def}}{=} \begin{bmatrix} \tilde{D}_{1,1}(\tilde{\ell}(k)) & \cdots & \tilde{D}_{1,N_\text{p}}(\tilde{\ell}(k)) \\ \vdots & \ddots & \vdots \\ \tilde{D}_{N_\text{p},1}(\tilde{\ell}(k)) & \cdots & \tilde{D}_{N_\text{p},N_\text{p}}(\tilde{\ell}(k)) \end{bmatrix}.
\end{aligned} \tag{5-10}$$

This prediction model does not yet consider any mode constraints or constraints on the set of possible discrete control inputs. Instead, it offers a concise way of predicting a system's state at a future point.

Incorporate now the switching uncertainty in the MS by defining the probability for all switching sequences $\tilde{\ell}(k)$ as:

$$\begin{aligned}
&\tilde{P}\left[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1),\, x(k-1),\, \tilde{u}(k),\, \tilde{v}(k)\right] \\
&= P\left[L(k) = \ell(k) \mid \ell(k-1),\, x(k-1),\, u(k),\, v(k)\right] \\
&\quad \cdot P\left[L(k+1) = \ell(k+1) \mid \ell(k),\, x(k),\, u(k+1),\, v(k+1)\right] \cdot \ldots \cdot \\
&\quad \cdot \ldots \cdot P\left[L(k+N_\text{p}-1) = \ell(k+N_\text{p}-1) \mid \right. \\
&\qquad \left. \ell(k+N_\text{p}-2),\, x(k+N_\text{p}-2),\, u(k+N_\text{p}-1),\, v(k+N_\text{p}-1)\right]
\end{aligned}$$

Recall that the cost function $J(k)$ of an MPC setup at event step k usually consists of the combination of an output cost function $J_\text{out}(k)$ and a scaled input cost function $J_\text{in}(k)$:

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

where $\beta \geq 0$ is a user-chosen tuning parameter. The output cost function represents the system's tardiness by comparing its output to the reference value:

$$\begin{aligned}
J_\text{out}(k) &= \mathbb{E}\left\{ \sum_{j=0}^{N_\text{p}-1} \sum_{i=1}^{n_y} \max\Big(y_i(k+j) - r_i(k+j),\, 0\Big) \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{n_y N_\text{p}} \max\Big(\tilde{y}_i(k) - \tilde{r}_i(k),\, 0\Big) \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{n_y N_\text{p}} \Big[(\tilde{y}(k) - \tilde{r}(k)) \oplus \mathbb{1}\Big]_i \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{n_y N_\text{p}} \Big[\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) - \tilde{r}(k)\Big) \oplus \mathbb{1}\Big]_i \right\} \\
&= \sum_{\tilde{\ell}(k) \in \mathcal{L}_N} \left\{ \sum_{i=1}^{n_y N_\text{p}} \Big[\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) - \tilde{r}(k)\Big) \oplus \mathbb{1}\Big]_i \right. \\
&\qquad \left. \cdot\, \tilde{P}[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1),\, x(k-1),\, \tilde{u}(k),\, \tilde{v}(k)] \right\}
\end{aligned}$$

where $\mathbb{E}\{\cdot\}$ is a function that calculates the expectation over the switching sequences and $\mathbb{1}$ is a vector of zeros of appropriate size. The input cost function gives preference to delays through $u(k)$ while penalising discrete control action $v(k)$:

$$J_{\text{in}}(k) = -\sum_{i=1}^{n_u N_\text{p}} [\tilde{u}(k)]_i + \sum_{i=1}^{n_v N_\text{p}} \tilde{\alpha}_i [\tilde{v}(k)]_i \tag{5-11}$$

where $\tilde{\alpha}$ is a vector containing weights to scale the discrete control action.

The resulting MPC problem for structurally controllable and structurally observable SMPL systems seeks to minimise the cost function $J(k)$ over the control sequences $\tilde{u}(k)$ and $\tilde{v}(k)$ within the following constraints:

$$\tilde{y}(k) = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \tag{5-12a}$$
$$\Delta u(k+j) \geq 0, \quad j = 0, \ldots, N_\text{p} - 1 \tag{5-12b}$$
$$|u_i(k+j) - \rho_\text{r} \cdot (k+j)| \leq \nu_{\max}, \quad \text{for } i = 1, \ldots, n_u, \quad j = 0, \ldots, N_\text{p} - 1 \tag{5-12c}$$
$$Q\tilde{v}(k) \leq q \tag{5-12d}$$
$$R\tilde{u}(k) \leq s \tag{5-12e}$$

where Eq. (5-12a) represents the system dynamics, Eq. (5-12b) enforces the continuous input signal to be nondecreasing, Eq. (5-12c) guarantees stability according to Theorem 5.1 on page 68, and Eq. (5-12d) and Eq. (5-12e) represent additional linear constraints on the control sequences. Note that the resulting optimisation problem is a nonlinear one that can be solved by employing, for instance, multi-start optimisation methods.

Alternatively, Van den Boom and De Schutter show that in many cases, the problem can be rewritten into a more efficient one [4]. For deterministic switching, the original type-1 SMPL description can be converted into a type-2 one where the stochastic signal $d(k)$ is absent, allowing for transforming the system into a deterministic piecewise-affine (PWA) system or deterministic max-min-plus-scaling (MMPS) system. See Section 3-2-2 on page 31 for information regarding type-1 and type-2 SMPL systems. There are many MPC algorithms available for PWA systems, e.g., [61, 62], and for MMPS systems, e.g., [63].

If the MS only depends on the discrete control variable $v(k)$, the MPC problem can be recast into a mixed-integer linear programming (MILP) problem. MSs that describe mode-dependent and control-independent switching can be translated to linear programming (LP) problems, as shown in the following theorem by Van den Boom and De Schutter:

Theorem 5.4. [4, Theorem 2] Consider a type-1 SMPL system with mode-dependent stochastic switching, so the switching probability P_s and switching sequence probability $\tilde{P}_\text{s}$ are such that:

$$P\left[L(k) = \ell(k) \mid \ell(k-1),\, x(k-1),\, u(k),\, v(k)\right] = P_\text{s}\left[L(k) = \ell(k) \mid \ell(k-1)\right]$$
$$\tilde{P}\left[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1),\, x(k-1),\, u(k),\, v(k)\right] = \tilde{P}_\text{s}\left[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1)\right].$$

Assume that $\mathcal{L}_{N_\text{p}}$ can be rewritten as $\mathcal{L}_{N_\text{p}} = \left\{\tilde{\ell}^1, \tilde{\ell}^2, \ldots, \tilde{\ell}^M\right\}$ for $M = n_L^{N_\text{p}}$. The MPC

problem can be recast as an LP problem:

$$\begin{aligned}
\min_{\tilde{u}(k),\, t_{i,m}} \quad & \sum_{i=1}^{n_y N_\mathrm{p}} \sum_{m=1}^{M} t_{i,m} \tilde{P}_\mathrm{s} \left[\tilde{L}(k) = \tilde{\ell}^m \mid \ell(k-1) \right] - \beta \sum_{i=1}^{n_u N_\mathrm{p}} \tilde{u}_i(k) \\
\text{s.t.} \quad & t_{i,m} \geq \left[\tilde{C} \left(\tilde{\ell}^m \right) \right]_{i,l} + x_l(k-1) - \tilde{r}_i(k), \quad \forall i,\, m,\, l \\
& t_{i,m} \geq \left[\tilde{D} \left(\tilde{\ell}^m \right) \right]_{i,l} + \tilde{u}_l(k) - \tilde{r}_i(k), \quad \forall i,\, m,\, l \\
& t_{i,m} \geq 0, \quad \forall i,\, m \\
& \Delta u(k+j) \geq 0, \quad \forall i,\, j \\
& |u_i(k+j) - \rho_\mathrm{r} \cdot (k+j)| \leq \nu_{\max}, \quad \forall i,\, j \\
& R\tilde{u}(k) \leq s.
\end{aligned}$$

The mode constraints as described by the sFSM in the DHSA can be accounted for in the set $\mathcal{L}_{N_\mathrm{p}}$, leading to $M \leq n_L^{N_\mathrm{p}}$.

Whenever the switching probability $P\left[L(k) = \ell(k)\right]$ depends solely and linearly on a control variable such as $v(k)$, the cost function and resulting optimisation problem of the form depicted above become quadratic:

$$\sum_{i=1}^{n_y N_\mathrm{p}} \sum_{m=1}^{M} \sum_{j=1}^{V} t_{i,m} \tilde{P}_\mathrm{s} \left[\tilde{L}(k) = \tilde{\ell}^m \mid \tilde{v}^j \right] \delta_{\tilde{v}^j} \tag{5-14}$$

where V is the number of possible control sequences $\tilde{v}^j$ over horizon N_p and $\delta_{\tilde{v}^j}$ is one when control sequence $\tilde{v}^j$ is activated and zero otherwise.

If, however, we define

$$M_t \geq \max t_{i,m}, \quad m_t \leq \min t_{i,m},$$

we can introduce auxiliary real variables $t_{i,m}^j = \delta_{\tilde{v}^j} t_{i,m}$ to linearise the cost function. To this end, we replace the cost function in Eq. (5-14) with:

$$\sum_{i=1}^{n_y N_\mathrm{p}} \sum_{m=1}^{M} \sum_{j=1}^{V} t_{i,m}^j \tilde{P}_\mathrm{s} \left[\tilde{L}(k) = \tilde{\ell}^m \mid \tilde{v}^j \right]$$

that is subject to the constraints:

$$\begin{aligned}
& t_{i,m}^j \leq M_t \delta_{\tilde{v}^j} \\
& t_{i,m}^j \geq m_t \delta_{\tilde{v}^j} \\
& t_{i,m}^j \leq t_{i,m} - m_t \left(1 - \delta_{\tilde{v}^j}\right) \\
& t_{i,m}^j \geq t_{i,m} - M_t \left(1 - \delta_{\tilde{v}^j}\right)
\end{aligned}$$

After this transformation, the mixed-integer quadratic programming (MIQP) problem is converted into an MILP one. The bounds M_t and m_t that transform the system into a mixed logical dynamical (MLD)-like description are an arbitrary upper and lower bound to the actual maximum of the variables, with stricter values providing computational benefits [64, p. 171]. For our application, we consider the following bounds:

$$M_t = \overline{\rho} \cdot N_\mathrm{sim} + y(0), \quad m_t = y(0).$$

Since the variables $t_{i,m}$ represent the system's output, they are upper bounded by a function of the maximum growth rate $\overline{\rho}$ and simulation horizon N_{sim}, and lower bounded by 0. These bounds may be offset by any nonzero initial system output $y(0)$.

In summary, any switching function that depends on the discrete control signal $v(k)$, combined with continuous control in the optimisation algorithm, can result in an MILP problem. Individual assessment of the algorithm's efficiency may still result in a preference towards MIQP. This selection is discussed in Chapter 6 on page 87 for the individual case studies.

Timing issues The irregularity of MPC for SMPL systems is that the event counter k does not directly relate to a specific time. Instead, it represents the number of events that have happened. As a result, the state vector $x(k)$ is often not fully known at the same time instant. Van den Boom and De Schutter address these timing issues in the following manner [60, Sec. 3]:

For the case of full state information[1], let $[x_{\text{est}}(k,\, t)]_i$ be an estimation of the kth occurrence time of an internal event i with a measured, true occurrence time of $[x_{\text{true}}(k)]_i$. Let $l(t)$ be the smallest integer such that $[x_{\text{true}}(k-l(t))]_i$ is known at time t. Then, define $x_{\text{est}}(k-l(t),\, t) = x_{\text{true}}(k-l(t))$ and reconstruct the unknown state components using the recursion

$$x_{\text{est}}(k-l(t)+j,\, t) = A \otimes x_{\text{est}}(k-l(t)+j-1,\, t) \oplus B \otimes \tilde{u}(k-l(t)+j-1,\, t) \quad \text{for } j = 1,\, \ldots,\, l(t).$$

Note that $\tilde{u}(k-l(t)+j-1,\, t)$ consists of past applied input times and future computed input times.

The second time-related issue for MPL-MPC is to find the best time $t = t(k)$ to determine $x(k)|_t$ and to start the optimisation algorithm for computing $\tilde{u}(k)$. Here, $x(k)|_t$ is the value of the state $x(k)$ that can be used at time t:

$$[x(k)|_t]_i = \begin{cases} [x_{\text{true}}(k)]_i & \text{if } [x_{\text{true}}(k)]_i \text{ is known at time } t \text{ (so } [x_{\text{true}}(k)]_i \leq t) \\ [x_{\text{ext}}(k,t)]_i & \text{otherwise} \end{cases}$$

The previously computed sequence $\tilde{u}(k-1)$ offers insight into the appropriate time instant to start the computation. Namely,

$$t(k) = \min_j \left[u(k \mid k-1)\right]_j - t_{\text{c}},$$

where t_{c} is an upper bound for the computation time. If $t_{\text{c}} \ll \Delta t(k+1)$, at least one additional intermediate optimisation step can be done with new information.

In this report, we do not consider these timing issues, and instead assume full state availability without measurement delay. The following sections in this chapter detail the specifics of MPC for the three control cases introduced earlier.

5-3-2 Discrete control of a mode-constrained deterministic SMPL system

This section and the succeeding two discuss the differences of the system model, prediction model, cost function, constraints, and optimisation problem with respect to the general MPC model introduced in Section 5-3-1 on page 72. The subject of this section is discrete control of a deterministic SMPL system.

[1] Since the components of the state correspond to event times, they are, in general, easy to measure.

System model

Similarly, the structurally observable deterministic SMPL system under discrete control is represented as a discrete hybrid automaton (DHA). The single-output SAS has the following form:

$$x(k) = A^{(\ell(k))} \otimes x(k-1)$$
$$y(k) = C \otimes x(k)$$

and is paired with a deterministic MS:

$$\ell(k) = \phi(\ell(k-1),\, v(k))$$
$$\ell(k) \in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N}$$

with no switching stochastics:

$$P\left[L(k) = \ell(k) \mid \ell(k-1),\, v(k)\right] \in \{0,\, 1\}$$

System matrix $A^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{n_x \times n_x}$ varies over the modes $\ell(k)$, whereas matrix $C \in \mathbb{R}_{\varepsilon}^{1 \times n_x}$ is assumed to be constant across the modes.

The discrete state x_b of the finite state machine (FSM) evolves according to the dynamics specified by the unspecified function $f_{\mathrm{FSM}}(\cdot)$:

$$x_b(k+1) = f_{\mathrm{FSM}}(x_b(k),\, \ell(k))$$

Lastly, we assume a reference signal of the form in Eq. (5-5) on page 68.

Prediction model

The prediction vectors $\tilde{\ell}(k)$, $\tilde{r}(k)$, $\tilde{y}(k)$ and $\tilde{v}(k)$ over horizon N_{p} are defined as in Eq. (5-7) on page 73. Similarly, the matrices $\tilde{A}_m(\tilde{\ell}(k))$, $\tilde{C}_m(\tilde{\ell}(k))$ and $\tilde{C}(\tilde{\ell}(k))$ are defined as in Eq. (5-8), Eq. (5-9) and Eq. (5-10) on page 73. The prediction model as a function of the switching sequence $\tilde{\ell}(k)$ is given by:

$$x(k+j) = \tilde{A}_j(\tilde{\ell}(k)) \otimes x(k-1)$$
$$\tilde{y}(k) = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1)$$

Furthermore, we can predict the future FSM state using the unspecified function $\tilde{f}_{\mathrm{FSM}}(\cdot)$ that acts as an expansion to the function $f_{\mathrm{FSM}}(\cdot)$:

$$x_b(k+j) = \tilde{f}_{\mathrm{FSM}}(x_b(k),\, \tilde{\ell}(k))$$

Cost function

With the omission of continuous control and no penalisation on discrete control, the cost function is solely dependent on the system's output. As a result, the optimisation problem loses the inclination to minimise the positive distance between the reference signal $r(k)$ and

the system's output $y(k)$, a property we aim to keep. To that end, we consider both the positive difference $y(k) - r(k)$ and the negative difference $r(k) - y(k)$ in a weighted manner:

$$\begin{aligned}
J(k) &= \mathbb{E}\left\{ \sum_{j=0}^{N_\mathrm{p}-1} \max\Big(y(k+j) - r(k+j),\, \gamma(r(k+j) - y(k+j))\Big) \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{N_\mathrm{p}} \max\Big(\tilde{y}_i(k) - \tilde{r}_i(k),\, \gamma(\tilde{r}_i(k) - \tilde{y}_i(k))\Big) \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{N_\mathrm{p}} \Big[(\tilde{y}(k) - \tilde{r}(k)) \oplus \gamma(\tilde{r}(k) - \tilde{y}(k)) \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{N_\mathrm{p}} \Big[\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) - \tilde{r}(k)\Big) \oplus \gamma\Big(\tilde{r}(k) - \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1)\Big) \right\} \\
&= \sum_{\tilde{\ell}(k) \in \mathcal{L}_N} \left\{ \sum_{i=1}^{N_\mathrm{p}} \Big[\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) - \tilde{r}(k)\Big) \oplus \gamma\Big(\tilde{r}(k) - \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1)\Big) \right. \\
&\qquad \left. \cdot\, \tilde{P}[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1),\, \tilde{v}(k)] \right\}.
\end{aligned} \tag{5-15}$$

The nonnegative parameter $\gamma \geq 0$ balances the output's convergence speed with respect to the reference and the strictness of the soft constraint that keeps the output below the reference. The influence of the weight for the case study is evaluated in Chapter 6-2-2 on page 97.

Constraints

Many constraints present in the general case in Eq. (5-12) on page 75 can be discarded due to the omission of continuous control. We keep the following constraints:

$$\tilde{y}(k) = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \tag{5-16a}$$
$$Q\tilde{v}(k) \leq q \tag{5-16b}$$

The constraint matrix Q and vector q embody the mode constraints as represented by constraints on the discrete control sequence $\tilde{v}(k)$, such that for every k, $v(k) \in C_{x_b}^{\top} x_b$ as discussed in Section 3-3 on page 34. The specific implementation of these constraints for the case study is discussed in Chapter 6-2-3 on page 98.

Optimisation problem

The individual control components discussed in this section are rewritten in the following LP problem:

$$\min_{\tilde{v},\, t_{i,m}} \quad \sum_{i=1}^{N_\mathrm{p}} \sum_{m=1}^{n_L} t_{i,m} \tilde{P}_\mathrm{s} \left[\tilde{L}(k) = \tilde{\ell}^m \mid \tilde{v}(k) \right] \tag{5-17a}$$

$$\text{s.t.} \quad t_{i,m} \geq \left[\tilde{C} \left(\tilde{\ell}^m \right) \right]_{i,l} + x_l(k-1) - \tilde{r}_i(k), \quad \forall\, i,\, m,\, l \tag{5-17b}$$

$$t_{i,m} \geq \gamma \left(\tilde{r}_i(k) - \left[\tilde{C} \left(\tilde{\ell}^m \right) \right]_{i,l} - x_l(k-1) \right), \quad \forall\, i,\, m,\, l \tag{5-17c}$$

$$t_{i,m} \geq 0, \quad \forall\, i,\, m \tag{5-17d}$$

$$Q\tilde{v}(k) \leq q \tag{5-17e}$$

where $\tilde{P}_\mathrm{s}[\cdot]$ is the probability of a specific mode sequence. Notice that the inclusion of Eq. (5-17c) ensures the minimisation of the distance $r(k) - y(k)$, scaled through the weight factor γ. Due to the specific implementation discussed in Section 6-2-5 on page 99, the problem results in an MILP one.

For a structurally observable and max-plus Lipschitz stable system and a reference signal with a growth rate of ρ_r in the set $[\underline{\rho},\, \overline{\rho}]$, the absolute difference $|y(k) - r(k)|$ will remain bounded according to Hypothesis 5.1 on page 69, thereby ensuring stability according to the stability conditions of Eq. (5-4) on page 67.

5-3-3 Hybrid control of a mode-constrained deterministic SMPL system

This section details the specifics of the hybrid control problem of a deterministic SMPL system. It is subdivided into the following parts: system model, prediction model, cost function, constraints and optimisation problem. The implementation of the theory is discussed in Section 6-3 on page 101.

System model

The structurally controllable and observable single-input single-output (SISO) SAS of the DHA under hybrid control is described by:

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

and is paired with a deterministic MS:

$$\begin{aligned} \ell(k) &= \phi(\ell(k-1),\, v(k)) \\ \ell(k) &\in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N} \end{aligned}$$

with no switching stochastics:

$$P\left[L(k) = \ell(k) \mid \ell(k-1),\, v(k) \right] \in \{0,\, 1\}$$

System matrices $A^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{n_x \times n_x}$ and $B^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{n_x \times 1}$ vary over the modes $\ell(k)$, whereas matrix $C \in \mathbb{R}_{\varepsilon}^{1 \times n_x}$ is assumed to be constant across the modes.

The discrete state x_b of the FSM evolves according to the dynamics specified by the function $f_{\text{FSM}}(\cdot)$:

$$x_b(k+1) = f_{\text{FSM}}(x_b(k),\, \ell(k))$$

Lastly, we assume a reference signal of the form in Eq. (5-5) on page 68.

Prediction model

The prediction vectors $\tilde{\ell}(k)$, $\tilde{r}(k)$, $\tilde{y}(k)$, $\tilde{u}(k)$ and $\tilde{v}(k)$ over horizon N_{p} are defined as in Eq. (5-7) on page 73. Similarly, the matrices $\tilde{A}_m(\tilde{\ell}(k))$, $\bar{B}_j(\tilde{\ell}(k))$, $\tilde{B}_{m,n}(\tilde{\ell}(k))$, $\tilde{C}(\tilde{\ell}(k))$, $\tilde{C}_m(\tilde{\ell}(k))$, $\tilde{D}(\tilde{\ell}(k))$ and $\tilde{D}_{m,n}(\tilde{\ell}(k))$ are defined in Eq. (5-8), Eq. (5-9) and Eq. (5-10) on page 73. The prediction model of the deterministic model under hybrid control given a switching sequence $\tilde{\ell}(k)$ is:

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

Furthermore, we can predict the future FSM state:

$$x_b(k+j) = \tilde{f}_{\text{FSM}}(x_b(k),\, \tilde{\ell}(k))$$

Cost function

The cost function is the sum of the output cost function and the scaled input cost function:

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

where,

$$\begin{aligned} J_{\text{out}}(k) &= \mathbb{E}\left\{ \sum_{j=0}^{N_{\text{p}}-1} \max\Big(y(k+j) - r(k+j), 0\Big) \right\} \\ &= \mathbb{E}\left\{ \sum_{i=1}^{N_{\text{p}}} \max\Big(\tilde{y}_i(k) - \tilde{r}_i(k), 0\Big) \right\} \\ &= \mathbb{E}\left\{ \sum_{i=1}^{N_{\text{p}}} \Big[(\tilde{y}(k) - \tilde{r}(k)) \oplus \mathbb{1} \Big]_i \right\} \\ &= \mathbb{E}\left\{ \sum_{i=1}^{N_{\text{p}}} \Big[\Big(\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \Big) - \tilde{r}(k) \Big) \oplus \mathbb{1} \Big]_i \right\} \\ &= \sum_{\tilde{\ell}(k) \in \mathcal{L}_N} \left\{ \sum_{i=1}^{N_{\text{p}}} \Big[\Big(\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \Big) - \tilde{r}(k) \Big) \oplus \mathbb{1} \Big]_i \right. \\ &\qquad \left. \cdot\, \tilde{P}[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1),\, \tilde{v}(k)] \right\} \end{aligned}$$

and

$$J_{\text{in}}(k) = -\sum_{i=1}^{N_{\text{p}}} [\tilde{u}(k)]_i$$

As opposed to the input cost function in Eq. (5-11) on page 75, there is no penalisation on the discrete control action $\tilde{v}(k)$, since we assign no value to specific modes. One could opt to enforce soft constraints on mode sequences via this cost function instead of hard constraints in the optimisation problem.

Constraints

We have the following constraints:

$$\tilde{y}(k) = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \tag{5-18a}$$
$$\Delta u(k+j) \geq 0, \quad j = 0, \ldots, N_{\text{p}} - 1 \tag{5-18b}$$
$$|u(k+j) - \rho_{\text{r}} \cdot (k+j)| \leq \nu_{\max}, \quad \text{for } j = 0, \ldots, N_{\text{p}} - 1 \tag{5-18c}$$
$$Q\tilde{v}(k) \leq q \tag{5-18d}$$
$$R\tilde{u}(k) \leq s \tag{5-18e}$$

Eq. (5-18a) represents the system dynamics, Eq. (5-18b) and Eq. (5-18c) dictate the behaviour of the continuous input sequence, and Eq. (5-18d) and Eq. (5-18e) are additional linear constraints on the input sequences. The mode constraints can be captured in Eq. (5-18d).

Optimisation problem

This again results in an LP optimisation problem:

$$\begin{aligned}
\min_{\tilde{u},\,\tilde{v}} \quad & \sum_{i=1}^{N_{\text{p}}} \sum_{m=1}^{n_L} t_{i,m} \tilde{P}_{\text{s}} \left[\tilde{L}(k) = \tilde{\ell}^m \mid \tilde{v}(k)\right] - \beta \sum_{i=1}^{N_{\text{p}}} \tilde{u}_i(k) \\
\text{s.t.} \quad & t_{i,m} \geq \left[\tilde{C}\left(\tilde{\ell}^m\right)\right]_{i,l} + x_l(k-1) - \tilde{r}_i(k), \quad \forall i,\, m,\, l \\
& t_{i,m} \geq \left[\tilde{D}\left(\tilde{\ell}^m\right)\right]_{i,l} + \tilde{u}(k) - \tilde{r}(k), \quad \forall m,\, l \\
& t_{i,m} \geq 0, \quad \forall i,\, m \\
& u(k+j) - u(k+j-1) \geq 0, \quad \forall j \\
& |u(k+j) - \rho_{\text{r}} \cdot (k+j)| \leq \nu_{\max}, \quad \forall j \\
& R\tilde{u}(k) \leq s, \\
& Q\tilde{v}(k) \leq q
\end{aligned}$$

The algorithm can be implemented as either an MILP or an MIQP problem, as discussed in Section 5-3-1 on page 72. For a structurally observable, structurally controllable and max-plus Lipschitz stable SMPL system and a reference signal with a growth rate of ρ_{r} no lower than $\underline{\rho}$, the absolute difference $|y(k) - r(k)|$ and positive distance $y(k) - u(k)$ will remain bounded according to Hypothesis 5.2 on page 70, thereby ensuring stability according to the stability conditions of Eq. (5-4) on page 67.

5-3-4 Hybrid control of a mode-constrained stochastic SMPL system

At last, we discuss MPC for a mode-constrained stochastic SMPL system. The section introduces the system model, prediction model, cost function, constraints and optimisation problem. Section 6-4 on page 109 details the implementation of the individual parts in the case study.

System model

We consider a structurally controllable and observable SISO SAS of a DHSA under hybrid control:

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

paired with a stochastic MS:

$$\begin{aligned} \ell(k) &= \phi_{\text{stoch}}(\ell(k-1),\, v(k)) \\ \ell(k) &\in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N} \end{aligned}$$

with switching stochastics:

$$P\left[L(k) = \ell(k) \mid \ell(k-1),\, v(k)\right] \in (0,\, 1)$$

System matrices $A \in \mathbb{R}_{\varepsilon}^{n_x \times n_x}$ and $B \in \mathbb{R}_{\varepsilon}^{n_x \times 1}$ vary over the modes $\ell(k)$, whereas matrix $C \in \mathbb{R}_{\varepsilon}^{1 \times n_x}$ is constant. The number of system modes is n_L.

The discrete state x_b of the sFSM evolves according to the stochastic dynamics specified by the function $f_{\text{sFSM}}(\cdot)$:

$$P\left[x_b(k+1) = \hat{x}_b\right] = f_{\text{sFSM}}(x_b(k),\, v(k))$$

Finally, we consider a reference signal of the form in Eq. (5-5) on page 68.

Prediction model

The prediction vectors $\tilde{\ell}(k)$, $\tilde{r}(k)$, $\tilde{y}(k)$, $\tilde{u}(k)$ and $\tilde{v}(k)$ over horizon N_{p} are defined as in Eq. (5-7) on page 73. Similarly, the matrices $\tilde{A}_m(\tilde{\ell}(k))$, $\bar{B}_j(\tilde{\ell}(k))$, $\tilde{B}_{m,n}(\tilde{\ell}(k))$, $\tilde{C}(\tilde{\ell}(k))$, $\tilde{C}_m(\tilde{\ell}(k))$, $\tilde{D}(\tilde{\ell}(k))$ and $\tilde{D}_{m,n}(\tilde{\ell}(k))$ are defined in Eq. (5-8), Eq. (5-9) and Eq. (5-10) on page 73. The prediction model of the stochastic model under hybrid control given a switching sequence $\tilde{\ell}(k)$ is:

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

Furthermore, we can predict the future sFSM state:

$$x_b(k+j) = \tilde{f}_{\text{sFSM}}(x_b(k),\, \tilde{\ell}(k))$$

Note that these prediction models are deterministic under the given mode sequence $\tilde{\ell}(k)$. The stochastics arise in the switching to mode $\ell(k+1)$.

Cost function

The cost function is again a sum of two parts:

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

where,

$$\begin{aligned}
J_{\text{out}}(k) &= \mathbb{E}\left\{ \sum_{j=0}^{N_{\text{p}}-1} \max\Big(y(k+j) - r(k+j),\, 0\Big) \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{N_{\text{p}}} \max\Big(\tilde{y}_i(k) - \tilde{r}_i(k),\, 0\Big) \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{N_{\text{p}}} \Big[(\tilde{y}(k) - \tilde{r}(k)) \oplus \mathbb{1} \Big]_i \right\} \\
&= \mathbb{E}\left\{ \sum_{i=1}^{N_{\text{p}}} \Big[\Big(\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \Big) - \tilde{r}(k) \Big) \oplus \mathbb{1} \Big]_i \right\} \\
&= \sum_{\tilde{\ell}(k) \in \mathcal{L}_N} \left\{ \sum_{i=1}^{N_{\text{p}}} \Big[\Big(\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \Big) - \tilde{r}(k) \Big) \oplus \mathbb{1} \Big]_i \right. \\
&\qquad \left. \cdot\, \tilde{P}[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1),\, \tilde{v}(k)] \right\}
\end{aligned}$$

and

$$J_{\text{in}}(k) = -\sum_{i=1}^{N_{\text{p}}} [\tilde{u}(k)]_i$$

Recall from Section 5-3-3 on page 80 that there is no penalisation on $\tilde{v}(k)$.

Constraints

We have the following constraints:

$$\tilde{y}(k) = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \tag{5-20a}$$
$$\Delta u(k+j) \geq 0, \quad j = 0, \ldots, N_{\text{p}} - 1 \tag{5-20b}$$
$$|u(k+j) - \rho_{\text{r}} \cdot (k+j)| \leq \nu_{\max}, \quad \text{for } j = 0, \ldots, N_{\text{p}} - 1 \tag{5-20c}$$
$$Q\tilde{v}(k) \leq q \tag{5-20d}$$
$$R\tilde{u}(k) \leq s \tag{5-20e}$$

Eq. (5-20a) represents the system dynamics, Eq. (5-20b) and Eq. (5-20c) dictate the behaviour of the continuous input sequence, and Eq. (5-20d) and Eq. (5-20e) are additional linear constraints on the input sequences.

Optimisation problem

We obtain the following LP optimisation problem:

$$\begin{aligned}
\min_{\tilde{u},\,\tilde{v}} \quad & \sum_{i=1}^{N_\mathrm{p}} \sum_{m=1}^{n_L} t_{i,m} \tilde{P}_\mathrm{s} \left[\tilde{L}(k) = \tilde{\ell}^m \mid \tilde{v}(k) \right] - \beta \sum_{i=1}^{N_\mathrm{p}} \tilde{u}_i(k) \\
\text{s.t.} \quad & t_{i,m} \geq \left[\tilde{C} \left(\tilde{\ell}^m \right) \right]_{i,l} + x_l(k-1) - \tilde{r}_i(k), \quad \forall i,\, m,\, l \\
& t_{i,m} \geq \left[\tilde{D} \left(\tilde{\ell}^m \right) \right]_{i,l} + \tilde{u}(k) - \tilde{r}(k), \quad \forall m,\, l \\
& t_{i,m} \geq 0, \quad \forall i,\, m \\
& u(k+j) - u(k+j-1) \geq 0, \quad \forall j \\
& |u(k+j) - \rho_\mathrm{r} \cdot (k+j)| \leq \nu_{\max}, \quad \forall j \\
& R\tilde{u}(k) \leq s, \\
& Q\tilde{v}(k) \leq q
\end{aligned}$$

The hybrid control description of the stochastic system is very similar to the one for the deterministic system. This similarity results from the uncertainty being represented only in the probability $\tilde{P}_\mathrm{s}$ of a particular mode sequence $\tilde{\ell}(k)$ given a discrete control sequence $\tilde{v}(k)$. The uncertainty is dealt with during the online execution of the MPC and not in the predictor step of the controller. This is discussed in Chapter 6 on page 87.

Again, the algorithm results in an MILP or an MIQP problem as discussed in Section 5-3-1 on page 72. For a structurally observable, structurally controllable, max-plus Lipschitz stable and stochastically predictable SMPL system and a reference signal with a growth rate of ρ_r no smaller than $\underline{\rho}$, the absolute difference $|y(k) - r(k)|$ and positive distance $y(k) - u(k)$ will remain bounded according to Hypothesis 5.3 on page 71, thereby ensuring stability according to the stability conditions of Eq. (5-4) on page 67.

The next chapter discusses the practical implementation of such systems into MATLAB to validate the presented theory.

Chapter 6

Case Study

This penultimate chapter deals with the application of the model predictive control (MPC) method introduced in Section 5-3 on page 72 to suitable switching max-plus linear (SMPL) systems according to the following three familiar cases:

1. Discrete Control of a Mode-Constrained Deterministic SMPL System
2. Hybrid Control of a Mode-Constrained Deterministic SMPL System
3. Hybrid Control of a Mode-Constrained Stochastic SMPL System

These cases are discussed in Section 6-2 on page 96, Section 6-3 on page 101 and Section 6-4 on page 109, respectively. The sections offer details on the specific implementation in MATLAB and the results of the simulations. First, Section 6-1 on page 88 introduces the system description and the control structure that are used throughout the chapter.

6-1 Problem Setup

This section discusses the SMPL system used for visualising the results of this work. In order, it reviews the application of MPC to the three cases as specified on page 87. Case 1 and Case 2 in this study employ the same underlying system as in Case 3. However, switching is instead assumed to be deterministic, and Case 1 does not permit the use of continuous control through $u(k)$. This section introduces the problem setup and the system with uncertain switching under hybrid control as used for Case 3. Then, Section 6-2 on page 96 and Section 6-3 on page 101 explain the specific alterations to this general model for the first two cases.

The sections in this chapter introduce the system description, control objective, system constraints, optimisation problem and controller setup. The latter three sections also review results obtained through a MATLAB simulation.

6-1-1 System description

The mode-constrained SMPL system used for the three case studies is described as a discrete hybrid stochastic automaton (DHSA), introduced in Section 3-3 on page 34. It is modelled as an interconnection between a switched affine system (SAS), a mode selector (MS) and a stochastic finite state machine (sFSM) that are described below. The switching stochasticity has been modelled as a part of the MS instead of the sFSM. Still, we opt to regard it as a DHSA instead of a discrete hybrid automaton (DHA) despite the lack of stochasticity in the sFSM as a function of $\ell(k)$. We do still consider stochasticity in the sFSM as a function of $v(k)$.

SAS The switched affine system (SAS) is a structurally controllable, structurally observable and max-plus Lipschitz stable single-input single-output (SISO) SMPL system with the following description:

$$\begin{aligned} x(k) &= A^{(\ell(k))} \otimes x(k-1) \oplus B^{(\ell(k))} \otimes u(k) \\ y(k) &= C \otimes x(k) \end{aligned} \tag{6-1}$$

where $A^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{n_x \times n_x}$, $B^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{n_x \times 1}$ and $C \in \mathbb{R}_{\varepsilon}^{1 \times n_x}$. We assume the output matrix C to be constant across the modes and have only zeros as its entries:

$$C = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}_{\varepsilon}^{1 \times n_x}$$

MS Secondly, the MS, denoting the switching behaviour, is described as:

$$\begin{aligned} &\ell(k) = \phi_{\text{stoch}}(\ell(k-1),\, v(k)) \\ &\ell(k) \in \mathcal{L} = \{1,\, 2,\, 3\}, \quad k \in \mathbb{N} \\ &P\left[L(k) = \ell(k) \mid \ell(k-1),\, v(k)\right] = \begin{cases} 0.8 & \text{for } \ell(k) = v(k) \\ 0.1 & \text{otherwise} \end{cases} \end{aligned} \tag{6-2}$$

where $v(k) \in \{1,\, 2,\, 3\}$ is an integer. Note that the switching function does not explicitly depend on $\ell(k-1)$. However, this previous mode does affect the set of admissible control signals for $v(k)$. Therefore, the dependence is displayed in the function description.

sFSM Lastly, the sFSM used in these studies enforces the constraint that no mode should occur more than three times in a row. Figure 6-1 depicts the automaton constraining the system assuming deterministic switching. In the graph, discrete state Q_{ij} represents the j-th instance of mode i in a row. The labels along the edges denote the allowed discrete control inputs $v(k)$ and, therefore, the succeeding mode.

In this case of uncertain switching, however, the sFSM is set up in a way that allows for the violation of the constraint. The controller may not decide to violate the constraint through $v(k)$, but through stochastics, this may still happen. Namely, the discrete control signal $v(k)$ can only influence the probability of switching, as shown in Eq. (6-2). Note that a different setup may rule out the violation for all cases.

Figure 6-2 on page 90 is an adaptation to the automaton in Figure 6-1 that takes switching uncertainty into account. Now, the possible discrete control inputs, denoted by the sets a, b, c and d of Eq. (6-3) on page 90, do not relate one-to-one with the state evolution of the automaton. Instead, they depict all possible paths and allowed control signals at each automaton state. Together with the stochasticity information in Eq. (6-2), it reflects the mode evolution behaviour of the SMPL system. These admissible control inputs collected in the sets in Eq. (6-3) on page 90 form the alphabet of the sFSM. The dotted paths in Figure 6-2 on page 90 denote ones that result in the violation of the mode sequence constraint. Note that the discrete state evolution of the sFSM is governed deterministically by the mode $\ell(k)$ and stochastically by the signal $v(k)$.

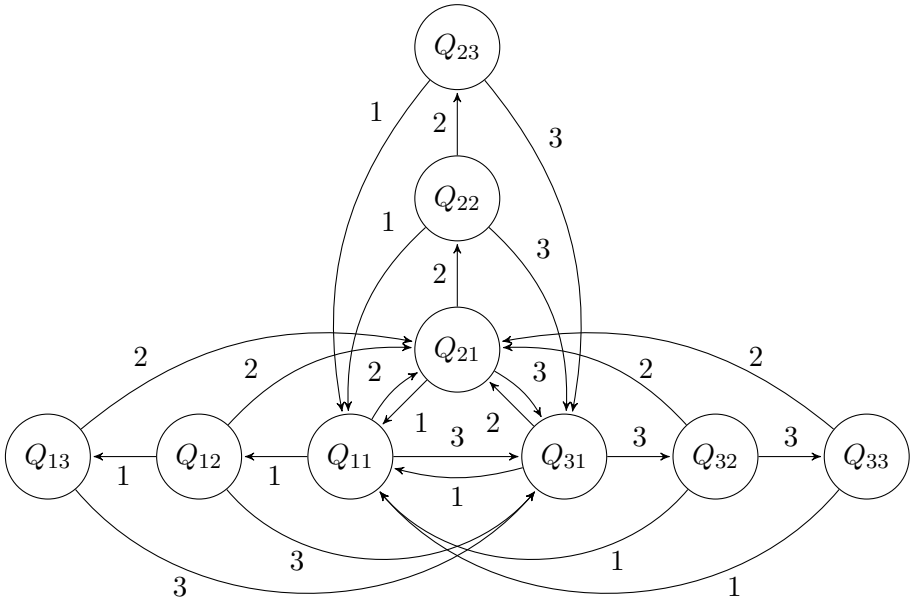

Figure 6-1: Finite state machine (FSM) capturing the constraint enforcing a maximum number of consecutive equal modes. Discrete state Q_{ij} represents the j-th instance of mode i in a row. The labels on the paths denote the admissible control inputs.

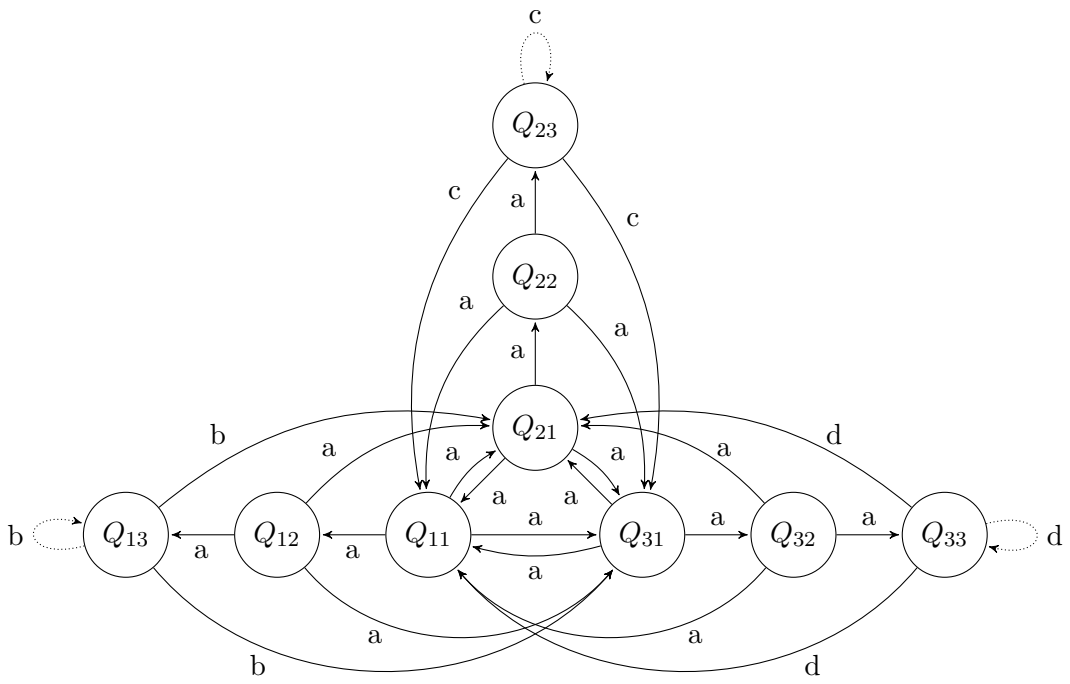

Figure 6-2: Stochastic finite state machine (sFSM) capturing the constraint enforcing a maximum number of consecutive equal modes. Discrete state Q_{ij} represents the j-th instance of mode i in a row. The labels on the paths relate to the admissible control inputs as shown in Eq. (6-3). The dotted paths denote ones that result in the violation of the constraint.

$$\begin{aligned} a &= \{1, 2, 3\} \\ b &= \{2, 3\} \\ c &= \{1, 3\} \\ d &= \{1, 2\} \end{aligned} \tag{6-3}$$

The system used in this chapter is generated using the `generateSystem.m` script in MATLAB that is shown in Appendix A-2-9 on page 164, with the following dimensions:

$$\begin{aligned} n_x &= 3 \quad \text{(Number of states)} \\ n_y &= 1 \quad \text{(Number of outputs)} \\ n_u &= 1 \quad \text{(Number of continuous inputs)} \\ n_v &= 1 \quad \text{(Number of discrete inputs)} \\ n_L &= 3 \quad \text{(Number of modes)} \end{aligned} \tag{6-4}$$

It results in a structurally controllable, structurally observable and max-plus Lipschitz stable system with three irreducible matrices $A^{(\ell)}$ with unique eigenvalues approximately equal to:

$$\begin{aligned} \lambda(A^{(1)}) &\approx 1.91 \\ \lambda(A^{(2)}) &\approx 2.07 \\ \lambda(A^{(3)}) &\approx 2.19 \end{aligned} \tag{6-5}$$

These values have been calculated using a Max-Plus Algebra Toolbox for MATLAB [65] that uses the work by Olsder et al. [66]. Furthermore, the maximum growth rate of the system as calculated using Eq. (4-3) on page 43, and the minimum (expected) growth rate with cyclic control signals of period $N_{\mathrm{p}} = 6$, simulation horizon $N_{\mathrm{sim}} = 96$ and $N_{\mathrm{t}} = 10$ are approximately equal to:

$$\begin{aligned} \overline{\rho} &\approx 2.85 \\ \underline{\underline{\rho}}^{\mathrm{con}}_{N_{\mathrm{p}}} &\approx 2.09 \\ \underline{\rho}^{\mathrm{con}}_{N_{\mathrm{p}}} &\approx 1.94 \end{aligned}$$

Note that these values correspond to the mode-constrained system that does not allow for mode sequences of the same mode larger than 3. For illustration, the unconstrained values are equal to $\underline{\underline{\rho}}_{N_{\mathrm{p}}} = 2.08$ and $\underline{\rho}_{N_{\mathrm{p}}} = 1.91$. Hereafter, we omit the superscript 'con' for brevity while still assuming a constrained system. The reader is referred to Algorithm 4.1 on page 50 and the script `growthRate.m` in Appendix A-2-10 on page 165 for the algorithm to compute these values. The distribution of the growth rate $\overline{\overline{\rho}}_{N_{\mathrm{p}}}$ given all possible periodic mode sequences $\tilde{\ell}_{N_{\mathrm{p}}}$ with a period of 6 over a horizon of 96 is visualised by box plots in Figure 6-3, and the expected growth rate given all possible periodic control sequences $\tilde{v}_{N_{\mathrm{p}}}$ is visualised in Figure 6-4 on page 92.

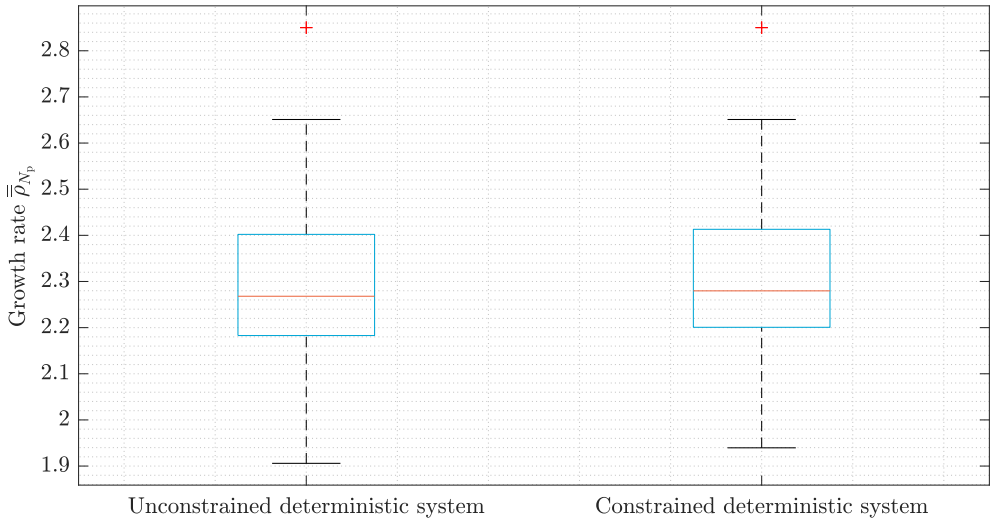

Figure 6-3: Distribution of the growth rate $\overline{\overline{\rho}}_{N_{\mathrm{p}}}$ given all periodic mode sequences $\tilde{\ell}_{N_{\mathrm{p}}}$ with a period of $N_{\mathrm{p}} = 6$ over a horizon of $N_{\mathrm{sim}} = 96$. They have been calculated using Algorithm 4.1 on page 50 implemented in the script `growhtRate.m` of Appendix A-2-10 on page 165.

6-1-2 Objective function

We aim to track a reference signal with a growth rate of $\rho_{\mathrm{r}} = 2.2$ of the form:

$$r(k) = \rho_{\mathrm{r}}\, k + c \tag{6-6}$$

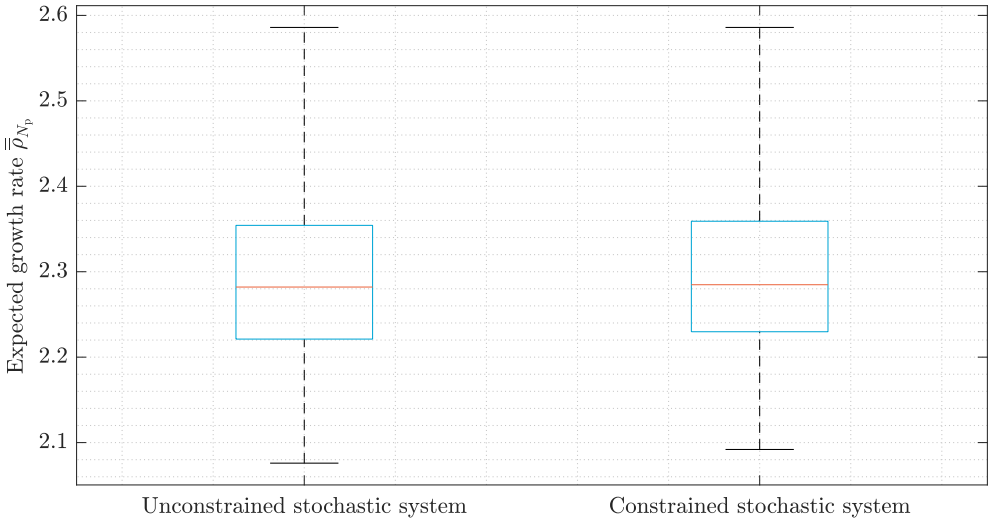

Figure 6-4: Distribution of the expected growth rate $\overline{\overline{\rho}}_{N_\mathrm{p}}$ given all periodic control sequences $\tilde{v}_{N_\mathrm{p}}$ with a period of $N_\mathrm{p} = 6$ over a horizon of $N_\mathrm{sim} = 96$. They have been calculated using Algorithm 4.1 on page 50 implemented in the script `growhtRate.m` of Appendix A-2-10 on page 165.

where $c = -2$ in order to achieve slower convergence in the simulation, such that:

$$\rho_\mathrm{o} \leq \rho_\mathrm{r}$$

and the distance $|\rho_\mathrm{o} - \rho_\mathrm{r}|$ is bounded and minimised.

Furthermore, we want to achieve a system output growth rate ρ_o that is smaller than the maximum growth rate $\overline{\rho}$:

$$2.09 \approx \underline{\underline{\rho}}_{N_\mathrm{p}} \leq \rho_\mathrm{o} < \overline{\rho} \approx 2.85$$

To this end, we employ an objective function of the form

$$J(k) = J_\mathrm{out}(k) + \beta J_\mathrm{in}(k)$$

where

$$\begin{aligned} J_\mathrm{out}(k) &= \mathbb{E}\left\{ \sum_{j=0}^{N_\mathrm{p}-1} \max\Big(y(k+j) - r(k+j), 0\Big) \right\} \\ &= \sum_{\tilde{\ell}(k)\in\mathcal{L}_N} \left\{ \sum_{i=1}^{N_\mathrm{p}} \left[\Big(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) - \tilde{r}(k) \Big) \oplus \mathbb{1} \right]_i \right. \\ &\qquad \left. \cdot\, \tilde{P}[\tilde{L}(k) = \tilde{\ell}(k) \mid \ell(k-1),\, x(k-1),\, \tilde{u}(k),\, \tilde{v}(k)] \right\} \end{aligned}$$

and

$$J_\mathrm{in}(k) = -\sum_{i=1}^{N_\mathrm{p}} [\tilde{u}(k)]_i$$

with $\tilde{C}(\tilde{\ell}(k))$ and $\tilde{D}(\tilde{\ell}(k))$ defined as in Eq. (5-8), Eq. (5-9) and Eq. (5-10) on page 73, and the vectors $\tilde{\ell}(k)$, $\tilde{r}(k)$, $\tilde{v}(k)$ and $\tilde{u}(k)$ as in Eq. (5-7) on page 73.

6-1-3 Formulation of constraints

The system behaviour and input sequences are subject to several constraints that were introduced in Section 5-3-1 on page 72:

1. The system state evolution is according to the description of Eq. (6-1) on page 88.
2. The switching behaviour is according to Eq. (6-2) on page 88.
3. The admissible discrete control inputs $v(k)$ and discrete sFSM state evolution are defined by Figure 6-2 on page 90.
4. The continuous input signal $u(k)$ and its rate of change are nonnegative.
5. The absolute difference between the continuous input signal $u(k)$ and the nondecreasing reference signal $r(k)$ is bounded by a finite constant $\nu_{\max}$ as in Eq. (5-6) on page 68.

A linear implementation of these constraints into an optimisation problem has been discussed in Section 5-3-1 on page 72. The paragraphs below discuss an altered implementation of these five constraints that has been used for these case studies.

Constraint 1 The system state evolution has been implemented in the same way as introduced before:

$$\begin{aligned} t_{i,m} &\geq \left[\tilde{C}\left(\tilde{\ell}^m\right)\right]_{i,l} + x_l(k-1) - \tilde{r}_i(k), \quad \forall i,\, m,\, l \\ t_{i,m} &\geq \left[\tilde{D}\left(\tilde{\ell}^m\right)\right]_{i,l} + \tilde{u}_l(k) - \tilde{r}_i(k), \quad \forall i,\, m,\, l \\ t_{i,m} &\geq 0 \end{aligned}$$

Constraint 2 The switching behaviour is incorporated into the objective function as stochasticity information in $\tilde{P}_{\mathrm{s}}$:

$$J_{\text{out}} = \sum_{i=1}^{N_{\mathrm{p}}} \sum_{m=1}^{M} t_{i,m} \tilde{P}_{\mathrm{s}} \left[\tilde{L}(k) = \tilde{\ell}^m \mid \tilde{v}(k)\right] \tag{6-7}$$

where $\tilde{\ell}$ is a set of M possible mode sequences over horizon N_{p}. Then, the mode switching occurs online during the execution of the MPC.

Constraint 3 At every control step, the discrete sFSM state advances according to the behaviour in Figure 6-2 on page 90 through various logical statements and is stored in the logical vector x_b, such that $[x_b]_i = 1$ if state Q_i of the sFSM is active and 0 otherwise, with:

$$Q = \begin{bmatrix} \tilde{Q}_1 \\ \tilde{Q}_2 \\ \tilde{Q}_3 \end{bmatrix} \in \{0,1\}^{9\times 1}, \quad \tilde{Q}_i = \begin{bmatrix} Q_{i1} \\ Q_{i2} \\ Q_{i3} \end{bmatrix} \quad \text{for } i = 1,2,3$$

The admissible control action $v(k)$ is then read from the vector C_{x_b}:

$$C_{x_b} = \begin{bmatrix} a \\ a \\ b \\ a \\ a \\ c \\ a \\ a \\ d \end{bmatrix},$$

with the sets a, b, c and d defined in Eq. (6-3) on page 90. This information is translated into a set of admissible control sequences $\tilde{v}_{\text{adm}}$ over prediction horizon N_{p} and fed into the MPC algorithm, such that:

$$\begin{aligned} &\tilde{v}(k) + \neg(\tilde{v}_{\text{adm}}) \leq 1.5 \\ &\sum \tilde{v}(k) = 1 \end{aligned} \tag{6-8}$$

where $\neg(\tilde{v}_{\text{adm}})$ denotes the inverse of the logical vector $\tilde{v}_{\text{adm}}$. The result of these constraints is that the optimiser can only choose one discrete control sequence $\tilde{v}(k)$ over the prediction horizon N_{p}, and it has to be within the set of admissible sequences $\tilde{v}_{\text{adm}}$.

Constraint 4 The nonnegativity of the input signal $u(k)$ and its rate of change is incorporated as:

$$\begin{aligned} u(k) &\geq 0, \quad \forall k \\ \Delta u(k) &\geq 0, \quad \forall k \end{aligned}$$

Constraint 5 The input signal $u(k)$ is bounded with respect to the reference signal $r(k)$ as:

$$\begin{aligned} u(k) - r(k) &\leq \nu_{\max} \quad \forall k \\ u(k) - r(k) &\geq -\nu_{\max} \quad \forall k \end{aligned}$$

The next section discusses the optimisation problem that takes these constraints into account.

6-1-4 Optimisation problem

The setup of the optimisation problem over horizon N_p that takes into account the objective function and the constraints is of the form

$$\begin{aligned}
\min_{\tilde{u}(k),\,\tilde{v}(k)} \quad & J_\mathrm{out}(k) + \beta J_\mathrm{in}(k) && \text{(6-9a)}\\
\text{s.t.} \quad & \tilde{y} = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) && \text{(6-9b)}\\
& \tilde{\ell}(k) = \tilde{\phi}_\mathrm{stoch}(\tilde{\ell}(k-1),\, \tilde{v}(k)) && \text{(6-9c)}\\
& x_b(k+1) = f_\mathrm{sFSM}(x_b(k),\, \ell(k)) && \text{(6-9d)}\\
& \tilde{v}(k) \in \tilde{v}_\mathrm{adm}(k) && \text{(6-9e)}\\
& u(k) \geq 0 && \text{(6-9f)}\\
& \Delta u(k+j) \geq 0, \quad j = 0, \ldots, N_\mathrm{p} - 1 && \text{(6-9g)}\\
& |u(k+j) - \rho_\mathrm{r} \cdot (k+j)| \leq \nu_\mathrm{max}, \quad j = 0, \ldots, N_\mathrm{p} - 1 && \text{(6-9h)}
\end{aligned}$$

Eq. (6-9b) and Eq. (6-9c) reflect the general SMPL dynamics and Eq. (6-9d) the sFSM dynamics. The last four, Eq. (6-9e) to Eq. (6-9h) constrain the input sequences $\tilde{u}(k)$ and $\tilde{v}(k)$.

The prediction horizon N_p is set to a value of 3 as a result of balancing efficiency and performance, and no separate control horizon N_c is employed.

6-1-5 Controller setup

In both frameworks, the input cost function $J_\mathrm{in}(k)$ accounts in a linear fashion for continuous control. In contrast, the output cost function $J_\mathrm{out}(k)$ in Eq. (6-7) on page 93 multiplies the optimisation variables $t_{i,m}$ with the probability $\tilde{P}_\mathrm{s}$ that depends on the optimisation variable $\tilde{v}(k)$. As a result, the objective function $J(k)$ has a quadratic formulation, leading to a mixed-integer quadratic programming (MIQP) framework. This form may be converted into a mixed-integer linear programming (MILP) formulation through the transformation described in Section 5-3-1 on page 72.

The constraints of the MILP framework that were discussed in this section are represented in MATLAB [67] by YALMIP [68] as a set of 9262 or 9355 linear matrix inequalities (LMIs) for discrete and hybrid control, respectively. In contrast, the constraints are represented by 514 or 607 LMIs when modelled as an MIQP problem. This large discrepancy in the number of LMIs may manifest as longer computation times during the setup phase, as well as during the optimisation phase. Empirically, the best framework for the first and second case studies is the MIQP one, where the third one is modelled as an MILP problem.

The MPC problem for the three cases described using YALMIP in MATLAB is solved by Gurobi [69]. The controller object has the system's state vector $x(k-1)$ as described in Eq. (6-1) on page 88, the reference signal value $r(k)$ as in Eq. (6-6) on page 91, and the set of admissible discrete control signals $\tilde{v}_\mathrm{adm}$ as in Eq. (6-8) as inputs. The controller outputs the values of $u(k)$ and $v(k)$ over the prediction horizon N_p and does so for 200 simulation steps.

6-2 Discrete Control of a Mode-Constrained Deterministic SMPL System

This section shows the results of the control method when applied to a mode-constrained deterministic SMPL system under discrete control. It discusses the adaptations to the general setup as introduced in Section 6-1 on page 88 and presents the simulation results. First, the adaptations are introduced in the same order as before: System description, Objective function, Formulation of constraints, Optimisation problem and Controller setup.

6-2-1 System description

Since the system is deterministic, it is presented in the form of a DHA, introduced in Section 2-3-3 on page 18, consisting of a SAS, an MS and an FSM.

SAS The system is adapted from the general switched affine system (SAS) in Eq. (6-1) on page 88 to have the following form with no continuous inputs:

$$\begin{aligned} x(k) &= A^{(\ell(k))} \otimes x(k-1) \\ y(k) &= C \otimes x(k) \end{aligned} \tag{6-10}$$

where $A^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{3\times 3}$ and $C \in \mathbb{R}_{\varepsilon}^{1\times 3}$. The A-matrix is generated randomly for the three different modes and is approximately equal to:

$$\begin{aligned} A^{(1)} &\approx \begin{bmatrix} \varepsilon & \varepsilon & 1.5126 \\ 1.9395 & \varepsilon & 1.0671 \\ 1.9174 & 2.2776 & 1.2858 \end{bmatrix}, \\ A^{(2)} &\approx \begin{bmatrix} 0.4229 & 0.5541 & 1.9906 \\ \varepsilon & 2.0655 & 1.7282 \\ 0.7906 & 1.0565 & 1.3915 \end{bmatrix}, \\ A^{(3)} &\approx \begin{bmatrix} \varepsilon & \varepsilon & 3.6583 \\ 1.9722 & \varepsilon & 3.4227 \\ \varepsilon & 0.9468 & \varepsilon \end{bmatrix}. \end{aligned} \tag{6-11}$$

The system generation has been performed according to the procedure described in Example 4.3 on page 57 and using the script `generateSystem.m` in Appendix A-2-9 on page 164. Note that the values shown here are an approximation to the real entries of the matrices that are defined by up to 56 digits. The C-matrix is constant across the modes:

$$C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}. \tag{6-12}$$

Lastly, the initial state $x(0)$ and initial mode $\ell(0)$ are set to:

$$x(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \ell(0) = 1.$$

Since C consists of only finite values and every column of the three A-matrices has at least one finite element, the observability matrices $O_\alpha^M(\tilde{\ell})$ as defined in Definition 3.3 on page 30 are column-finite for every positive integer M. Therefore, the system is structurally observable. The B-matrices in this system can be assumed to have only ε as entries, and the system is not structurally controllable.

MS The mode selector (MS) is adjusted to reflect the deterministic switching behaviour:

$$\begin{aligned} &\ell(k) = \phi(\ell(k-1),\, v(k)) \\ &\ell(k) \in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N} \\ &P\left[L(k) = \ell(k) \mid \ell(k-1),\, v(k)\right] = \begin{cases} 1 & \text{for } \ell(k) = v(k) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6-13}$$

where $v(k) \in \{1,\, 2,\, 3\}$ is an integer.

FSM The deterministic finite state machine (FSM) constraining the mode sequence of the system is equal to the one shown in Figure 6-1 on page 89.

Lastly, the system has the following dimensions that are unchanged compared to Eq. (6-4) on page 90, except for n_u:

$$\begin{aligned} n_x &= 3 \quad \text{(Number of states)} \\ n_y &= 1 \quad \text{(Number of outputs)} \\ n_u &= 0 \quad \text{(Number of continuous inputs)} \\ n_v &= 1 \quad \text{(Number of discrete inputs)} \\ n_L &= 3 \quad \text{(Number of modes)} \end{aligned}$$

And has the following growth rate values, calculated using $N_\mathrm{p} = 6$, $N_\mathrm{sim} = 96$ and $N_\mathrm{t} = 10$ according to Algorithm 4.1 on page 50 using the script `growthRate.m` in Appendix A-2-10 on page 165:

$$\begin{aligned} \overline{\rho} &\approx 2.85 && \text{(Maximum growth rate)} \\ \underline{\underline{\rho}}_{N_\mathrm{p}} &\approx 1.91 && \text{(Finite-horizon estimation of minimum expected growth rate)} \\ \underline{\underline{\rho}}^{\mathrm{con}}_{N_\mathrm{p}} &\approx 1.94 && \text{(Finite-horizon estimation of minimum expected constrained growth rate)} \\ \underline{\rho}_{N_\mathrm{p}} &\approx 1.91 && \text{(Finite-horizon estimation of minimum growth rate)} \\ \underline{\rho}^{\mathrm{con}}_{N_\mathrm{p}} &\approx 1.94 && \text{(Finite-horizon estimation of minimum constrained growth rate)} \end{aligned} \tag{6-14}$$

Since there are no switching stochastics, the minimum expected growth rate is equal to the minimum growth rate.

6-2-2 Objective function

As we stabilise the system using only discrete control and no continuous control, we cannot expect the system to track a reference signal $r(k)$ flawlessly. Instead, we aim to achieve a

growth rate ρ_o that is at least as low as the rate of change of the reference signal ρ:

$$\rho_\mathrm{o} \leq \rho_\mathrm{r} + \epsilon.$$

Here, the finite margin ϵ is a function of the weight parameter γ in the cost function below and as introduced in Section 5-3-2 on page 77. Furthermore, we want to minimise the distance $|\rho_\mathrm{o} - \rho_\mathrm{r}|$.

To this end, we employ an objective function of the form

$$J(k) = \sum_{\tilde{\ell} \in \mathcal{L}_N} \left\{ \sum_{i=1}^{N_\mathrm{p}} \left[\left(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) - \tilde{r}(k) \right) \oplus \gamma \left(\tilde{r}(k) - \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \right) \right]_i \right\} \quad (6\text{-}15)$$

with $\tilde{C}(\ell(k))$ defined as in Eq. (5-9) on page 73, and the vectors $\tilde{\ell}(k)$, $\tilde{r}(k)$ as in Eq. (5-7) on page 73. The cost function does not penalise the discrete control signal $v(k)$. Instead, introducing the second term in the maximisation operation forces the optimisation problem to consider minimising the scaled distance $r(k) - y(k)$. The weight factor γ, introduced in Section 5-3-2 on page 77, balances the convergence speed and the penalisation of positive differences $r(k) - y(k)$. In this simulation, the value is set to $\gamma = 0.05$; however, additional simulations with $\gamma = 0$, $\gamma = 0.01$ and $\gamma = 1$ are shown in Appendix A-1 on page 124 to visualise its influence.

6-2-3 Formulation of constraints

Without the employment of continuous control through $u(k)$, the system is subject to only three of the five constraints introduced in Section 6-1-3 on page 93:

1. The system state evolution is according to the description of Eq. (6-10) on page 96.
2. The switching behaviour is according to Eq. (6-13) on page 97.
3. The admissible discrete control inputs $v(k)$ and discrete FSM state evolution are defined by Figure 6-1 on page 89.

They are implemented in the same way as in Section 6-1-3 on page 93.

6-2-4 Optimisation problem

With the omission of continuous control, the optimisation problem has the following form:

$$\begin{aligned}
\min_{\tilde{v}(k)} \quad & J(k) && (6\text{-}16a)\\
\text{s.t.} \quad & \tilde{y} = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) && (6\text{-}16b)\\
& \tilde{\ell}(k) = \phi(\tilde{\ell}(k-1),\, \tilde{v}(k)) && (6\text{-}16c)\\
& x_b(k+1) = f_\mathrm{FSM}(x_b(k),\, \ell(k)) && (6\text{-}16d)\\
& \tilde{v}(k) \in \tilde{v}_\mathrm{adm}(k) && (6\text{-}16e)
\end{aligned}$$

6-2-5 Controller setup

The controller setup is similar to the one described in Section 6-1-5 on page 95.

The constraints take the form of 514 LMIs, and the objective function is quadratic with 108 optimisation variables, leading to an MIQP problem. The controller object takes the system's state vector $x(k-1)$ as described in Eq. (6-1) on page 88, the reference signal value $r(k)$ as in Eq. (6-6) on page 91, and the set of admissible discrete control signals $\tilde{v}_{\mathrm{adm}}$ as in Eq. (6-8) on page 94 as inputs. The controller outputs the values of $v(k)$ over the prediction horizon N_{p} and does so for 200 simulation steps. The simulation performed by MATLAB version R2022a takes on average 0.8 s on an HP ZBook Studio G5 with an Intel Core i7-8750H CPU with a base clock speed of 2.20 GHz and 16.0 GB of RAM. Appendix A-2-1 on page 128 contains the MATLAB script used for the simulation.

6-2-6 Results of the simulation

Figure 6-5 on page 100 shows the simulation results of this case study. As expected, the difference signal $y(k)-r(k)$ stays close to and predominantly below zero after first convergence. This is to be expected since the reference signal $r(k)$ is not below the set of achievable growth rates as indicated by the first plot. The third plot shows no violation of the constraint on the mode sequence as there is no sequence of equal modes with a length longer than three. Furthermore, the fourth plot shows that there is seemingly no synchronised evolution in the system.

The reader is referred to the simulation results in Appendix A-1 on page 124 for insight into the influence of parameter γ in Eq. (6-15).

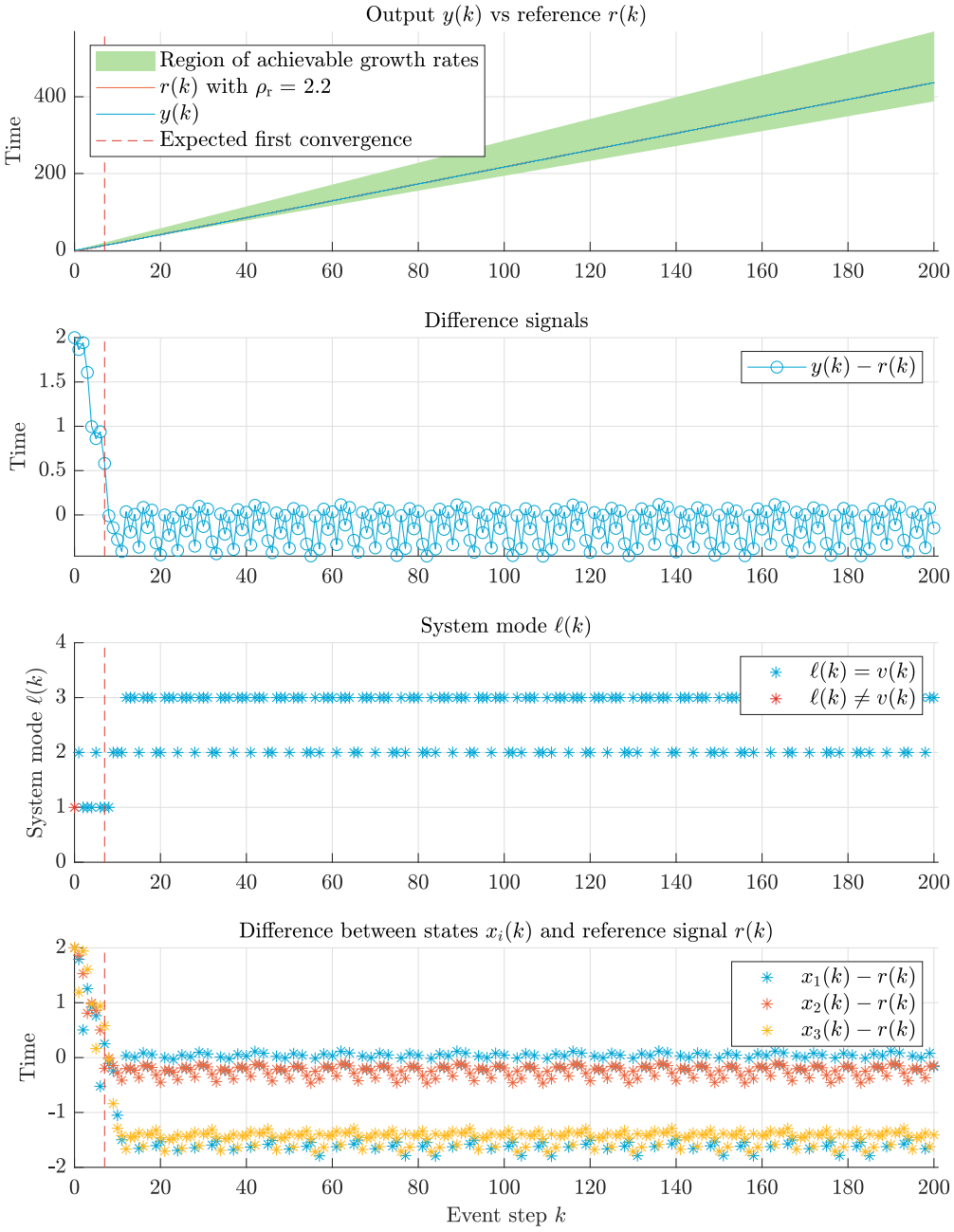

Figure 6-5: Simulation results of discrete control of a mode-constrained deterministic SMPL system with $\gamma = 0.05$. The plots show the output $y(k)$ versus the reference $r(k)$, the difference signal of $y(k) - r(k)$, the system mode $\ell(k)$ and the difference between the states $x_i(k)$ and the reference signal $r(k)$. Furthermore, it shows the expected event step at which the output $y(k)$ converges with the reference signal $r(k)$ based on the calculated growth rate $\underset{=N_p}{\rho^{con}}$ of the system, shown in Eq. (6-14) on page 97.

6-3 Hybrid Control of a Mode-Constrained Deterministic SMPL System

This section discusses the case study of hybrid control on a system similar to the one of Section 6-2 on page 96. It is subdivided into the following sections: System description, Objective function, Formulation of constraints, Optimisation problem, Controller setup and Results of the simulation.

6-3-1 System description

The system used in this study shares many similarities with the one introduced in Section 6-1-1 on page 88. In fact, it lacks only the stochasticity in the mode switching. The system is presented in the DHA framework.

SAS We utilise the following switched affine system (SAS):

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

where $A^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{3\times 3}$, $B^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{3\times 1}$ and $C \in \mathbb{R}_{\varepsilon}^{1\times 3}$. Compared to the previous case study, the system now has B-matrices with finite entries to enable continuous control. The A-matrices are as shown in Eq. (6-11) on page 96, and the B-matrices are approximately equal to:

$$B^{(1)} \approx \begin{bmatrix} 0.4271 \\ 4.8797 \\ 4.6932 \end{bmatrix},$$

$$B^{(2)} \approx \begin{bmatrix} 5.9921 \\ 1.9095 \\ 2.0984 \end{bmatrix},$$

$$B^{(3)} \approx \begin{bmatrix} 3.8990 \\ 3.1257 \\ 2.9753 \end{bmatrix}.$$

As with the A-matrices, these values are the rounded values of the real entries that consist of over 50 digits. The C-matrix consists of only zeros, as in Eq. (6-12) on page 96. We again use an initial state vector of only zeros and mode 1 as the initial mode. Besides being structurally observable as noted in Section 6-2 on page 96, the system is structurally controllable due to the finiteness of every element in the B-matrices. The reader is advised to revisit Section 3-1 on page 28 for the definitions of structural observability and structural controllability.

MS The mode selector (MS) is equal to the one of the previous case in Section 6-2 on page 96:

$$\begin{aligned}
&\ell(k) = \phi(\ell(k-1),\, v(k)) \\
&\ell(k) \in \mathcal{L} = \underline{n_L}, \quad k \in \mathbb{N} \\
&P\left[L(k) = \ell(k) \mid \ell(k-1),\, v(k)\right] = \begin{cases} 1 & \text{for } \ell(k) = v(k) \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

where $v(k) \in \{1,\, 2,\, 3\}$ can take the value of the three modes.

FSM For this deterministically switching system, the finite state machine (FSM) is as depicted in Figure 6-1 on page 89.

Lastly, the system dimension are equal to the ones presented in Section 6-1 on page 88, too:

$$\begin{aligned}
n_x &= 3 \quad \text{(Number of states)} \\
n_y &= 1 \quad \text{(Number of outputs)} \\
n_u &= 1 \quad \text{(Number of continuous inputs)} \\
n_v &= 1 \quad \text{(Number of discrete inputs)} \\
n_L &= 3 \quad \text{(Number of modes)}
\end{aligned}$$

And has the following growth rate values, calculated using $N_\mathrm{p} = 6$, $N_\mathrm{sim} = 96$ and $N_\mathrm{t} = 10$ according to Algorithm 4.1 on page 50 using the script `growthRate.m` in Appendix A-2-10 on page 165:

$$\begin{aligned}
\overline{\rho} &\approx 2.85 \quad \text{(Maximum growth rate)} \\
\underline{\rho}_{=N_\mathrm{p}} &\approx 1.91 \quad \text{(Finite-horizon estimation of minimum expected growth rate)} \\
\underline{\rho}^{\mathrm{con}}_{=N_\mathrm{p}} &\approx 1.94 \quad \text{(Finite-horizon estimation of minimum expected constrained growth rate)} \\
\underline{\rho}_{N_\mathrm{p}} &\approx 1.91 \quad \text{(Finite-horizon estimation of minimum growth rate)} \\
\underline{\rho}^{\mathrm{con}}_{N_\mathrm{p}} &\approx 1.94 \quad \text{(Finite-horizon estimation of minimum constrained growth rate)}
\end{aligned} \tag{6-17}$$

Since there are no switching stochastics, the minimum expected growth rate is equal to the minimum growth rate.

6-3-2 Objective function

Due to the employment of both discrete and continuous control, i.e., hybrid control, we expect the system output $y(k)$ to follow the reference signal $r(k)$ at least as effectively as with only discrete control. In this particular case, however, we expect the controller to perform better. The application of continuous control to a structurally controllable system can push output values up and closer to the reference value. Still, we aim to achieve a growth rate ρ_o that is at least as low as the growth rate of the reference signal $r(k)$:

$$\rho_\mathrm{o} \leq \rho_\mathrm{r}$$

Naturally, we want to minimise the distance $|\rho_{\mathrm{o}} - \rho_{\mathrm{r}}|$. Note that we cannot expect the system to achieve equality between these two values. In order to not overshoot the reference signal at a certain event step, the controller may at times decide to keep the system's output too low at earlier event steps. This peculiarity results from a single $u(k)$ value that steers multiple states, resulting in an underactuated system.

We utilise an objective function $J(k)$ that is equal to the one introduced in Section 6-1-2 on page 91:

$$\begin{aligned} J(k) &= J_{\mathrm{out}}(k) + \beta J_{\mathrm{in}}(k) \\ &= \sum_{\tilde{\ell}(k)\in\mathcal{L}_N} \left\{ \sum_{i=1}^{N_{\mathrm{p}}} \left[\left(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) - \tilde{r}(k) \right) \oplus \mathbb{1} \right]_i + \beta \sum_{i=1}^{N_{\mathrm{p}}} [-\tilde{u}(k)]_i \right. \end{aligned}$$

with $\tilde{C}(\tilde{\ell}(k))$ and $\tilde{D}(\tilde{\ell}(k))$ defined as in Eq. (5-8), Eq. (5-9) and Eq. (5-10) on page 73, and the vectors $\tilde{\ell}(k)$, $\tilde{r}(k)$, $\tilde{v}(k)$ and $\tilde{u}(k)$ as in Eq. (5-7) on page 73. The value of β is set to $\beta = 1 \times 10^{-2}$ to balance the influence of the two cost functions $J_{\mathrm{out}}(k)$ and $J_{\mathrm{in}}(k)$.

6-3-3 Formulation of constraints

The setup under hybrid control is subject to the same five constraints that were discussed in Section 6-1-3 on page 93:

1. The system state evolution is according to the description of Eq. (6-1) on page 88.
2. The switching behaviour is according to Eq. (6-2) on page 88.
3. The admissible discrete control inputs $v(k)$ and discrete FSM state evolution are defined by Figure 6-2 on page 90.
4. The continuous input signal $u(k)$ and its rate of change are nonnegative.
5. The absolute difference between the continuous input signal $u(k)$ and the nondecreasing reference signal $r(k)$ is bounded by a finite constant $\nu_{\max}$ as in Eq. (5-6) on page 68.

Their implementation into the MATLAB-script is as described in that section.

6-3-4 Optimisation problem

The aforementioned parts of the optimisation problem are summarised in the following description:

$$\min_{\tilde{u}(k),\,\tilde{v}(k)} \quad J_{\text{out}}(k) + \beta J_{\text{in}}(k) \tag{6-18a}$$

$$\text{s.t.} \quad \tilde{y} = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) \tag{6-18b}$$

$$\tilde{\ell}(k) = \tilde{\phi}(\tilde{\ell}(k-1),\, \tilde{v}(k)) \tag{6-18c}$$

$$x_b(k+1) = f_{\text{FSM}}(x_b(k),\, \ell(k)) \tag{6-18d}$$

$$\tilde{v}(k) \in \tilde{v}_{\text{adm}}(k) \tag{6-18e}$$

$$u(k) \geq 0 \tag{6-18f}$$

$$\Delta u(k+j) \geq 0, \quad j = 0,\, \ldots,\, N_{\text{p}} - 1 \tag{6-18g}$$

$$|u(k+j) - \rho_{\text{r}} \cdot (k+j)| \leq \nu_{\max}, \quad j = 0,\, \ldots,\, N_{\text{p}} - 1 \tag{6-18h}$$

6-3-5 Controller setup

The controller setup of this case is similar to the one of Section 6-1-5 on page 95 and consists of 607 LMIs describing the constraints paired with a quadratic objective function with 111 optimisation variables. The MIQP problem described by YALMIP in MATLAB is solved by Gurobi. The study is performed with a prediction horizon N_{p} of 3 over a period of 200 event steps. The simulation performed by MATLAB version R2022a takes on average 6.6 s on an HP ZBook Studio G5 with an Intel Core i7-8750H CPU with a base clock speed of 2.20 GHz and 16.0 GB of RAM. Appendix A-2-2 on page 137 contains the MATLAB script used to perform the case study.

6-3-6 Results of the simulation

Figure 6-6 on page 107 shows the results of the MATLAB simulation. The difference signal $y(k) - r(k)$ is on average closer to 0 than in the first case study without continuous control. Since the additional application of $u(k)$ is the only difference between the two studies, it is the sole cause of this improvement. Notably, the difference signal does not stay at 0 at all times. At every third event step, as seen in Figure 6-7 on page 108, the difference drops down to approximately -0.057.

To show the justification of the controller's decision to allow a nonzero difference $y(k) - r(k)$, let us consider the system's state at event step 20 and its future reference values:

$$x(20) \approx \begin{bmatrix} 42.00 \\ 41.87 \\ 41.19 \end{bmatrix}$$

$$r(20) = 42.00$$

$$r(21) = 44.20$$

$$r(22) = 46.40$$

Given that $\ell(21) = 1$, the system may choose to employ a continuous signal of $u(21) \approx 39.32$, resulting in a state vector of:

$$x(21 \mid \ell(21) = 1,\, u(21) \approx 39.32) \approx \begin{bmatrix} 42.70 \\ 44.20 \\ 44.14 \end{bmatrix}$$

Instead, the MPC selects a signal of $u(21) \approx 39.24$, resulting in a state vector of:

$$x(21 \mid \ell(21) = 1,\, u(21) \approx 39.24) \approx \begin{bmatrix} 42.70 \\ 44.12 \\ 44.14 \end{bmatrix}$$

As a last step, assuming $\ell(22) = 1$, the state vector is:

$$x(22 \mid \ell(22) = 1,\, u(21) \approx 39.32,\, u(22) \approx 41.52) \approx \begin{bmatrix} 45.66 \\ 45.21 \\ 46.48 \end{bmatrix} \oplus \begin{bmatrix} 41.95 \\ 46.40 \\ 46.21 \end{bmatrix} = \begin{bmatrix} 45.66 \\ 46.40 \\ 46.48 \end{bmatrix}$$

$$x(22 \mid \ell(22) = 1,\, u(21) \approx 39.24,\, u(22) \approx 41.52) \approx \begin{bmatrix} 45.66 \\ 45.21 \\ 46.40 \end{bmatrix} \oplus \begin{bmatrix} 41.95 \\ 46.40 \\ 46.21 \end{bmatrix} = \begin{bmatrix} 45.66 \\ 46.40 \\ 46.40 \end{bmatrix}$$

Evidently, even with $u(22) = \varepsilon$, the state vector $x(22 \mid u(21) \approx 39.32)$ would have an entry larger than $r(22) = 46.40$. The only option is for the system to choose $\ell(22) = 2$, leading to:

$$x(22 \mid \ell(22) = 2,\, u(21) \approx 39.32,\, u(22) \approx 40.41) \approx \begin{bmatrix} 46.13 \\ 46.27 \\ 45.53 \end{bmatrix} \oplus \begin{bmatrix} 46.40 \\ 42.32 \\ 42.51 \end{bmatrix} = \begin{bmatrix} 46.40 \\ 46.27 \\ 45.53 \end{bmatrix}$$

Evidently, while being suboptimal at event step 21, the choice of $u(21) \approx 39.24$ leads to a significantly larger control value of $u(22) \approx 41.52$ at event step 22. This positive difference is reflected in the value of the objective function. Over the 200 event steps, the cost functions have the following values:

$$\begin{aligned} J_{\text{out}} &\approx +26.5370 \\ J_{\text{in}} &\approx -423.519 \\ J &\approx -369.982 \end{aligned}$$

Forcing the controller to use $u(22) \approx 39.32$ would lead to the following values:

$$\begin{aligned} J_{\text{out}}(u(22) \approx 39.32) &\approx +26.5370 \\ J_{\text{in}}(u(22) \approx 39.32) &\approx -423.509 \\ J(u(22) \approx 39.32) &\approx -396.972 \end{aligned}$$

Thus, the system employs a suboptimal value of $u(21)$ as a sacrifice for better long-term results, underlining the efficacy of predictive control methods. With this, we justify the nonzero difference $y(k) - r(k)$ at regular event intervals.

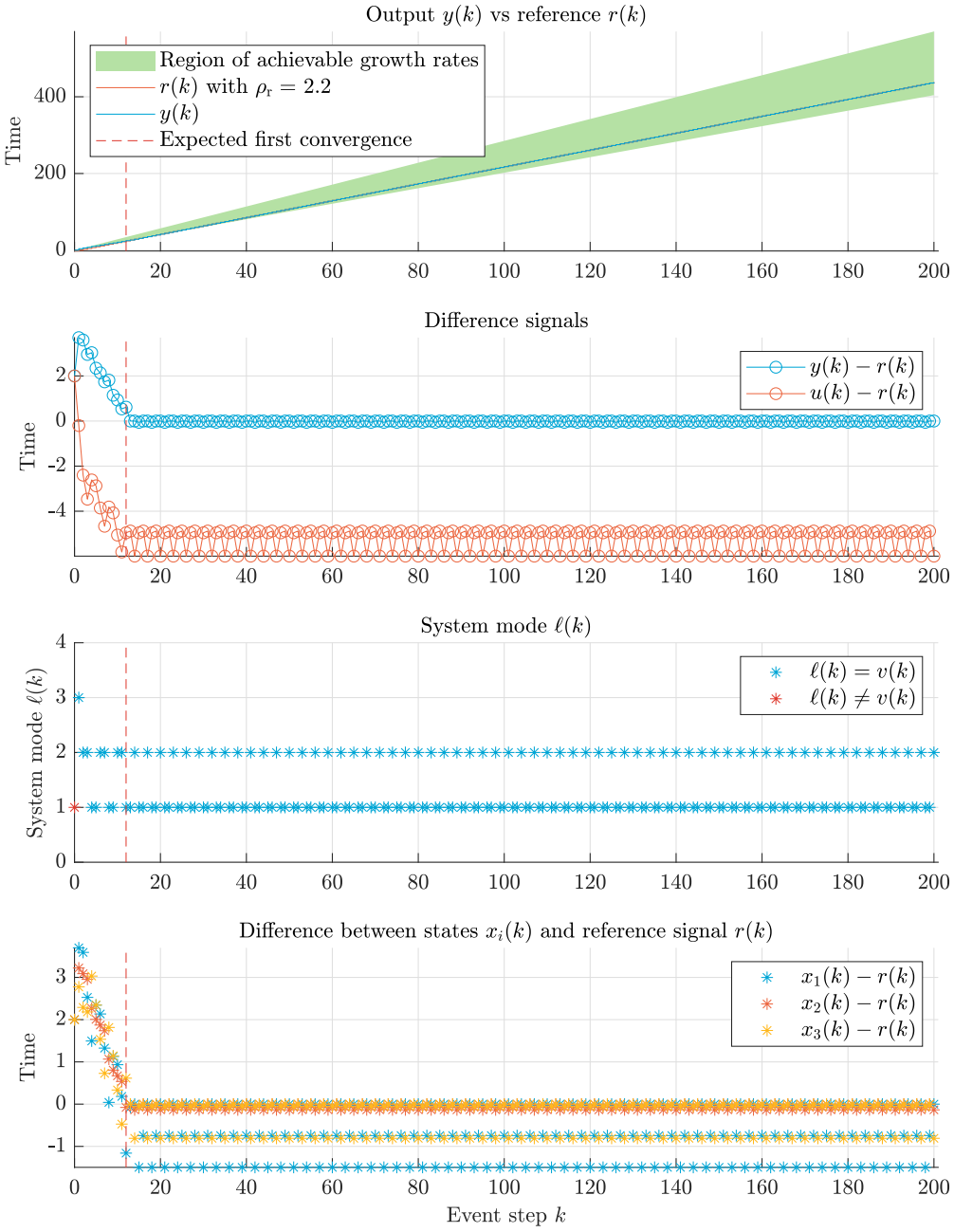

Figure 6-6: Simulation results of hybrid control of a mode-constrained deterministic SMPL system. The plots show the output $y(k)$ versus the reference $r(k)$, the difference signals of $y(k) - r(k)$ and $u(k) - r(k)$, the system mode $\ell(k)$ and the difference between the states $x_i(k)$ and the reference signal $r(k)$. Furthermore, it shows the expected event step at which the output $y(k)$ converges with the reference signal $r(k)$ based on the calculated growth rate $\underset{=N_p}{\rho^{con}}$ of the system, shown in Eq. (6-17) on page 102.

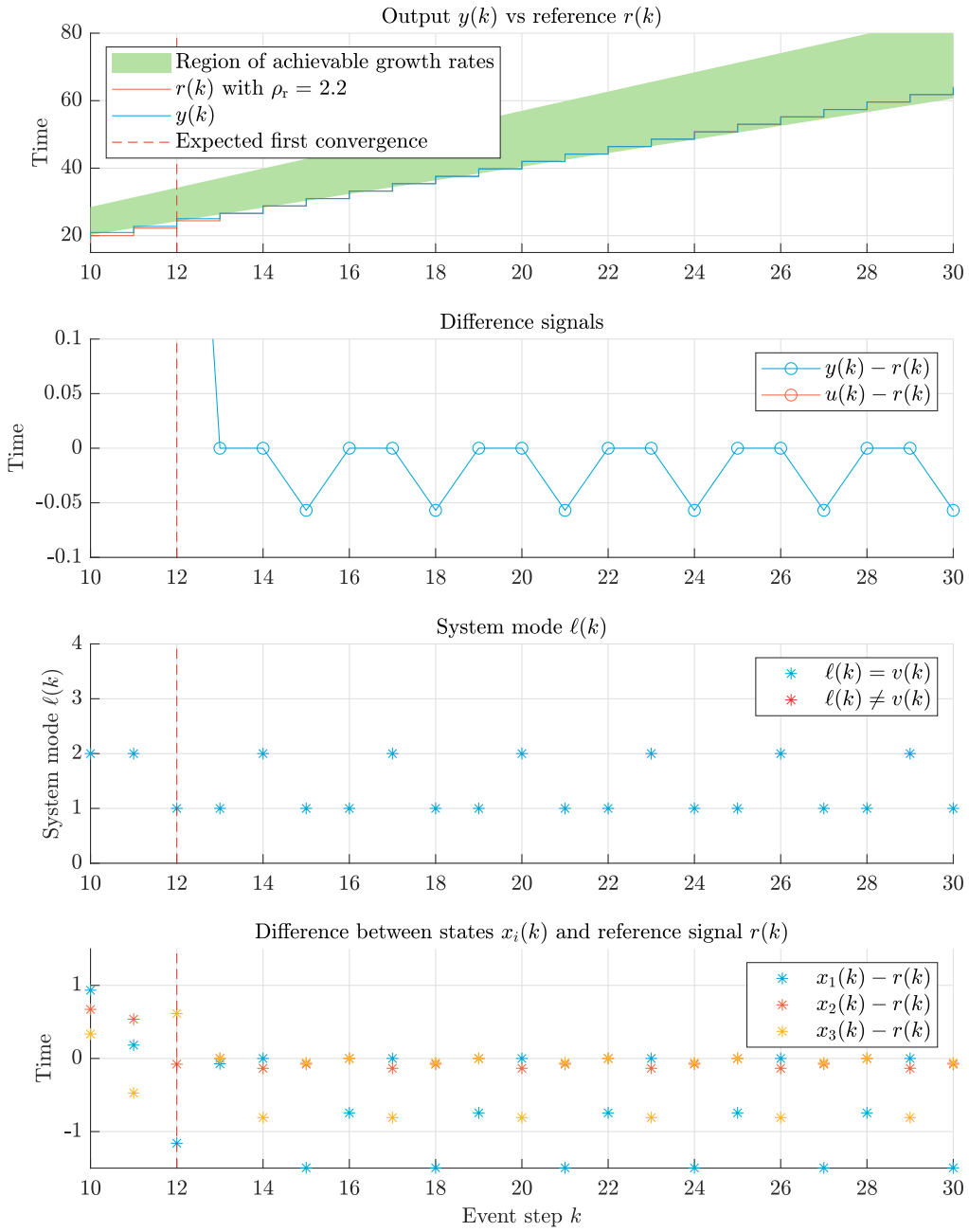

Figure 6-7: Zoomed-in version of Figure 6-6 on page 107 showing periodic nonzero values of the difference signal $y(k)-r(k)$. This peculiarity is predicted in Section 6-3-2 on page 102 and justified in the discussion of the results in Section 6-3-6 on page 104.

6-4 Hybrid Control of a Mode-Constrained Stochastic SMPL System

The sections System description, Objective function, Formulation of constraints, Optimisation problem and Controller setup discuss this case study's specific modelling approach, and the section Results of the simulation shows the outcome of the simulation. The setup is similar to the one introduced in Section 6-1 on page 88 and in Section 6-3 on page 101.

6-4-1 System description

The system is represented in the DHSA-framework due to the stochastics in the switching behaviour. It now consists of a SAS, an MS and an sFSM. The reader is referred to Section 6-1 on page 88 for a detailed explanation of the system.

SAS The switched affine system (SAS) takes the form of an SMPL system:

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

where $A^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{3\times 3}$, $B^{(\ell(k))} \in \mathbb{R}_{\varepsilon}^{3\times 1}$ and $C \in \mathbb{R}_{\varepsilon}^{1\times 3}$. The A, B and C matrices used in this case study are shown in Section 6-2-1 on page 96 and Section 6-3-1 on page 101. Again, the system is structurally controllable and structurally observable.

MS The mode selector (MS) is as depicted in Section 6-1 on page 88:

$$\begin{aligned} &\ell(k) = \phi_{\text{stoch}}(\ell(k-1),\, v(k)) \\ &\ell(k) \in \mathcal{L} = \{1,\, 2,\, 3\}, \quad k \in \mathbb{N} \\ &P\left[L(k) = \ell(k) \mid \ell(k-1),\, v(k)\right] = \begin{cases} 0.8 & \text{for } \ell(k) = v(k) \\ 0.1 & \text{otherwise} \end{cases} \end{aligned}$$

where $v(k) \in \{1,\, 2,\, 3\}$ is employed to alter the switching behaviour.

sFSM The stochastic finite state machine (sFSM) is explained in Section 6-1-1 on page 88 and visualised in Figure 6-2 on page 90.

The system dimensions are:

$$\begin{aligned} n_x &= 3 \quad \text{(Number of states)} \\ n_y &= 1 \quad \text{(Number of outputs)} \\ n_u &= 1 \quad \text{(Number of continuous inputs)} \\ n_v &= 1 \quad \text{(Number of discrete inputs)} \\ n_L &= 3 \quad \text{(Number of modes)} \end{aligned}$$

And has the following growth rate values, calculated using $N_\mathrm{p} = 6$, $N_\mathrm{sim} = 96$ and $N_\mathrm{t} = 10$ according to Algorithm 4.1 on page 50 using the script `growthRate.m` in Appendix A-2-10 on page 165:

$$\begin{aligned}
\overline{\rho} &\approx 2.85 \quad \text{(Maximum growth rate)} \\
\underline{\underline{\rho}}_{N_\mathrm{p}} &\approx 2.08 \quad \text{(Finite-horizon estimation of minimum expected growth rate)} \\
\underline{\underline{\rho}}^{\mathrm{con}}_{N_\mathrm{p}} &\approx 2.09 \quad \text{(Finite-horizon estimation of minimum expected constrained growth rate)} \\
\underline{\rho}_{N_\mathrm{p}} &\approx 1.91 \quad \text{(Finite-horizon estimation of minimum growth rate)} \\
\underline{\rho}^{\mathrm{con}}_{N_\mathrm{p}} &\approx 1.94 \quad \text{(Finite-horizon estimation of minimum constrained growth rate)}
\end{aligned} \tag{6-19}$$

Due to the switching stochastics, the minimum expected growth rate $\underline{\underline{\rho}}_{N_\mathrm{p}}$ differs from the minimum growth rate $\underline{\rho}_{N_\mathrm{p}}$.

6-4-2 Objective function

As in the previous case study, we employ hybrid control in order to achieve a growth rate ρ_o that is at least as low as the rate of change ρ_r of the reference signal $r(k)$:

$$\rho_\mathrm{o} \leq \rho_\mathrm{r}$$

and we aim to minimise the distance $|\rho_\mathrm{o} - \rho_\mathrm{r}|$. As a result of the uncertainty in the switching behaviour, we expect the controller to be less effective than when applied to a deterministic system. Indeed, the system needs to optimise for a future state that is unknown and unpredictable. The optimisation problem is set up such that the controller does not stay cautious in order to avoid overshooting $r(k)$ with $y(k)$; it merely minimises the expected overshoot as illustrated by choice of objective function $J(k)$:

$$\begin{aligned}
J(k) &= J_\mathrm{out}(k) + \beta J_\mathrm{in}(k) \\
&= \sum_{\tilde{\ell}(k) \in \mathcal{L}_N} \left\{ \sum_{i=1}^{N_\mathrm{p}} \left[\left(\tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) - \tilde{r}(k) \right) \oplus \mathbb{1} \right]_i + \beta \sum_{i=1}^{N_\mathrm{p}} [-\tilde{u}(k)]_i \right.
\end{aligned}$$

with $\tilde{C}(\tilde{\ell}(k))$ and $\tilde{D}(\tilde{\ell}(k))$ defined as in Eq. (5-8), Eq. (5-9) and Eq. (5-10) on page 73, and the vectors $\tilde{\ell}(k)$, $\tilde{r}(k)$, $\tilde{v}(k)$ and $\tilde{u}(k)$ as in Eq. (5-7) on page 73. The value of β is set to $\beta = 1 \times 10^{-2}$ in order to balance the influence of the two cost functions $J_\mathrm{out}(k)$ and $J_\mathrm{in}(k)$. As a result of this objective function, we expect the system to overshoot the reference signal at times. However, the controller will aim to minimise the positive difference $y(k) - r(k)$ while maximising $u(k)$.

6-4-3 Formulation of constraints

The setup is subject to the same five constraints as the one for the deterministic system under hybrid control:

1. The system state evolution is according to the description of Eq. (6-1) on page 88.
2. The switching behaviour is according to Eq. (6-2) on page 88.
3. The admissible discrete control inputs $v(k)$ and discrete FSM state evolution are defined by Figure 6-2 on page 90.
4. The continuous input signal $u(k)$ and its rate of change are nonnegative.
5. The absolute difference between the continuous input signal $u(k)$ and the nondecreasing reference signal $r(k)$ is bounded by a finite constant $\nu_{\max}$ as in Eq. (5-6) on page 68.

Their implementation into the MATLAB script is as described in Section 6-1-3 on page 93.

6-4-4 Optimisation problem

The optimisation problem takes the following form:

$$\begin{aligned}
\min_{\tilde{u}(k),\,\tilde{v}(k)} \quad & J_{\text{out}}(k) + \beta J_{\text{in}}(k) && \text{(6-20a)}\\
\text{s.t.} \quad & \tilde{y} = \tilde{C}(\tilde{\ell}(k)) \otimes x(k-1) \oplus \tilde{D}(\tilde{\ell}(k)) \otimes \tilde{u}(k) && \text{(6-20b)}\\
& \tilde{\ell}(k) = \tilde{\phi}(\tilde{\ell}(k-1),\, \tilde{v}(k)) && \text{(6-20c)}\\
& x_b(k+1) = f_{\text{sFSM}}(x_b(k),\, \ell(k)) && \text{(6-20d)}\\
& \tilde{v}(k) \in \tilde{v}_{\text{adm}}(k) && \text{(6-20e)}\\
& u(k) \geq 0 && \text{(6-20f)}\\
& \Delta u(k+j) \geq 0, \quad j = 0,\, \ldots,\, N_{\text{p}} - 1 && \text{(6-20g)}\\
& |u(k+j) - \rho_{\text{r}} \cdot (k+j)| \leq \nu_{\max}, \quad j = 0,\, \ldots,\, N_{\text{p}} - 1 && \text{(6-20h)}
\end{aligned}$$

6-4-5 Controller setup

The controller object described by YALMIP in MATLAB consists of 9355 LMIs describing the constraints paired with a linear objective function with 2190 optimisation variables. Note that the objective function has been linearised through the procedure discussed in Section 5-3-1 on page 72, such that the resulting optimisation problem is an MILP one instead of an MIQP one. The MILP study is performed with a prediction horizon N_{p} of 3 for 200 event steps. The simulation performed by MATLAB version R2022a takes on average 43 s on an HP ZBook Studio G5 with an Intel Core i7-8750H CPU with a base clock speed of 2.20 GHz and 16.0 GB of RAM. Appendix A-2-3 on page 147 contains the MATLAB script used to perform the case study.

6-4-6 Results of the simulation

To find a lower bound on the growth rate of the reference signal, we employed Algorithm 4.2 on page 59. Using a simulation horizon N_{sim} of 200 steps, equal to the simulation horizon of the

control case study, we sought to find the minimum offset o on the finite-horizon approximation of the minimum expected growth rate $\underline{\rho}_{=N_\mathrm{p}}$ such that 95 % of the control cases would be stabilisable using a discrete control sequence $\tilde{v}$ equal to the one employed to achieve $\underline{\rho}_{=N_\mathrm{p}}$. We used a multisample of cardinality 3.00×10^4, in order to make the statement with an accuracy of 1 % and confidence 0.995. Figure 6-8 on page 113[0] shows the result of the investigation, proposing an offset $o \geq 0.045$, such that we want a reference signal with a growth rate ρ_r that satisfies:

$$\rho_\mathrm{r} \geq \underline{\rho}^{\mathrm{con}}_{=N_\mathrm{p}} + o \approx 2.09 + 0.045 = 2.135$$

This result confirms that a reference signal with a growth rate of $\rho_\mathrm{r} = 2.2$ yields a stable response. We keep such a signal for faster convergence and better disturbance rejection.

Figure 6-9 on page 114 shows the final case study's simulation results. The red markers in the second and third plots denote event steps where the mode was not as intended, i.e., $\ell(k) \neq v(k)$, as a result of the modelled uncertainty. Notably, the desire of having $\rho_\mathrm{o} \leq \rho_\mathrm{r}$ for all $k > k_0$ with k_0 a finite integer is not achieved. The most evident deviation starts at event step 107, where several erroneous modes occurred close together. As seen in the third plot of Figure 6-9 and in the zoomed-in version in Figure 6-10 on page 115, the system often operated during that period in mode 3, the slowest mode according to Eq. (6-5) on page 90. Since the reference growth rate ρ_r is close to the finite-horizon approximation of the minimum expected constrained growth rate, as shown in Eq. (6-19) on page 110, there is not much room for the controller to recover from the overshoot. Hence, it takes 16 steps to fall below 0 again.

Note that the difference signal $y(k) - r(k)$ is expected to converge to 0 in 22 steps but does so after only 17. This, too, is a result of uncertainty in the switching behaviour. Although the mode switching has an error rate of around 27 % in the first 22 steps, as opposed to the expected 20 %, it never erroneously operates in the relatively slow mode $\ell = 3$. Thus, convergence occurs faster than expected in this specific study.

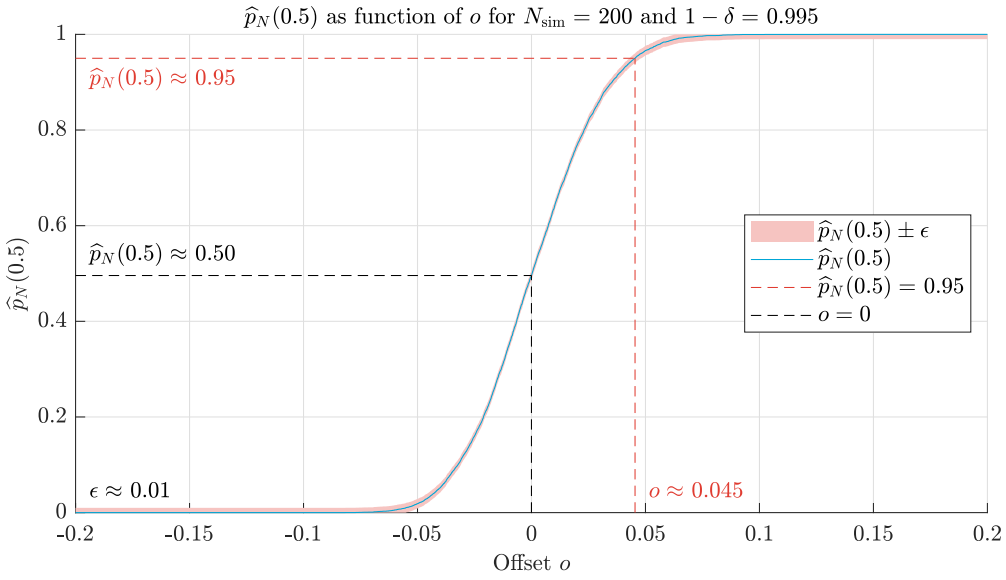

Figure 6-8: Minimum offset o necessary to obtain a probability of performance $\widehat{p}_N(0.5) = 0.95$ with accuracy $\epsilon = 0.01$ and confidence $1 - \delta = 0.995$ as calculated using Algorithm 4.2 on page 59. The algorithm proposes a minimum reference signal growth rate of $\rho_{\text{r}} = \underset{=N_{\text{p}}}{\rho} + 0.045 \approx 2.14$.

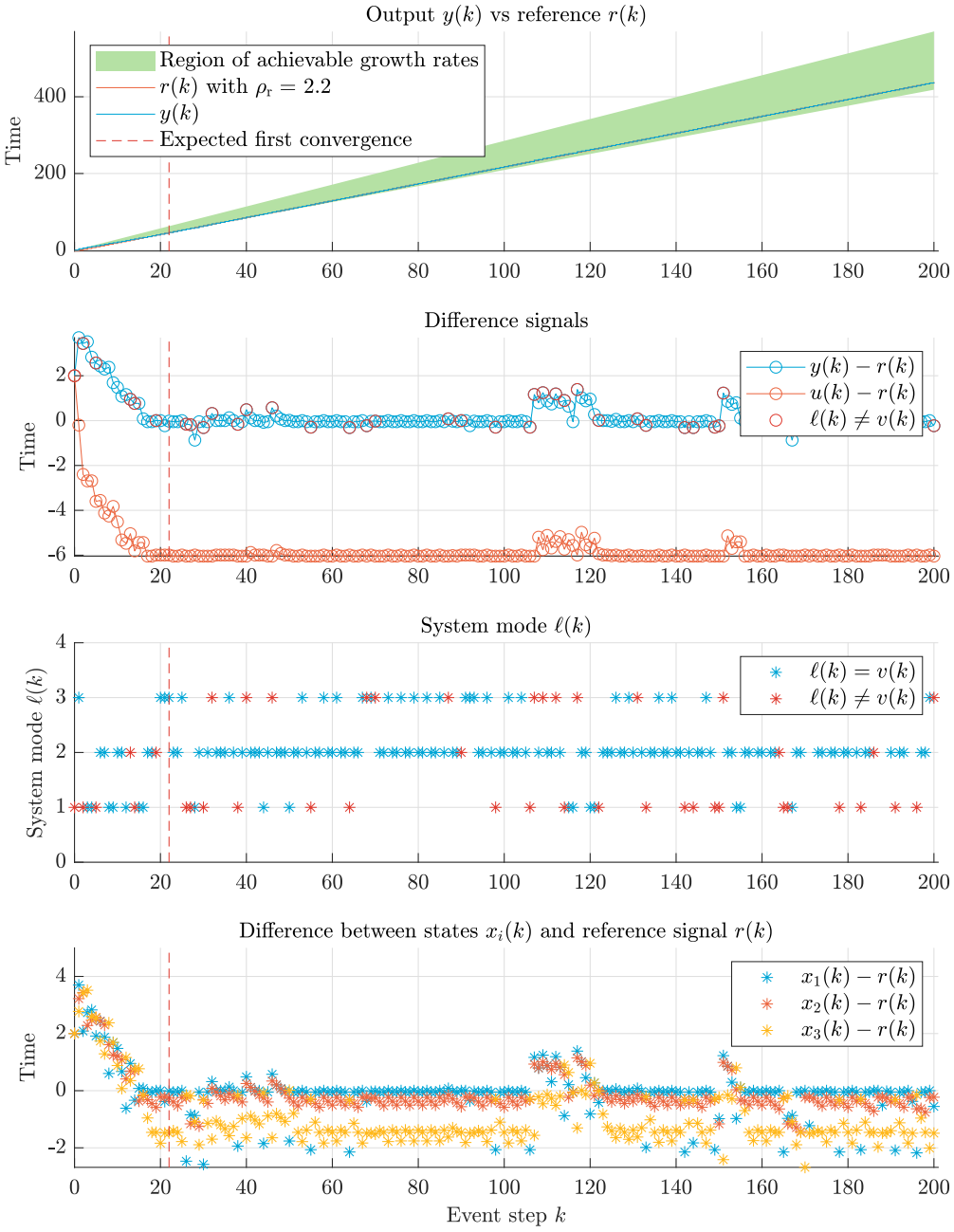

Figure 6-9: Simulation results of hybrid control of a mode-constrained deterministic SMPL system. The plots show the output $y(k)$ versus the reference $r(k)$, the difference signals of $y(k) - r(k)$ and $u(k) - r(k)$, the system mode $\ell(k)$ and the difference between the states $x_i(k)$ and the reference signal $r(k)$. Furthermore, it shows the expected event step at which the output $y(k)$ converges with the reference signal $r(k)$ based on the calculated growth rate $\underset{=N_p}{\rho^{con}}$ of the system, shown in Eq. (6-19) on page 110.

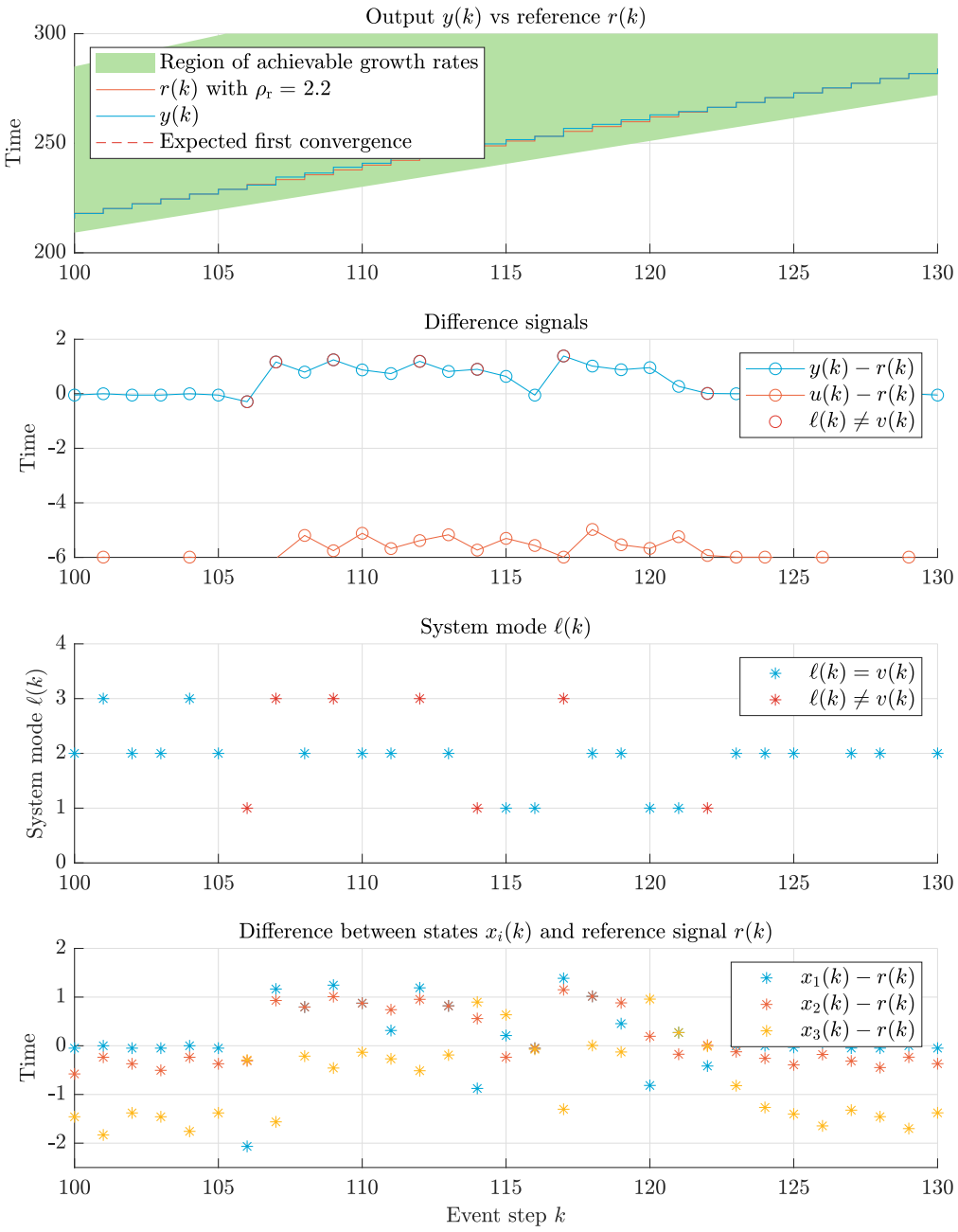

Figure 6-10: Zoomed-in version of Figure 6-9 showing nonzero values of the difference signal $y(k) - r(k)$.

Chapter 7

Conclusion

This final chapter evaluates the research described in this report, consisting of the chapters Preliminaries on page 5, Switching Max-Plus Linear Systems on page 27, Growth Rate of SMPL Systems on page 39, MPC Stabilisability of SMPL Systems on page 65 and Case Study on page 87. Section 7-1 on page 118 examines the obtained results and the steps taken to get there and evaluates the contributions of this work to the scientific community. Section 7-2 on page 121 proposes directions for future research based on the findings of this work.

7-1 Discussion and Contributions

Within the field of control theory, researchers and engineers aim to achieve satisfactory future system behaviour in a world governed by uncertainty and complexity. Many times, in light of efficiency and cost-effectiveness, uncertainty and complexity are discarded by introducing large margins around the mean performance. Uncertainty in system behaviour is neglected by considering worst-case scenarios or approximated by simulating the behaviour without consideration of bounds on the uncertainty set.

As a means to reduce modelling complexity for a specific type of nonlinear systems, the discrete-event systems (DESs), researchers produced the framework of switching max-plus linear (SMPL) systems. This framework offers a way to elegantly capture the nonlinear nature of maximisation and incorporate changes in system behaviour through switching. The field of model predictive control (MPC) has been extended to incorporate such systems and provide a powerful way of dealing with uncertainty and constraints. This research builds on the promise of MPC to deal with uncertainty and constraints in the switching behaviour of DESs. It poses a framework with which to assess the uncertain set of growth rates of SMPL systems and promises to incorporate many types of constraints on the switching sequence of such systems. Through simulations, this research validates the efficacy of the proposed methods and further reinforces the power of MPC when applied to systems described by max-plus algebra (MPA).

Chapter 2 on page 5 of this report offers a succinct introduction to MPA, MPC, graph theory, and Monte Carlo (MC) randomised algorithms. Chapter 3 on page 27 builds on this foundation by summarising the existing theory surrounding deterministic and stochastic SMPL systems and proposes a way of incorporating constraints on the systems' mode sequences. It does so by utilising the universality of discrete hybrid automata (DHAs) and discrete hybrid stochastic automata (DHSAs) introduced in the same two chapters. The report further explores the modest research on the growth rate of deterministic and stochastic SMPL systems in Chapter 4 on page 39 and poses new definitions and approximations that are helpful when assessing and predicting the behaviour of such systems. Chapter 5 on page 65 incorporates these definitions and examines the stability conditions for this type of system and its stabilisability. Furthermore, it presents a general MPC framework for control within a specific set of growth rates. Lastly, Chapter 6 on page 87 validates the research by exploring three case studies that show the efficacy of the research discussed in earlier chapters.

7-1-1 Contributions of this research

The contributions of this research are summarised in the following four paragraphs.

Contribution 1 – Comprehensive literature review A large part of the report forms a comprehensive review of the existing literature on max-plus algebra (MPA), the framework of switching max-plus linear (SMPL) systems and model predictive control (MPC) applied within that framework.

Contribution 2 – Growth rate definitions and approximations As an addition to the limited research on growth rate of switching max-plus linear (SMPL) systems, this report offers a way

to distinct various growth rates and take into account switching uncertainty and switching sequence constraints. It proposes computationally efficient algorithms to compute finite-horizon approximations of the infinite-horizon definitions. Furthermore, it presents a framework to assess the validity of these approximations.

Contribution 3 – Assessment of stability and stabilisability The report offers assessments of the stability and stabilisability of mode-constrained switching max-plus linear (SMPL) systems subject to deterministic or stochastic switching rules. This theory is paired with a universal framework of model predictive control (MPC) that can be applied to such systems.

Contribution 4 – Validation through simulation Lastly, the previously mentioned contributions are validated by exploring three case studies and performing simulations to show the potency of the frameworks presented by this research.

7-1-2 Answers to the research questions

Together, the four contributions mentioned in the previous section answer the research questions posed in Section 1-2 on page 3 in the following way.

Research Question 1 – What is the state of the art of stabilising deterministic and stochastic SMPL systems? The first research question has been answered in Chapter 4 on page 39 and Chapter 5 on page 65. A literature review on the subject has revealed that there exists an SMPL-MPC framework for systems under deterministic or stochastic switching. However, it considers only stabilisability above the maximum growth rate $\overline{\rho}$ and neglects any mode-constraints.

Research Question 2 – How can we incorporate mode constraints into the control framework? Section 3-3 on page 34 proposes a way to incorporate automaton-based mode constraints by combining the framework of SMPL systems with the one of DHAs and DHSAs. The incorporation of these constraints into the MPC controller is discussed in Section 6 on page 87. The implementation into the control algorithm to exlude the possibility of violation is discussed in Section 6-1-3 on page 93.

Research Question 3 – How can we quantify the performance of SMPL systems? The theory presented in Chapter 4 on page 39 offers a way to quantify system performance by measuring four growth rate quantities:

- Maximum growth rate $\overline{\rho}$
- Expected growth rate $\overline{\overline{\rho}}$
- Minimum expected growth rate $\underline{\underline{\rho}}$
- Minimum growth rate $\underline{\rho}$

The chapter introduces finite-horizon approximations of these infinite-horizon values and evaluates them in a statistical validation framework. Through that framework, we devised an algorithm to propose a lower bound on the growth rate of a system's reference signal to ensure stabilisability.

Research Question 4 – How can we stabilise deterministic and stochastic mode-constrained SMPL systems using discrete and hybrid control? Section 5-3 on page 72 introduces a framework to stabilise mode-constrained SMPL systems. Chapter 6 on page 87 offers three exemplary control cases to validate the theory. In short, we extended the SMPL-MPC framework to work for reference signals with growth rates lower than the maximum growth rate $\overline{\rho}$. Furthermore, we incorporated the mode constraints by simulating discrete-state system behaviour using stochastic finite state machines (sFSMs). Lastly, we discussed a way of converting mixed-integer quadratic programming (MIQP) problems that result from hybrid control to occasionally more efficient mixed-integer linear programming (MILP) ones.

Section 7-2 explores directions for future research outside of the original scope of this research or ones that were found to be appealing during the execution of the research.

7-2 Recommendations for Future Research

During the research, we were tempted to investigate many enticing branches that were considered outside of the scope of this thesis. Nonetheless, these branches could prove helpful in understanding the topics and extending or generalising the frameworks. Furthermore, they could increase the practical utility of the predominantly theoretical concepts of MPA and SMPL systems. A selection of them is detailed in the following paragraphs.

Research bounds on the infinite-horizon growth rate metrics Unlike the approximations proposed in this work that aim to be unbiased estimators of their infinite-horizon counterparts, it would prove helpful in a practical sense to additionally find upper and lower bounds. For example, considering that a system's growth rate may be increased arbitrarily through continuous control, it is helpful to know with certainty a minimum achievable growth rate.

Generalise the probability of performance as a function of the offset to other systems Whereas the conclusions drawn from the probability of performance investigation are now valid only for the system under investigation, it would be efficient to generalise the findings to other systems. For example, one could research ways of predicting the order of magnitude of the offset necessary to achieve a certain probability of performance without running the time-consuming simulations. This prediction may be based on the structure of the system matrices and the scale of their entries.

Investigate ways to make the algorithm to approximate infinite-horizon growth rate more efficient As is, the algorithm to calculate the finite-horizon growth rate approximations considers all possible mode sequences and control sequences over a certain horizon. However, many sequences may have a negligible probability of occurrence, and a growth rate close to the weighted average one. A way to improve the efficiency is to only consider sequences with a probability of occurrence above a certain threshold while ignoring or even incorporating their influence. Additional research is needed to investigate the bias that results from ignoring certain sequences.

Investigate ways to make the MPC algorithm more efficient In a similar fashion, one could improve the efficiency of the predictive control method by discarding certain unlikely mode sequences. Furthermore, one could incorporate a control horizon N_c as an addition to the prediction horizon N_p and introduce control heuristics for the region in between. Also, further research may reveal ways of optimising the form of the optimisation problem, yielding faster computation times.

Find proofs for the stabilisability hypotheses Little effort has been put into the a posteriori validation of the stabilisability hypotheses proposed in Section 5-2 on page 68. They are based on theorems from earlier work but have not been formally proven or otherwise substantiated. Future research may validate the hypotheses or propose alterations.

Generalise findings to more general frameworks such as DHSAs or max-min-plus-scaling (MMPS) systems Lastly, the definitions and frameworks targeted towards stochastic SMPL systems with mode constraints may be generalised towards broader classes of systems, such as all DHSAs, or MMPS systems.

Appendix A

Appendices

A-1 Influence of the Parameter γ in the MPC Cost Function

This appendix visualises the influence of the parameter γ in the cost function of Eq. (5-15) on page 79. It does so by showing the simulation results of the case study in Section 6-2 on page 96 with the following values of the parameter:

- $\gamma = 0$, Visualised by the simulation results in Figure A-1
- $\gamma = 0.01$, Visualised by the simulation results in Figure A-2 on page 126
- $\gamma = 1$, Visualised by the simulation results in Figure A-3 on page 127

The influence of these parameter values is compared to the value that was used in the case study of Section 6-2 on page 96:

- $\gamma = 0.05$, Visualised by the simulation results in Figure 6-5 on page 100

Results and interpretation The simulation with $\gamma = 0$ represents an objective function that only aims to minimise the positive difference $y(k)-r(k)$ and disregards the negative difference $r(k) - y(k)$. Therefore, any behaviour that leads to the output $y(k)$ being smaller than the reference value $r(k)$ is deemed equivalent in value. This leads to the arbitrary behaviour of Figure A-1, in which the difference $r(k) - y(k)$ increases steadily after reaching zero.

A value of $\gamma = 0.01$ is a factor five smaller than the one used in the case study, which is apparent from the system's behaviour in Figure A-2 on page 126. In the visualisation of the results, the output $y(k)$ is seen to converge slowly from below to the reference $r(k)$. Since the optimisation algorithm considers the maximum of $(y(k) - r(k))$ and $0.01 \cdot (r(k) - y(k))$, it is a lot more anxious about any positive difference $y(k) - r(k)$ than any negative. Therefore, the slower convergence compared to the case study's results is attributed to the factor five difference between the values of γ.

In contrast, for a value $\gamma = 1$, the system considers an equal penalty on both the positive and negative difference between the reference signal $r(k)$ and the system's output $y(k)$. It is, therefore, expected that we see an equal deviation in both directions, as is confirmed by Figure A-3 on page 127. The signal resembles a triangle wave with an apparent mean value of 0. While its rather large amplitude and period are likely an attribute of the small prediction horizon, additional research is needed for confirmation.

The value of $\gamma = 0.05$ used in the case study results from the trade-off between convergence speed and the strictness of the nonpositiveness of the difference signal $y(k) - r(k)$. Different applications may require different values of the parameter.

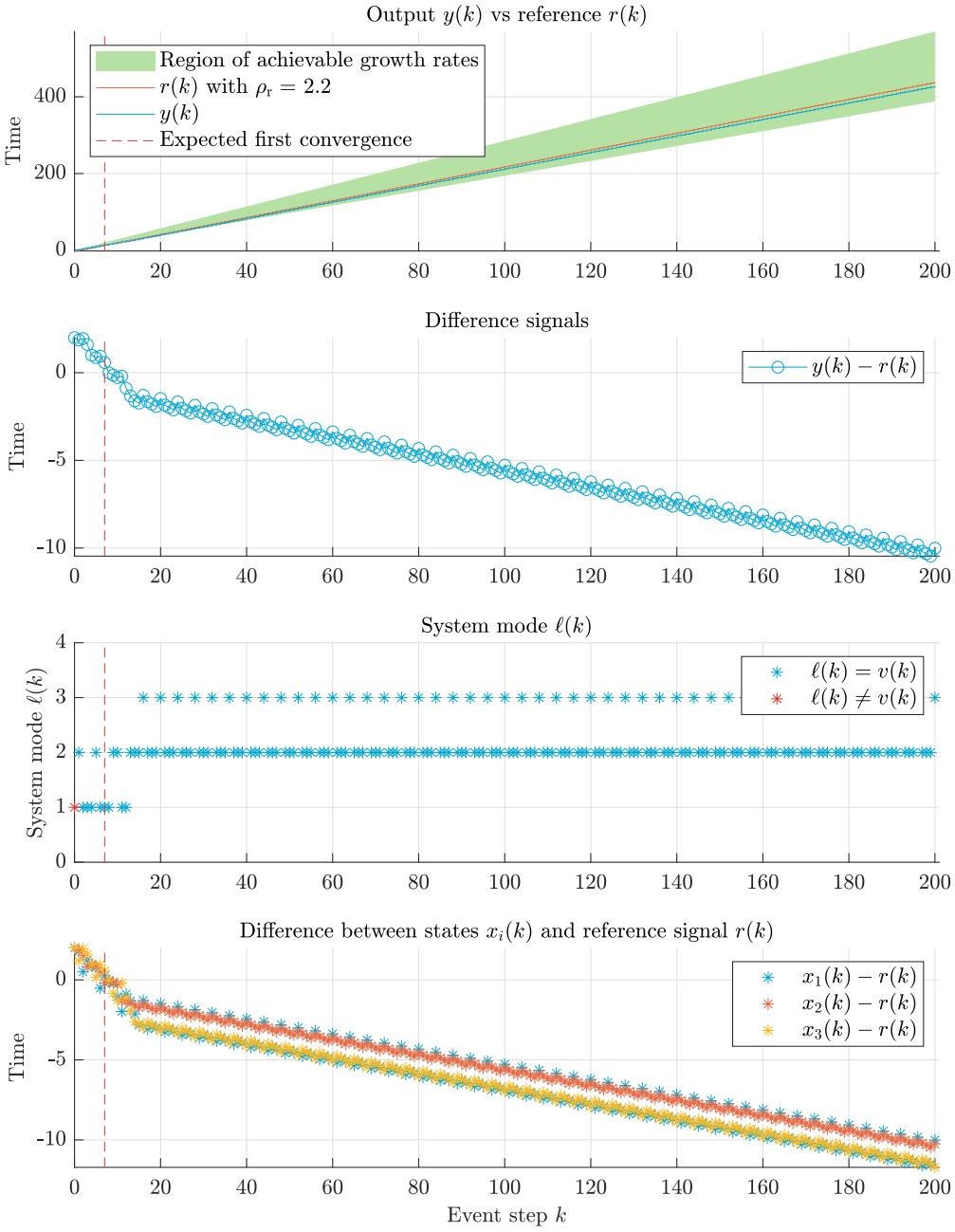

Figure A-1: Simulation results of discrete control of a mode-constrained deterministic SMPL system with $\gamma = 0$. The plots show the output $y(k)$ versus the reference $r(k)$, the difference signal of $y(k) - r(k)$, the system mode $\ell(k)$ and the difference between the states $x_i(k)$ and the reference signal $r(k)$. Furthermore, it shows the expected event step at which the output $y(k)$ converges with the reference signal $r(k)$ based on the calculated growth rate $\underset{=N_p}{\rho^{con}}$ of the system, shown in Eq. (6-14) on page 97.

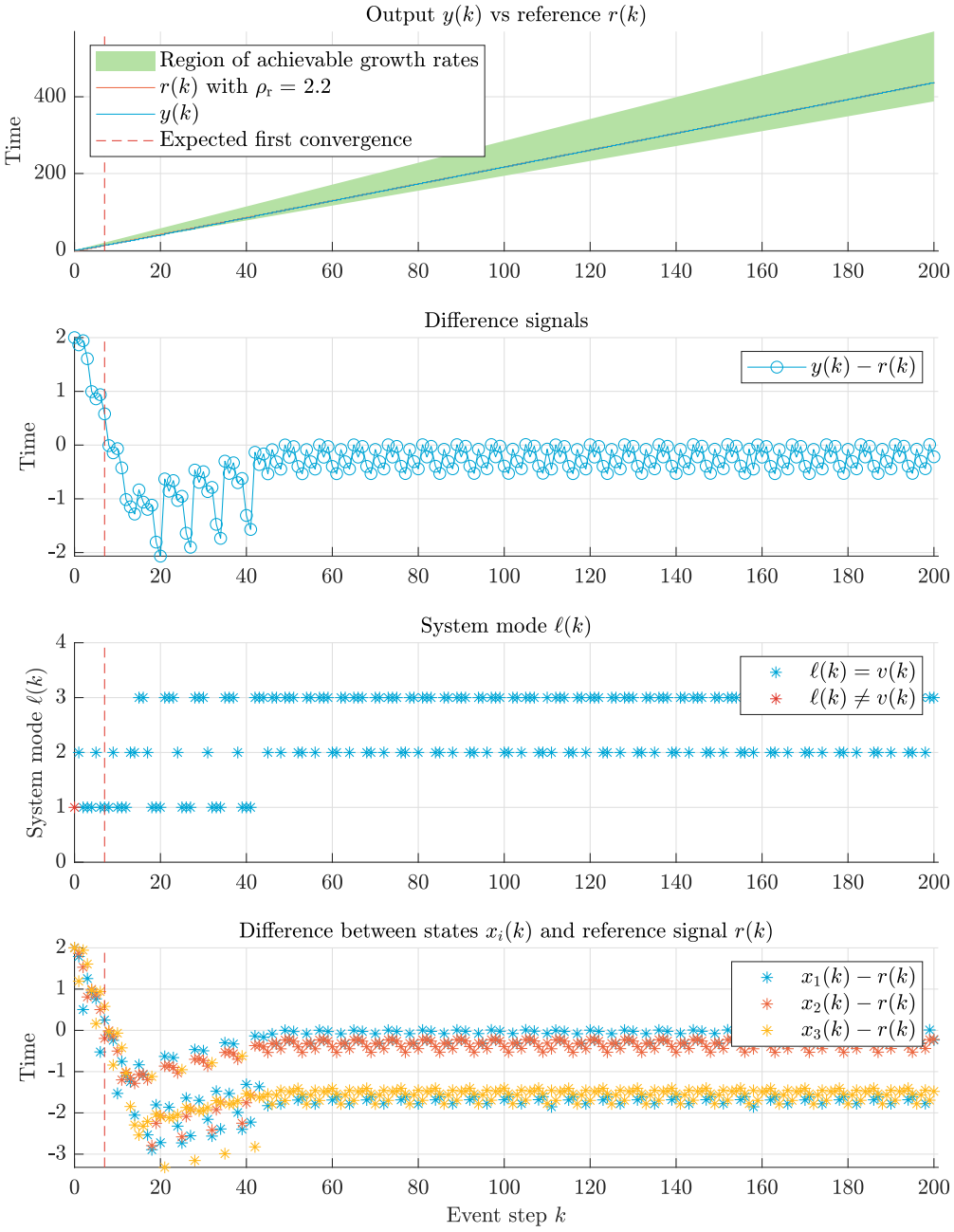

Figure A-2: Simulation results of discrete control of a mode-constrained deterministic SMPL system with $\gamma = 0.01$. The plots show the output $y(k)$ versus the reference $r(k)$, the difference signal of $y(k) - r(k)$, the system mode $\ell(k)$ and the difference between the states $x_i(k)$ and the reference signal $r(k)$. Furthermore, it shows the expected event step at which the output $y(k)$ converges with the reference signal $r(k)$ based on the calculated growth rate $\underset{=N_p}{\rho^{con}}$ of the system, shown in Eq. (6-14) on page 97.

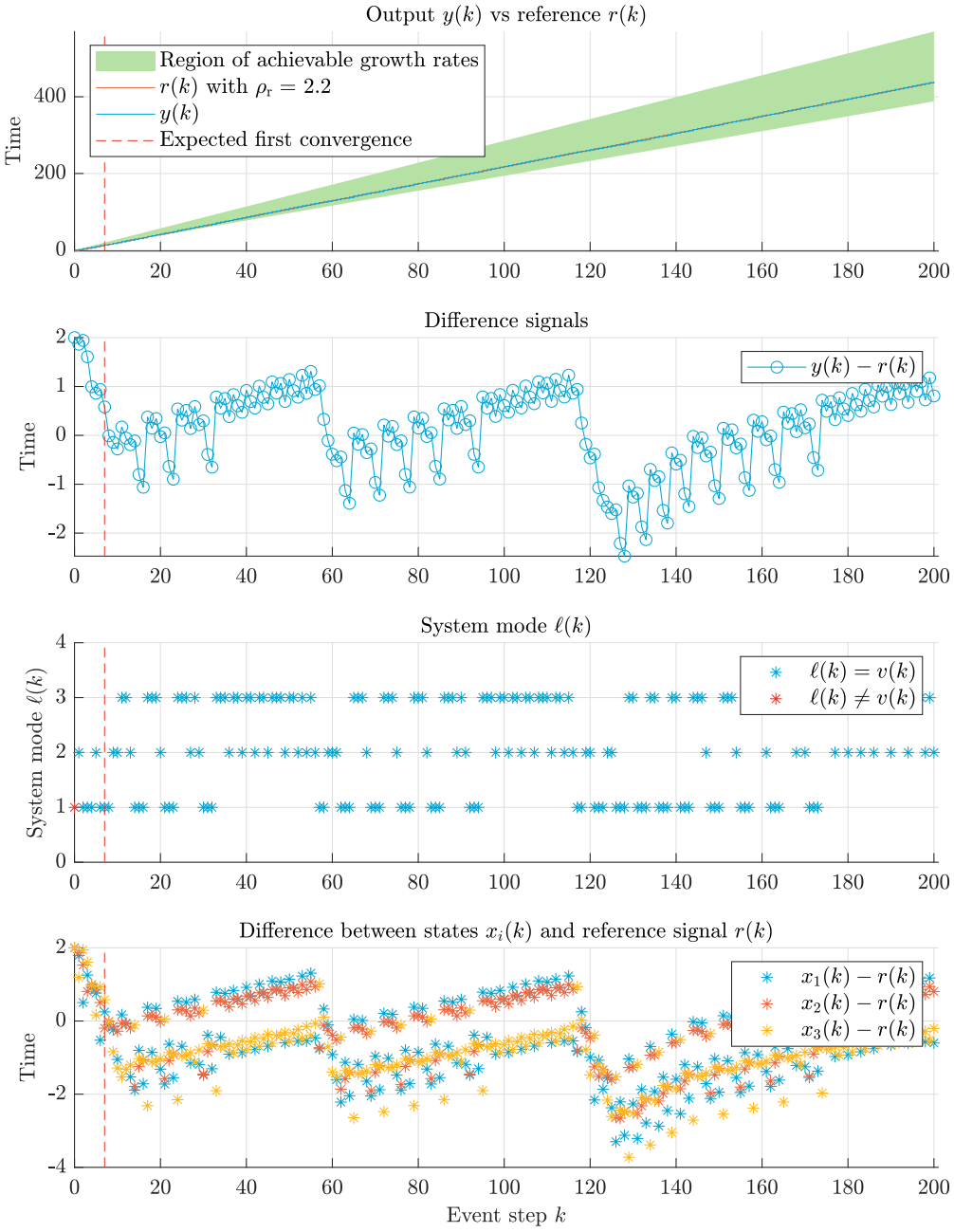

Figure A-3: Simulation results of discrete control of a mode-constrained deterministic SMPL system with $\gamma = 1$. The plots show the output $y(k)$ versus the reference $r(k)$, the difference signal of $y(k) - r(k)$, the system mode $\ell(k)$ and the difference between the states $x_i(k)$ and the reference signal $r(k)$. Furthermore, it shows the expected event step at which the output $y(k)$ converges with the reference signal $r(k)$ based on the calculated growth rate $\underset{=N_p}{\rho^{con}}$ of the system, shown in Eq. (6-14) on page 97.

A-2 Supporting MATLAB Scripts

A-2-1 `Case1.mlx`

The following script is a MATLAB live script.

```
%% Case 1: Discrete control of a mode-constrained deterministic system
% *File info:*
%%
% * Name:     |Case1.mlx|
% * Author:   Bart de Jong
% * Date:      June, 2022
%%
% *Prerequisite software*
%%
% * MATLAB    (Tested version: R2022a)
% * YALMIP     (Tested version: R20200116)
% * GUROBI    (Tested version: gurobi901)
%%
% *Prerequisite custom function files*
%%
% * |allSequences.m|
% * |checkError.m|
% * |findOffset.m|
% * |generateSemigroup.m|
% * |generateSupport.m|
% * |generateSystem.m|
% * |growthRate.m|
% * |modeConstraints.m|
% * |mpAdd.m|
% * |mpMulti.m|
% * |predictionModel.m|
%%
% *Prerequisite toolbox*
%%
% * Max-Plus Algebra Toolbox for Matlab - Copyright (C) 2016 Jaroslaw Stanczyk
% * Used for |mp_mcm.m| and supporting files
%%
%
%% Initialisation and settings
%
%
% *Clear workspace*

clearvars
%%
%
%
% *Preliminaries and general settings*

% Max-plus zero and one elements
other.eps       = -inf;
other.e         = 0;

% Colour vector for figure elements
other.colourvec = {'#00A6D6', '#E03C31', '#FFB81C', '#6CC24A', '#EC6842'};

```

```
52 % Latex interpreter
53 set(0, 'defaultAxesTickLabelInterpreter', 'latex');
54 set(0, 'defaultLegendInterpreter'       , 'latex');
55 set(0, 'defaultTextInterpreter'         , 'latex');
56 %%
57 %
58 %
59 % *Simulation settings*
60
61 Sim.Np            = 3;        % Prediction horizon
62 Sim.rho_r         = 2.2;      % Growth rate reference signal (constant in
       reference signal)
63 Sim.kmax          = 200;      % Number of event steps in the simulation
64 Sim.maxnummode    = 3;        % Maximum number of same mode in a row
65 Sim.lineariseMIQP = false;    % Turn MIQP into MILP
66
67 Sim.nu_max        = 10;       % Maximum difference between input and reference
68 Sim.gamma         = 0.05;     % Weight in output cost function
69 %%
70 %
71 %% Construct SMPL system
72 %
73 %
74 % *Define dimensions of the SMPL system*
75
76 sys.nx = 3;       % Number of states
77 sys.nu = 1;       % Number of contimous inputs (u(k)) > 0
78 sys.ny = 1;       % Number of outputs
79 sys.nL = 3;       % Number of modes
80 sys.nv = sys.nL;  % Number of discrete inputs (v(k)) - v(k) is a scalar, not a
       vector
81 %%
82 %
83 %
84 % *Generate SMPL system with no switching stochastics*
85
86 rng(9)   % Seed for random number generator
87 [sys.A, ~, sys.C, ~] = generateSystem(sys.nx, sys.nu, sys.ny, sys.nL);   %
       Construct autonomous system
88
89 % Define switching probabilities {0, 1}
90 sys.Ps = zeros(sys.nL, sys.nL, sys.nv);
91 for i = 1:sys.nL
92    sys.Ps(:, i, i) = 1;
93 end
94 %%
95 %% Construct prediction model
96 %
97 %
98 % *Find all possible mode sequences over the prediction horizon and put them
99 % in a matrix*
100
101 [sys.seqm, sys.seqv, sys.Ps_tilde] = allSequences(sys.nL, Sim.Np, sys.Ps);
102 %%
103 %
104 %
105 % *Find all possible sequences that do not violate the mode-constraint*
```

```
106
107 sys.violate_con = sum(diff(sys.seqm,[],2)==0, 2) > Sim.maxnummode-1; % Find all
      mode sequences over horizon Np
108 sys.seqm_con    = sys.seqm(~sys.violate_con, :);                    % Select
    the mode sequences
109 sys.seqv_con    = sys.seqv(~sys.violate_con, :);                    % Select
    the control sequences
110 %%
111 %
112 %
113 % *Calculate* $\tilde{A}$ *and* $\tilde{C}$ *matrices for all possible mode
114 % sequences*
115
116 [sys.A_tilde, sys.C_tilde, ~, ~] = predictionModel(sys.A, zeros(sys.nx, sys.nu,
      sys.nL), sys.C, sys.seqm);
117 %%
118 %
119 %% Calculate deterministic (constrained) growth rate
120 %
121 %
122 % *Calculate values*
123
124 clear growth
125 growth.Np_GR = 2*Sim.Np;   % Maximum period of cyclic mode sequences
126 [growth.seqm_GR, ~, growth.Ps_tilde_GR] = allSequences(sys.nL, growth.Np_GR,
    sys.Ps);
127
128 [growth_append] = growthRate(sys.A, sys.C, growth.seqm_GR, growth.Ps_tilde_GR,
    "maxnummode", Sim.maxnummode);
129 growth = cell2struct([struct2cell(growth);struct2cell(growth_append)],[
    fieldnames(growth);fieldnames(growth_append)]);
130 %%
131 %
132 %
133 % *Plot results*
134
135 figure; boxplot([growth.rho_dbar_Np; growth.rho_dbar_Np_con], ...
136    [repmat("Unconstrained deterministic system", [size(growth.rho_dbar_Np, 1),
         1]); ...
137    repmat("Constrained deterministic system", [size(growth.rho_dbar_Np_con, 1),
         1])], 'Whisker', 2);
138
139 a = get(get(gca,'children'),'children');   % Get the handles of all the objects
140 set(a(1:4), 'Color', other.colourvec{5});
141 set(a(5:6), 'Color', other.colourvec{1});
142
143 ylabel('Growth rate $\overline{\overline{\rho}}_{N_\mathrm{p}}$', 'Interpreter'
    , 'latex');
144 grid minor
145 set(gca, 'TickLabelInterpreter','latex')
146 %%
147 %% Find admissible control inputs based on FSM state
148 %
149
150 [sys.adm_cont, sys.adm_cont_seq] = modeConstraints(Sim.maxnummode, sys.nv, sys.
    seqv);
151 %%
```

```
152 %% Find lower bound on reference signal for required probability of perfomance
153 %
154
155 % Check for stochastics
156 Sim.ProbOfPerf = 0.95;
157 if any(sys.Ps(sys.Ps~=1)>0)
158    Nsim = 200;           % Growth rate simulation horizon
159    maxit = 3.00e4;       % Number of samples
160    offsetRange = 0.2;    % Maximum absolute offset
161    Sim.offsetPoP = findOffset(sys.A, sys.C, sys.Ps, Nsim, growth.
           rho_dubar_Np_con, growth.rho_dbar_Np, growth.seqm, maxit, Sim.ProbOfPerf,
            offsetRange);
162 else
163    Sim.offsetPoP = 0;
164 end
165 fprintf("- Offset necessary to obtain a probability of performance of %.2f with
        given accuracy and confidence: %.2f\n" + ...
166    "- Minimum expected constrained growth rate including offset: %.2f", Sim.
           ProbOfPerf, Sim.offsetPoP, growth.rho_dubar_Np_con + Sim.offsetPoP);
167 %% Setup MPC algorithm
168 %
169 %
170 % *Construct set of constraints*
171
172 Sim.constraints = [];
173
174 optimvar.t       = sdpvar(repmat(Sim.Np, 1, height(sys.seqm)), ones(1, height(
        sys.seqm)));   % Construct optimisation variables t
175 optimvar.x       = sdpvar(sys.nx, 1);              % Initial state
176 optimvar.ref     = sdpvar(Sim.Np, 1);              % Reference signal over
        prediction horizon
177 optimvar.v       = binvar(height(sys.seqv), 1);    % Vector denoting discrete
        control sequence
178 optimvar.vseqpos = binvar(height(sys.seqv), 1);    % Vector denoting which mode
        sequence is allowed
179
180 if Sim.lineariseMIQP
181    optimvar.tj = sdpvar(repmat(Sim.Np, 1, height(sys.seqm)*height(sys.seqv)),
           ones(1, height(sys.seqm)*height(sys.seqv)));
182    Sim.Mt = growth.rho_bar*(Sim.kmax+1);  % Upper bound on maximum of system's
           output for y(0)<=0
183    Sim.mt = 0;                            % Lower bound on minimum of system's
           output for y(0)>=0
184 end
185
186 for i = 1:sys.ny*Sim.Np          % Loop over prediction horizon
187    for j = 1:height(sys.seqm)    % Loop over all possible mode sequences
188       for k = 1:sys.nx           % Loop over all states
189          Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= 1*(sys.C_tilde
                 (i, k, j) + optimvar.x(k) - optimvar.ref(i))];             % System
                  dynamics
190          Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= Sim.gamma*(
                 optimvar.ref(i) - (sys.C_tilde(i, k, j) + optimvar.x(k)))];   %
                 System dynamics
191       end
192       if Sim.lineariseMIQP    % Linearise optimisation problem
193          for k = 1:height(sys.seqv)
```

```
194             Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                    seqv)+k}(i) <= Sim.Mt*optimvar.v(k)];
195             Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                    seqv)+k}(i) >= Sim.mt*optimvar.v(k)];
196             Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                    seqv)+k}(i) <= optimvar.t{j}(i)-Sim.mt*(1-optimvar.v(k))];
197             Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                    seqv)+k}(i) >= optimvar.t{j}(i)-Sim.Mt*(1-optimvar.v(k))];
198         end
199       end
200    end
201 end
202
203 Sim.constraints = [Sim.constraints, sum(optimvar.v)==1];
                                    % Select a single discrete control sequence
204
205 for i = 1:length(sys.seqv)
206    Sim.constraints = [Sim.constraints, optimvar.v(i)+~optimvar.vseqpos(i) <=
           1.5];      % Restrict control sequences
207 end
208 %%
209 %
210 %
211 % *Construct cost function*
212
213 Sim.options    = sdpsettings('solver', 'gurobi');                % Optimiser
       settings
214 Sim.input_var  = {optimvar.x, optimvar.ref, optimvar.vseqpos};    % Input
       variables
215 Sim.output_var = optimvar.v(:)';                                  % Output
       variables
216
217 Sim.objective_out = 0;              % Initialise objective function
218
219 for i = 1:Sim.Np                    % Loop over prediction horizon
220    for j = 1:height(sys.seqm)       % Loop over all possible mode sequences
221       for k = 1:height(sys.seqv)    % Loop over all possible control sequences
222          if Sim.lineariseMIQP
223             Sim.objective_out = Sim.objective_out + optimvar.tj{(j-1)*height(
                    sys.seqv)+k}(i)*sys.Ps_tilde(1, j, k);
224          else
225             Sim.objective_out = Sim.objective_out + optimvar.t{j}(i)*sys.
                    Ps_tilde(1, j, k)*optimvar.v(k);
226          end
227       end
228    end
229 end
230
231 %%
232 %
233 %
234 % *Construct controller object*
235
236 Sim.controller = optimizer(Sim.constraints,Sim.objective_out,Sim.options,Sim.
       input_var,Sim.output_var);
237 %%
238 %
```

```
239 %% Run simulation
240 %
241 %
242 % *Start loop for timing purposes*
243
244 % for z = 1:10; tic
245 %
246 % *Initialise control loop*
247
248 lpvar.V_imp       = zeros(Sim.kmax+1, 1);     % Implemented binary inputs
249 lpvar.output      = zeros(Sim.kmax+1, 1);     % System output
250 lpvar.mode        = zeros(Sim.kmax+1, 1);     % System mode
251 lpvar.diagnostics = zeros(Sim.kmax+1, 1);     % Yalmip diagnostics
252 lpvar.output_det  = zeros(Sim.kmax+1, 1);     % Output determined by which
        state
253
254 rng(1);
255 lpvar.num = rand(Sim.kmax+1,1);
256
257 lpvar.state(1,:)      = 3*zeros(1, sys.nx);                          % Initial
        state
258 lpvar.output(1)       = mpMulti(sys.C(:,:,1), lpvar.state(1,:)');    % Initial
        output
259 lpvar.mode(1)         = 1;                                           % Initial
        mode
260 lpvar.output_det(1)   = 0;                                           % Which
        state determines output
261
262 lpvar.state_FSM = zeros(sys.nL*Sim.maxnummode, Sim.kmax+1);               %
        FSM state
263 lpvar.state_FSM(Sim.maxnummode*lpvar.mode(1)-(Sim.maxnummode-1),1)  = 1;   %
        First FSM state
264
265 Sim.c_ref = -2;
266 Sim.ref       = Sim.rho_r*(0:Sim.kmax+sys.ny*Sim.Np)'+Sim.c_ref;  % Reference
        output signal
267
268 lpvar.differenceY(1,:) = lpvar.output(1,:) - Sim.ref(1);          % Difference
        output and reference signals
269
270 Sim.exp_conv_steps = ceil((lpvar.output(1)-Sim.c_ref)/(Sim.rho_r-growth.
        rho_dubar_Np));
271
272 % No errors have yet occured
273 lpvar.warnvar = false;
274
275 showprogress = zeros(1, 10);
276 progressval = linspace(0, 1, length(showprogress)+1);
277 %%
278 %
279 %
280 % *Start loop*
281
282 for k = 2:Sim.kmax+1
283
284    % determine admissible discrete control inputs as a function of the FSM
285    % state
```

```
286     lpvar.vseqposloop = sys.adm_cont_seq{lpvar.state_FSM(:,k-1)==1}';
287
288     % Run optimiser
289     lpvar.nameController = 'Sim.controller';   % Choose controller
290     eval(['[U, lpvar.diagnostics(k)] = ', lpvar.nameController, '{transpose(
            lpvar.state(k-1,:)), Sim.ref(k:k+Sim.Np-1), lpvar.vseqposloop};']);
291
292     % Check for errors
293     checkError(lpvar.diagnostics(k), lpvar.warnvar);
294
295     % Read output
296     lpvar.V_imp(k)    = sys.seqv(abs(U-1)<1e-5, 1);   % Save implemented binary
            control input
297
298     % Advance to next mode based on probabilities in sys.Ps and v(k)
299     for i = 1:sys.nL
300         if lpvar.num(k) < sum(sys.Ps(lpvar.mode(k-1), 1:i, lpvar.V_imp(k)))
301             lpvar.mode(k) = i;
302             break
303         end
304     end
305
306     % Advance FSM state
307     for i = 1:sys.nL
308         if lpvar.mode(k) == i
309             if ismember(find(lpvar.state_FSM(:,k-1)), 1+(i-1)*Sim.maxnummode:Sim.
                    maxnummode*i-1)
310                 lpvar.state_FSM(find(lpvar.state_FSM(:,k-1))+1,k) = 1;
311             elseif find(lpvar.state_FSM(:,k-1)) == Sim.maxnummode*i
312                 lpvar.state_FSM(:,k) = lpvar.state_FSM(:,k-1);
313             else
314                 lpvar.state_FSM(1+(i-1)*Sim.maxnummode, k) = 1;
315             end
316         end
317     end
318
319
320     % Advance system state
321     lpvar.state(k, :) = mpMulti(sys.A(:,:,lpvar.mode(k)), lpvar.state(k-1,:)');
322     lpvar.output(k,:) = mpMulti(sys.C(:,:,lpvar.mode(k)), lpvar.state(k,:)');
323
324     % Check which state determines output
325     [~, ind_os] = max(sys.C(:,:,lpvar.mode(k))+lpvar.state(k,:));
326     lpvar.output_det(k) = ind_os;
327
328     % Define error signals
329     lpvar.differenceY(k,:) = lpvar.output(k,:) - Sim.ref(k);     % Difference
            output and reference signals
330
331     % Display progress
332     for i = length(showprogress):-1:1
333         if k/(Sim.kmax+1) >= progressval(i+1) && showprogress(i) == 0
334             disp("progress: " + num2str(progressval(i+1)*100) + "%")
335             showprogress(i) = 1;
336             break
337         end
338     end
```

```
339
340 end
341 %%
342 %
343 %
344 % *Check which modes are not according to plan*
345
346 lpvar.logical_mode = lpvar.V_imp(1:end)==lpvar.mode(1:end);
347
348 lpvar.xvec_c  = nonzeros(double(lpvar.logical_mode).*(0:Sim.kmax)');     %
    States during correct modes
349 lpvar.mode_c  = lpvar.mode(lpvar.logical_mode);                           %
    Correct modes
350
351 lpvar.xvec_nc = double(~lpvar.logical_mode).*(0:Sim.kmax)';
352 lpvar.xvec_nc = [lpvar.xvec_nc(1); nonzeros(lpvar.xvec_nc(2:end))];      %
    States during incorrect modes
353 lpvar.mode_nc = lpvar.mode(~lpvar.logical_mode);                          %
    Incorrect modes
354 %%
355 %
356 %
357 % *Value objective functions*
358
359 lpvar.val_obj_out = sum(max(lpvar.differenceY, 0));          % Value output
    objective function
360 lpvar.val_obj_in  = 0;                                       % Value input
    objective function
361 lpvar.val_obj     = lpvar.val_obj_in + lpvar.val_obj_out;    % Value combined
    objective function
362 %%
363 %
364 %
365 % *Store runtime information*
366
367 % lpvar.runtime(z) = toc
368 % end
369 %%
370 %
371 %% Display results
372 %
373 %
374 % *Datatable*
375
376 VarNames  = {'Event step', 'Mode', 'Control input v(k)', 'Output', 'Reference',
     'Output determined by state #'};
377 lpvar.dataTable = table(transpose(1:Sim.kmax), lpvar.mode(2:Sim.kmax+1),  lpvar
    .V_imp(2:Sim.kmax+1), lpvar.output(2:Sim.kmax+1), Sim.ref(2:Sim.kmax+1),
    lpvar.output_det(2:Sim.kmax+1), 'VariableNames', VarNames);
378 disp(lpvar.dataTable)
379 %%
380 %
381 %
382 % *System signals*
383
384 % ---- Output vs reference ----
385 figure;
```

```
386 ax1 = subplot(4,1,1); hold on
387 % Plot growth rates
388 v = [0 0; Sim.kmax growth.rho_dubar_Np_con*Sim.kmax; Sim.kmax growth.rho_bar*
    Sim.kmax];
389 f = [1 2 3];
390 patch('Faces',f,'Vertices',v,'FaceColor',other.colourvec{4},'FaceAlpha',.5, '
    EdgeColor', 'none');
391 % Plot data
392 stairs(0:Sim.kmax, Sim.ref(1:Sim.kmax+1), 'Color', other.colourvec{5})
393 stairs(0:Sim.kmax, lpvar.output(1:Sim.kmax+1), 'Color', other.colourvec{1})
394 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
    {2}, 'LineStyle', '--')
395 % Other settings
396 legend('Region of achievable growth rates', "$r(k)$ with $\rho_\mathrm{r}=$ " +
     num2str(Sim.rho_r, 2), ...
397    '$y(k)$', 'Expected first convergence', 'Interpreter', 'latex', 'Location',
       'northwest')
398 xlim([0 Sim.kmax+1]); grid on
399 title('Output $y(k)$ vs reference $r(k)$')
400 ylabel('Time')
401 hold off
402
403 % ---- Difference signals ----
404 ax2 = subplot(4,1,2); hold on
405 plot(0:Sim.kmax, lpvar.differenceY(1:Sim.kmax+1), '-o', 'Color', other.
    colourvec{1})
406 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
    {2}, 'LineStyle', '--')
407 legend('$y(k)-r(k)$', 'Interpreter', 'latex')
408 xlim([0 Sim.kmax+1]); grid on
409 title('Difference signals')
410 ylabel('Time')
411 hold off
412
413 % ---- Mode sequence ----
414 ax3 = subplot(4,1,3); hold on
415 plot(lpvar.xvec_c, lpvar.mode_c, '*', 'Color', other.colourvec{1})
416 plot(lpvar.xvec_nc, lpvar.mode_nc, '*', 'Color', other.colourvec{2})
417 line([Sim.exp_conv_steps Sim.exp_conv_steps], [0 sys.nL+1], 'Color', other.
    colourvec{2}, 'LineStyle', '--')
418 legend('$\ell(k) = v(k)$', '$\ell(k) \neq v(k)$', 'Interpreter', 'latex')
419 set(gca, 'YTick', 0:(sys.nL+1))
420 ylim([0 sys.nL+1]);
421 xlim([0 Sim.kmax+1]); grid on
422 title('System mode $\ell(k)$')
423 ylabel('System mode $\ell(k)$')
424 hold off
425
426 % ---- State differences ----
427 ax4 = subplot(4,1,4); hold on
428 p = plot(0:Sim.kmax, lpvar.state(1:Sim.kmax+1,:)-Sim.ref(1:Sim.kmax+1),'*');
429 if sys.nx == 3
430    p(1).Color = other.colourvec{1}; p(2).Color = other.colourvec{5}; p(3).Color
        = other.colourvec{3};
431 end
432 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
    {2}, 'LineStyle', '--')
```

```
433 legend('$x_1(k) - r(k)$', '$x_2(k) - r(k)$', '$x_3(k) - r(k)$', 'Interpreter',
        'latex')
434 title('Difference between states $x_i(k)$ and reference signal $r(k)$')
435 xlabel('Event step $k$');
436 ylabel('Time')
437 xlim([0 Sim.kmax+1]); grid on
438 hold off
439
440 % ---- Link axes ----
441 set(gca, 'TickLabelInterpreter','latex')
442 linkaxes([ax1, ax2, ax3, ax4], 'x')
443 %%
444 %
445 %
446 % *Clear meaningless variables*
447
448 clear ax1 ax2 ax3 ax4 vec a growth_append v f check_vec i j k p showprogress
        progressval num_conseq_mode U VarNames vseqpos last_modes ind_os
        val_conseq_mode Nsim maxit offsetRange
449 %%
450 %
```

A-2-2 `Case2.mlx`

The following script is a MATLAB live script.

```
1  %% Case 2: Hybrid control of a mode-constrained deterministic system
2  % *File info:*
3  %%
4  % * Name:     |Case2.mlx|
5  % * Author:   Bart de Jong
6  % * Date:      June, 2022
7  %%
8  % *Prerequisite software*
9  %%
10 % * MATLAB    (Tested version: R2022a)
11 % * YALMIP     (Tested version: R20200116)
12 % * GUROBI    (Tested version: gurobi901)
13 %%
14 % *Prerequisite custom function files*
15 %%
16 % * |allSequences.m|
17 % * |checkError.m|
18 % * |findOffset.m|
19 % * |generateSemigroup.m|
20 % * |generateSupport.m|
21 % * |generateSystem.m|
22 % * |growthRate.m|
23 % * |modeConstraints.m|
24 % * |mpAdd.m|
25 % * |mpMulti.m|
26 % * |predictionModel.m|
27 %%
28 % *Prerequisite toolbox*
29 %%
30 % * Max-Plus Algebra Toolbox for Matlab - Copyright (C) 2016 Jaroslaw Stanczyk
```

```
% * Used for |mp_mcm.m| and supporting files
%%
%
%% Initialisation and settings
%
%
% *Clear workspace*

clearvars
%%
%
%
% *Preliminaries and general settings*

% Max-plus zero and one elements
other.eps       = -inf;
other.e         = 0;

% Colour vector for figure elements
other.colourvec = {'#00A6D6', '#E03C31', '#FFB81C', '#6CC24A', '#EC6842'};

% Latex interpreter
set(0, 'defaultAxesTickLabelInterpreter', 'latex');
set(0, 'defaultLegendInterpreter'       , 'latex');
set(0, 'defaultTextInterpreter'         , 'latex');
%%
%
%
% *Simulation settings*

Sim.Np            = 3;        % Prediction horizon
Sim.rho_r         = 2.2;      % Growth rate reference signal (constant in
    reference signal)
Sim.kmax          = 200;      % Number of event steps in the simulation
Sim.maxnummode    = 3;        % Maximum number of same mode in a row
Sim.lineariseMIQP = false;    % Turn MIQP into MILP

Sim.nu_max        = 10;       % Maximum difference between input and reference
Sim.beta          = 1e-2;     % Tuning parameter in cost function (beta > 0)
%%
%
%% Construct SMPL system
%
%
% *Define dimensions of the SMPL system*

sys.nx = 3;       % Number of states
sys.nu = 1;       % Number of contimous inputs (u(k))
sys.ny = 1;       % Number of outputs
sys.nL = 3;       % Number of modes
sys.nv = sys.nL;  % Number of discrete inputs (v(k)) - v(k) is a scalar, not a
    vector
%%
%
%
% *Generate SMPL system with no switching stochastics*

```

```
86  rng(9)   % Seed for random number generator
87  [sys.A, sys.B, sys.C, ~] = generateSystem(sys.nx, sys.nu, sys.ny, sys.nL);
        % Construct nonautonomous system
88
89  % Define switching probabilities {0, 1}
90  sys.Ps = zeros(sys.nL, sys.nL, sys.nv);
91  for i = 1:sys.nL
92      sys.Ps(:, i, i) = 1;
93  end
94  %%
95  %
96  %% Construct prediction model
97  %
98  %
99  % *Find all possible mode sequences over the prediction horizon and put them
100 % in a matrix*
101
102 [sys.seqm, sys.seqv, sys.Ps_tilde] = allSequences(sys.nL, Sim.Np, sys.Ps);
103 %%
104 %
105 %
106 % *Find all possible sequences that do not violate the mode-constraint*
107
108 sys.violate_con = sum(diff(sys.seqm,[],2)==0, 2) > Sim.maxnummode-1;
109 sys.seqm_con = sys.seqm(~sys.violate_con, :);
110 sys.seqv_con = sys.seqv(~sys.violate_con, :);
111 %%
112 %
113 %
114 % *Calculate* $\tilde{A}$*,* $\tilde{B}$*,* $\tilde{C}$ *and* $\tilde{D}$ *
        matrices
115 % for all possible mode sequences*
116
117 [sys.A_tilde, sys.C_tilde, sys.B_tilde, sys.D_tilde] = predictionModel(sys.A,
        sys.B, sys.C, sys.seqm);
118 %%
119 %
120 %% Calculate deterministic (constrained) growth rate
121 %
122 %
123 % *Calculate values*
124
125 clear growth
126 growth.Np_GR = 1*Sim.Np;   % Maximum period of cyclic mode sequences
127 [growth.seqm_GR, ~, growth.Ps_tilde_GR] = allSequences(sys.nL, growth.Np_GR,
        sys.Ps);
128
129 [growth_append] = growthRate(sys.A, sys.C, growth.seqm_GR, growth.Ps_tilde_GR,
        "maxnummode", Sim.maxnummode);
130 growth = cell2struct([struct2cell(growth);struct2cell(growth_append)],[
        fieldnames(growth);fieldnames(growth_append)]);
131 %%
132 %
133 %
134 % *Plot results*
135
136 figure; boxplot([growth.rho_dbar_Np; growth.rho_dbar_Np_con], ...
```

```
137     [repmat("Unconstrained deterministic system", [size(growth.rho_dbar_Np, 1),
            1]); ...
138     repmat("Constrained deterministic system", [size(growth.rho_dbar_Np_con, 1),
             1])], 'Whisker', 2);
139
140 a = get(get(gca,'children'),'children');   % Get the handles of all the objects
141 set(a(1:4), 'Color', other.colourvec{5});
142 set(a(5:6), 'Color', other.colourvec{1});
143
144 ylabel('Growth rate $\overline{\overline{\rho}}_{N_\mathrm{p}}$', 'Interpreter'
        , 'latex');
145 grid minor
146 set(gca, 'TickLabelInterpreter','latex')
147 %%
148 %
149 %% Find admissible control inputs based on FSM state
150 %
151
152 [sys.adm_cont, sys.adm_cont_seq] = modeConstraints(Sim.maxnummode, sys.nv, sys.
        seqv);
153 %%
154 %% Find lower bound on reference signal for required probability of performance
155 %
156
157 % Check for stochastics
158 Sim.ProbOfPerf = 0.95;
159 if any(sys.Ps(sys.Ps~=1)>0)
160     Nsim = 200;           % Growth rate simulation horizon
161     maxit = 3.00e4;       % Number of samples
162     offsetRange = 0.2;    % Maximum absolute offset
163     Sim.offsetPoP = findOffset(sys.A, sys.C, sys.Ps, Nsim, growth.
            rho_dubar_Np_con, growth.rho_dbar_Np, growth.seqm, maxit, Sim.ProbOfPerf,
             offsetRange);
164 else
165     Sim.offsetPoP = 0;
166 end
167 fprintf("- Offset necessary to obtain a probability of performance of %.2f with
         given accuracy and confidence: %.2f\n" + ...
168     "- Minimum expected constrained growth rate including offset: %.2f", Sim.
            ProbOfPerf, Sim.offsetPoP, growth.rho_dubar_Np_con + Sim.offsetPoP);
169 %%
170 %
171 %% Setup MPC algorithm
172 %
173 %
174 % *Construct set of constraints*
175
176 Sim.constraints = [];
177
178 optimvar.t       = sdpvar(repmat(Sim.Np, 1, height(sys.seqm)), ones(1, height(
        sys.seqm)));   % Construct optimisation variables t
179 optimvar.x       = sdpvar(sys.nx, 1);
                                                    % Initial state
180 optimvar.u       = sdpvar(Sim.Np+1, 1);
                                                  % Construct optimisation
        variables u (input)
```

```
181 optimvar.ref     = sdpvar(Sim.Np, 1);
                                                    % Reference signal over
    prediction horizon
182 optimvar.v       = binvar(height(sys.seqv), 1);
                                                % Variable denoting with sequence of v(
    k) is considered
183 optimvar.vseqpos = binvar(height(sys.seqv), 1);
                                                % Vector denoting which mode sequence
    is allowed
184
185 if Sim.lineariseMIQP
186    optimvar.tj = sdpvar(repmat(Sim.Np, 1, height(sys.seqm)*height(sys.seqv)),
       ones(1, height(sys.seqm)*height(sys.seqv)));
187    Sim.Mt = growth.rho_bar*(Sim.kmax+1);     % Upper bound on maximum of system'
       s output for y(0)<=0
188    Sim.mt = 0;                              % Lower bound on minimum of system's
       output for y(0)>=0
189 end
190
191 for i = 1:sys.ny*Sim.Np              % Loop over prediction horizon for all
    outputs
192     for j = 1:height(sys.seqm)  % Loop over all possible mode sequences
193         for k = 1:sys.nx          % Loop over all states
194             Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= sys.C_tilde
                (i, k, j) + optimvar.x(k) - optimvar.ref(i)];   %#ok<*AGROW> %
                Autonomous system dynamics
195         end
196         for k = 1:Sim.Np        % Loop over prediction horizon
197             Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= sys.D_tilde
                (i, k, j) + optimvar.u(k) - optimvar.ref(i)];   % Controlled
                system dynamics
198         end
199         if Sim.lineariseMIQP
200            for k = 1:height(sys.seqv)
201               Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                  seqv)+k}(i) <= Sim.Mt*optimvar.v(k)];
202               Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                  seqv)+k}(i) >= Sim.mt*optimvar.v(k)];
203               Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                  seqv)+k}(i) <= optimvar.t{j}(i)-Sim.mt*(1-optimvar.v(k))];
204               Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.
                  seqv)+k}(i) >= optimvar.t{j}(i)-Sim.Mt*(1-optimvar.v(k))];
205            end
206         end
207         Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= 0];
                                                % Disregard cost of y(k) if y(k
            )<=r(k)
208     end
209 end
210 for i = 1:sys.nu*Sim.Np              % Loop over prediction horizon
211     Sim.constraints = [Sim.constraints,  optimvar.u(i) >= 0];
212     Sim.constraints = [Sim.constraints,  optimvar.u(i+1) - optimvar.u(i)  >=
        0];                                  % input should be nondecreasing
213     Sim.constraints = [Sim.constraints,  optimvar.u(i) - optimvar.ref(i) <= Sim
        .nu_max];                              % Input should be close to
        reference signal
```

```
214     Sim.constraints = [Sim.constraints,  optimvar.u(i) - optimvar.ref(i) >= -
            Sim.nu_max];                                % Input should be close to
            reference signal
215 end
216
217 Sim.constraints = [Sim.constraints, sum(optimvar.v) == 1];
                                                        % Select a single discrete
    control sequence
218
219 for i = 1:length(sys.seqv)      % Loop over all possible control sequences
220     Sim.constraints = [Sim.constraints, optimvar.v(i)+~optimvar.vseqpos(i) <=
            1.5];                                       % Restrict control sequences
221 end
222 %%
223 %
224 %
225 % *Construct cost function*
226
227 Sim.options    = sdpsettings('solver', 'gurobi');   % Optimiser settings
228 Sim.input_var  = {optimvar.x, optimvar.ref, optimvar.vseqpos};
    % Input variables
229 Sim.output_var = [optimvar.u; optimvar.v];                              % Output
    variables
230
231 Sim.objective_in  = 0;
232 Sim.objective_out = 0;
233
234 % J_out
235 for i = 1:Sim.Np                    % Loop over prediction horizon
236     for j = 1:height(sys.seqm)      % Loop over all possible mode sequences
237         for k = 1:height(sys.seqv)
238             if Sim.lineariseMIQP
239                 Sim.objective_out = Sim.objective_out + optimvar.tj{(j-1)*height(
                        sys.seqv)+k}(i)*sys.Ps_tilde(1, j, k);
240             else
241                 Sim.objective_out = Sim.objective_out + optimvar.t{j}(i)*sys.
                        Ps_tilde(1, j, k)*optimvar.v(k);
242             end
243         end
244     end
245 end
246
247 % J_in
248 for i = 1:Sim.Np                    % Loop over prediction horizon
249     Sim.objective_in = Sim.objective_in - Sim.beta*optimvar.u(i);
250     for j = 1:height(sys.seqv)      % Loop over all v(k) sequences
251     end
252 end
253
254 %%
255 %
256 %
257 % *Construct controller*
258
259 Sim.controller = optimizer(Sim.constraints,Sim.objective_out+Sim.objective_in,
    Sim.options,Sim.input_var,Sim.output_var);
260 %%
```

```
%
%% Run simulation
%
%
% *Start loop for timing purposes*

% for z = 1:10; tic
%
% *Initialise control loop*

lpvar.U_imp       = zeros(Sim.kmax+1, 1);      % Implemented inputs
lpvar.V_imp       = zeros(Sim.kmax+1, 1);      % Implemented binary inputs
lpvar.output      = zeros(Sim.kmax+1, 1);      % System output
lpvar.mode        = zeros(Sim.kmax+1, 1);      % System mode
lpvar.diagnostics = zeros(Sim.kmax+1, 1);      % Yalmip diagnostics
lpvar.output_det  = zeros(Sim.kmax+1, 1);      % Output determined by which
    state
lpvar.state_det   = zeros(Sim.kmax+1, sys.nx);    % State determined by output
    (logical)

rng(1);
lpvar.num = rand(Sim.kmax+1,1);

lpvar.state(1,:)      = zeros(1, sys.nx);      % Initial state
lpvar.output(1)       = mpMulti(sys.C(:,:,1), lpvar.state(1,:)');
lpvar.mode(1)         = 1;                     % Initial mode
lpvar.output_det(1)   = 0;                     % Which state determines output
lpvar.state_det(1,:)  = zeros(1, sys.nx);      % State determined by output

lpvar.state_FSM = zeros(sys.nL*Sim.maxnummode, Sim.kmax+1);
lpvar.state_FSM(Sim.maxnummode*lpvar.mode(1)-(Sim.maxnummode-1),1)  = 1;
             % First FSM state

Sim.c_ref = -2;
Sim.ref   = Sim.rho_r*(0:Sim.kmax+sys.ny*Sim.Np)'+Sim.c_ref;   % Reference
    output signal

lpvar.differenceY(1,:) = lpvar.output(1,:) - Sim.ref(1);    % Difference output
     and reference signals
lpvar.differenceU(1,:) = 0 - Sim.ref(1);       % Difference input  and
    reference signals

Sim.exp_conv_steps = ceil((lpvar.output(1)-Sim.c_ref)/(Sim.rho_r-growth.
    rho_dubar_Np_con));

% No errors have yet occured
lpvar.warnvar = false;

showprogress = zeros(1, 10);
progressval = linspace(0, 1, length(showprogress)+1);
%%
%
%
% *Start loop*

for k = 2:Sim.kmax+1

```

```
311     % determine admissible discrete control inputs as a function of the FSM
312     % state
313     lpvar.vseqposloop = sys.adm_cont_seq{lpvar.state_FSM(:,k-1)==1}';
314
315     % Run optimiser
316     lpvar.nameController = 'Sim.controller';
317     eval(['[U, lpvar.diagnostics(k)] = ', lpvar.nameController, '{transpose(
            lpvar.state(k-1,:)), Sim.ref(k:k+Sim.Np-1), lpvar.vseqposloop};']); %#ok
            <EVLEQ>
318
319     % Check for errors
320     checkError(lpvar.diagnostics(k), lpvar.warnvar);
321
322     % Read output
323     lpvar.U_imp(k)    = U(1);                                % Save implemented
            control input
324     lpvar.V_imp(k)    = sys.seqv(find(U((Sim.Np+1)+1:end)), 1);   %#ok<FNDSB> %
            Save implemented binary control input
325
326     % Advance to next mode based on probabilities in Ps and v(k)
327     for i = 1:sys.nL
328         if lpvar.num(k) < sum(sys.Ps(lpvar.mode(k-1), 1:i, lpvar.V_imp(k)))
329             lpvar.mode(k) = i;
330             break
331         end
332     end
333
334     % Advance FSM state
335     for i = 1:sys.nL
336         if lpvar.mode(k) == i
337             if ismember(find(lpvar.state_FSM(:,k-1)), 1+(i-1)*Sim.maxnummode:
                    Sim.maxnummode*i-1)
338                 lpvar.state_FSM(find(lpvar.state_FSM(:,k-1))+1,k) = 1;
339             elseif find(lpvar.state_FSM(:,k-1)) == Sim.maxnummode*i
340                 lpvar.state_FSM(:,k) = lpvar.state_FSM(:,k-1);
341             else
342                 lpvar.state_FSM(1+(i-1)*Sim.maxnummode, k) = 1;
343             end
344         end
345     end
346
347     % Update state
348     lpvar.state(k, :) = mpAdd(mpMulti(sys.A(:,:,lpvar.mode(k)), lpvar.state(k
            -1,:)'), mpMulti(sys.B(:,:,lpvar.mode(k)), lpvar.U_imp(k)));
349     lpvar.output(k,:) = mpMulti(sys.C(:,:,lpvar.mode(k)), lpvar.state(k,:)');
350
351     % State determined by input?
352     lpvar.state_det(k,:) = mpMulti(sys.B(:,:,lpvar.mode(k)), lpvar.U_imp(k)) >=
            mpMulti(sys.A(:,:,lpvar.mode(k)), lpvar.state(k-1,:)');
353
354     % Check which state determines output
355     [~, ind_os] = max(sys.C(:,:,lpvar.mode(k))+lpvar.state(k,:));
356     lpvar.output_det(k) = ind_os;
357
358     % Define error signals
359     lpvar.differenceY(k,:) = lpvar.output(k,:) - Sim.ref(k);    % Difference
            output and reference signals
```

```
360     lpvar.differenceU(k,:) = lpvar.U_imp(k) - Sim.ref(k);       % Difference
            input  and reference signals
361
362     % Display progress
363     for i = length(showprogress):-1:1
364         if k/(Sim.kmax+1) >= progressval(i+1) && showprogress(i) == 0
365             disp("progress: " + num2str(progressval(i+1)*100) + "%")
366             showprogress(i) = 1;
367             break
368         end
369     end
370
371 end
372 %%
373 %
374 %
375 % *Check which modes are not according to plan*
376
377 lpvar.logical_mode = lpvar.V_imp(1:end)==lpvar.mode(1:end);
378
379 lpvar.xvec_c  = nonzeros(double(lpvar.logical_mode).*(0:Sim.kmax)');
380 lpvar.mode_c  = lpvar.mode(lpvar.logical_mode);
381
382 lpvar.xvec_nc = double(~lpvar.logical_mode).*(0:Sim.kmax)';
383 lpvar.xvec_nc = [lpvar.xvec_nc(1); nonzeros(lpvar.xvec_nc(2:end))];
384 lpvar.mode_nc = lpvar.mode(~lpvar.logical_mode);
385 %%
386 %
387 %
388 % *Value objective functions*
389
390 lpvar.val_obj_out = sum(max(lpvar.differenceY, 0));
391 lpvar.val_obj_in  = Sim.beta*sum(-lpvar.U_imp(1:end-1));
392 lpvar.val_obj     = lpvar.val_obj_in + lpvar.val_obj_out;
393 %%
394 %
395 %
396 % *Store runtime information*
397
398 % lpvar.runtime(z) = toc
399 % end
400 %%
401 %
402 %% Display results
403 %
404 %
405 % *Data table*
406
407 VarNames = {'Event step', 'Mode', 'Control input u(k)','Control input v(k)', '
        Output', 'Reference', 'Output determined by state #', 'Is state determined
        by input?'};
408 lpvar.dataTable = table(transpose(1:Sim.kmax), lpvar.mode(2:Sim.kmax+1), lpvar.
        U_imp(2:Sim.kmax+1), lpvar.V_imp(2:Sim.kmax+1), lpvar.output(2:Sim.kmax+1),
        Sim.ref(2:Sim.kmax+1), lpvar.output_det(2:Sim.kmax+1),mat2cell(lpvar.
        state_det(2:Sim.kmax+1,:), ones(1, Sim.kmax), sys.nx), 'VariableNames',
        VarNames);
409 disp(lpvar.dataTable)
```

```
410 %%
411 %
412 %
413 % *System signals*
414
415 % ---- Output vs reference ----
416 figure;
417 ax1 = subplot(4,1,1); hold on
418 % Plot growth rates
419 v = [0 0; Sim.kmax growth.rho_dubar_Np_con*Sim.kmax; Sim.kmax growth.rho_bar*
    Sim.kmax];
420 f = [1 2 3];
421 patch('Faces',f,'Vertices',v,'FaceColor',other.colourvec{4},'FaceAlpha',.5, '
    EdgeColor', 'none');
422 % Plot data
423 stairs(0:Sim.kmax, Sim.ref(1:Sim.kmax+1), 'Color', other.colourvec{5})
424 stairs(0:Sim.kmax, lpvar.output(1:Sim.kmax+1), 'Color', other.colourvec{1})
425 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
    {2}, 'LineStyle', '--')
426 % Other settings
427 legend('Region of achievable growth rates', "$r(k)$ with $\rho_\mathrm{r}=$ " +
     num2str(Sim.rho_r, 2), ...
428     '$y(k)$', 'Expected first convergence', 'Interpreter', 'latex', 'Location',
         'northwest')
429 xlim([0 Sim.kmax+1]); grid on
430 title('Output $y(k)$ vs reference $r(k)$')
431 % xlabel('Event step $k$');
432 ylabel('Time')
433 hold off
434
435 % ---- Difference signals ----
436 ax2 = subplot(4,1,2); hold on
437 p = plot(0:Sim.kmax, [lpvar.differenceY(1:Sim.kmax+1) lpvar.differenceU(1:Sim.
    kmax+1)], '-o');
438 p(1).Color = other.colourvec{1}; p(2).Color = other.colourvec{5};
439 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
    {2}, 'LineStyle', '--')
440 legend('$y(k)-r(k)$', '$u(k)-r(k)$', 'Interpreter', 'latex')
441 xlim([0 Sim.kmax+1]); grid on
442 title('Difference signals')
443 % xlabel('Event step $k$');
444 ylabel('Time')
445 hold off
446
447 % ---- Mode sequence ----
448 ax3 = subplot(4,1,3); hold on
449 plot(lpvar.xvec_c, lpvar.mode_c, '*', 'Color', other.colourvec{1})
450 plot(lpvar.xvec_nc, lpvar.mode_nc, '*', 'Color', other.colourvec{2})
451 line([Sim.exp_conv_steps Sim.exp_conv_steps], [0 sys.nL+1], 'Color', other.
    colourvec{2}, 'LineStyle', '--')
452 legend('$\ell(k) = v(k)$', '$\ell(k) \neq v(k)$', 'Interpreter', 'latex')
453 set(gca, 'YTick', 0:(sys.nL+1))
454 ylim([0 sys.nL+1]); xlim([0 Sim.kmax+1]);
455 grid on
456 title('System mode $\ell(k)$')
457 % xlabel('Event step $k$');
458 ylabel('System mode $\ell(k)$')
```

```
459
460 % ---- State differences ----
461 ax4 = subplot(4,1,4); hold on
462 p = plot(0:Sim.kmax, lpvar.state(1:Sim.kmax+1,:)-Sim.ref(1:Sim.kmax+1),'*');
463 if sys.nx == 3
464    p(1).Color = other.colourvec{1}; p(2).Color = other.colourvec{5}; p(3).Color
          = other.colourvec{3};
465 end
466 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
      {2}, 'LineStyle', '--')
467 legend('$x_1(k) - r(k)$', '$x_2(k) - r(k)$', '$x_3(k) - r(k)$', 'Interpreter',
      'latex')
468 title('Difference between states $x_i(k)$ and reference signal $r(k)$')
469 xlabel('Event step $k$');
470 ylabel('Time')
471 xlim([0 Sim.kmax+1]); grid on
472 hold off
473
474 % ---- Link axes ----
475 set(gca, 'TickLabelInterpreter','latex')
476 linkaxes([ax1, ax2, ax3, ax4], 'x')
477 %%
478 %
479 %
480 % *Clear meaningless variables*
481
482 clear ax1 ax2 ax3 ax4 vec a growth_append v f check_vec i j k p showprogress
      progressval num_conseq_mode U VarNames vseqpos last_modes ind_os
      val_conseq_mode mat Nsim maxit offsetRange
483 %%
484 %
```

A-2-3 Case3.mlx

The following script is a MATLAB live script.

```
1  %% Case 3: Hybrid control of a mode-constrained stochastic system
2  % *File info:*
3  %%
4  % * Name:     |Case3.mlx|
5  % * Author:   Bart de Jong
6  % * Date:      June, 2022
7  %%
8  % *Prerequisite software*
9  %%
10 % * MATLAB    (Tested version: R2022a)
11 % * YALMIP     (Tested version: R20200116)
12 % * GUROBI    (Tested version: gurobi901)
13 %%
14 % *Prerequisite custom function files*
15 %%
16 % * |allSequences.m|
17 % * |checkError.m|
18 % * |findOffset.m|
19 % * |generateSemigroup.m|
20 % * |generateSupport.m|
```

```
% * |generateSystem.m|
% * |growthRate.m|
% * |modeConstraints.m|
% * |mpAdd.m|
% * |mpMulti.m|
% * |predictionModel.m|
%%
% *Prerequisite toolbox*
%%
% * Max-Plus Algebra Toolbox for Matlab - Copyright (C) 2016 Jaroslaw Stanczyk
% * Used for |mp_mcm.m| and supporting files
%%
%
%% Initialisation and settings
%
%
% *Clear workspace*

clearvars
%%
%
%
% *Preliminaries and general settings*

% Max-plus zero and one elements
other.eps       = -inf;
other.e         = 0;

% Colour vector for figure elements
other.colourvec = {'#00A6D6', '#E03C31', '#FFB81C', '#6CC24A', '#EC6842'};

% Latex interpreter
set(0, 'defaultAxesTickLabelInterpreter', 'latex');
set(0, 'defaultLegendInterpreter'       , 'latex');
set(0, 'defaultTextInterpreter'         , 'latex');
%%
%
%
% *Simulation settings*

Sim.Np            = 3;                % Prediction horizon
Sim.rho_r         = 2.2;              % Growth rate reference signal
Sim.kmax          = 200;              % Number of event steps in the
    simulation
Sim.maxnummode    = 3;                % Maximum number of same mode in a row
Sim.lineariseMIQP = true;             % Turn MIQP into MILP

Sim.nu_max        = 10;               % Maximum difference between input and
    reference
Sim.beta          = 1e-2;             % Tuning parameter in cost function (
    beta > 0)
Sim.alpha         = zeros(1, Sim.Np); % Weighting vector in cost function
%%
%
%% Construct SMPL system
%
%
```

```
% *Define dimensions of the SMPL system*

sys.nx = 3;        % Number of states
sys.nu = 1;        % Number of contimous inputs (u(k))
sys.ny = 1;        % Number of outputs
sys.nL = 3;        % Number of modes
sys.nv = sys.nL;  % Number of discrete inputs (v(k)) - v(k) is a scalar, not a
    vector
%%
%
%
% *Generate SMPL system with switching stochastics*

rng(9)
[sys.A, sys.B, sys.C, sys.Ps] = generateSystem(sys.nx, sys.nu, sys.ny, sys.nL);
%%
%
%% Construct prediction model
%
%
% *Find all possible mode sequences over the prediction horizon and put them
% in a matrix*

[sys.seqm, sys.seqv, sys.Ps_tilde] = allSequences(sys.nL, Sim.Np, sys.Ps);
%%
%
%
% *Find all possible sequences that do not violate the mode-constraint*

sys.violate_con = sum(diff(sys.seqm,[],2)==0, 2) > Sim.maxnummode-1;
sys.seqm_con = sys.seqm(~sys.violate_con, :);
sys.seqv_con = sys.seqv(~sys.violate_con, :);
%%
%
%
% *Calculate* $\tilde{A}$*,* $\tilde{B}$*,* $\tilde{C}$ *and* $\tilde{D}$ *
    matrices
% for all possible mode sequences*

[sys.A_tilde, sys.C_tilde, sys.B_tilde, sys.D_tilde] = predictionModel(sys.A,
    sys.B, sys.C, sys.seqm);
%%
%
%% Calculate stochastic (constrained) growth rate
%
%
% *Calculate values*

clear growth
growth.Np_GR = 2*Sim.Np;   % Maximum period of cyclic mode sequences
[growth.seqm_GR, ~, growth.Ps_tilde_GR] = allSequences(sys.nL, growth.Np_GR,
    sys.Ps);

[growth_append] = growthRate(sys.A, sys.C, growth.seqm_GR, growth.Ps_tilde_GR,
    "maxnummode", Sim.maxnummode);
growth = cell2struct([struct2cell(growth);struct2cell(growth_append)],[
    fieldnames(growth);fieldnames(growth_append)]);
```

```
126 %%
127 %
128 %
129 % *Plot results*
130
131 figure; boxplot([growth.rho_dbar_Np; growth.rho_dbar_Np_con], ...
132     [repmat("Unconstrained stochastic system", [size(growth.rho_dbar_Np, 1), 1])
        ; ...
133     repmat("Constrained stochastic system", [size(growth.rho_dbar_Np_con, 1),
        1])], 'Whisker', 2);
134
135 a = get(get(gca,'children'),'children');   % Get the handles of all the objects
136 set(a(1:4), 'Color', other.colourvec{5});
137 set(a(5:6), 'Color', other.colourvec{1});
138
139 ylabel('Expected growth rate $\overline{\overline{\rho}}_{N_\mathrm{p}}$', '
    Interpreter', 'latex');
140 grid minor
141 set(gca, 'TickLabelInterpreter','latex')
142 %%
143 %
144 %% Find admissible control inputs based on sFSM state
145 %
146
147 [sys.adm_cont, sys.adm_cont_seq] = modeConstraints(Sim.maxnummode, sys.nv, sys.
    seqv);
148 %%
149 %
150 %% Find lower bound on reference signal for required probability of performance
151 %
152
153 % Check for stochastics
154 Sim.ProbOfPerf = 0.95;
155 if any(sys.Ps(sys.Ps~=1)>0)
156     Nsim = 200;           % Growth rate simulation horizon
157     maxit = 3.00e4;       % Number of samples
158     offsetRange = 0.2;    % Maximum absolute offset
159     Sim.offsetPoP = findOffset(sys.A, sys.C, sys.Ps, Nsim, growth.
        rho_dubar_Np_con, growth.rho_dbar_Np, growth.seqm, maxit, Sim.ProbOfPerf,
         offsetRange);
160 else
161     Sim.offsetPoP = 0;
162 end
163 fprintf("- Offset necessary to obtain a probability of performance of %.2f with
     given accuracy and confidence: %.2f\n" + ...
164     "- Minimum expected constrained growth rate including offset: %.2f", Sim.
        ProbOfPerf, Sim.offsetPoP, growth.rho_dubar_Np_con + Sim.offsetPoP);
165 %%
166 %
167 %% Setup MPC algorithm
168 %
169 %
170 % *Construct set of constraints*
171
172 Sim.constraints = [];
173 % Construct optimisation variables t
```

```
optimvar.t       = sdpvar(repmat(Sim.Np, 1, height(sys.seqm)), ones(1, height(sys.seqm)));
optimvar.x       = sdpvar(sys.nx, 1);              % Initial state
optimvar.u       = sdpvar(Sim.Np+1, 1);            % Construct optimisation variables u (input)
optimvar.ref     = sdpvar(Sim.Np, 1);              % Reference signal over prediction horizon
optimvar.v       = binvar(height(sys.seqv), 1);    % Variable denoting the sequence of v(k)
optimvar.vseqpos = binvar(height(sys.seqv), 1);    % Vector denoting which mode sequence is allowed

if Sim.lineariseMIQP
   optimvar.tj      = sdpvar(repmat(Sim.Np, 1, height(sys.seqm)*height(sys.seqv)), ones(1, height(sys.seqm)*height(sys.seqv)));
   Sim.Mt       = growth.rho_bar*(Sim.kmax+1);    % Upper bound on maximum of system's output for y(0)<=0
   Sim.mt       = 0;                              % Lower bound on minimum of system's output for y(0)>=0
end

for i = 1:sys.ny*Sim.Np              % Loop over prediction horizon for all outputs
   for j = 1:height(sys.seqm)  % Loop over all possible mode sequences
      for k = 1:sys.nx         % Loop over all states
         % Autonomous system dynamics
         Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= sys.C_tilde(i, k, j) + optimvar.x(k) - optimvar.ref(i)];
      end
      for k = 1:Sim.Np        % Loop over prediction horizon
         % Controlled system dynamics
         Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= sys.D_tilde(i, k, j) + optimvar.u(k) - optimvar.ref(i)];
      end
      if Sim.lineariseMIQP
         for k = 1:height(sys.seqv)
            Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.seqv)+k}(i) <= Sim.Mt*optimvar.v(k)];
            Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.seqv)+k}(i) >= Sim.mt*optimvar.v(k)];
            Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.seqv)+k}(i) <= optimvar.t{j}(i)-Sim.mt*(1-optimvar.v(k))];
            Sim.constraints = [Sim.constraints, optimvar.tj{(j-1)*height(sys.seqv)+k}(i) >= optimvar.t{j}(i)-Sim.Mt*(1-optimvar.v(k))];
         end
      end
      % Don't finish before reference
      Sim.constraints = [Sim.constraints, optimvar.t{j}(i) >= 0];       % Disregard cost of y(k) if y(k)<=r(k)
   end
end
for i = 1:sys.nu*Sim.Np              % Loop over prediction horizon
   % Input should be nonnegative, nondecreasing and close to reference signal
   Sim.constraints = [Sim.constraints,  optimvar.u(i) >= 0];
   Sim.constraints = [Sim.constraints,  optimvar.u(i+1) - optimvar.u(i)  >= 0];
   Sim.constraints = [Sim.constraints,  optimvar.u(i) - optimvar.ref(i) <= Sim.nu_max];
```

```
214     Sim.constraints = [Sim.constraints,  optimvar.u(i) - optimvar.ref(i) >= -Sim
            .nu_max];
215 end
216
217 Sim.constraints = [Sim.constraints, sum(optimvar.v) == 1]; % Select a single
        discrete control sequence
218
219 for i = 1:length(sys.seqv)  % Loop over all possible control sequences
220     Sim.constraints = [Sim.constraints, optimvar.v(i)+~optimvar.vseqpos(i) <=
            1.5]; % Restrict control sequences
221 end
222 %%
223 %
224 %
225 % *Construct cost function*
226
227 % Set controller options
228 Sim.options    = sdpsettings('solver', 'gurobi');   % Optimiser settings
229 Sim.input_var  = {optimvar.x, optimvar.ref, optimvar.vseqpos};
        % Input variables
230 Sim.output_var = [optimvar.u; optimvar.v];                              % Output
        variables
231
232 % Initialise objective functions
233 Sim.objective_in  = 0;
234 Sim.objective_out = 0;
235
236 % Check if probability of switching depends on previous mode
237 if sum(abs(diff(sys.Ps_tilde, 1)), 'all') ~= 0
238     sys.modedependent = true;
239     Sim.nL_loop = sys.nL;
240 else
241     sys.modedependent = false;
242     Sim.nL_loop = 1;
243 end
244
245 % Loop over all possible previous modes and make unique controller objects
246 for prev_mode = 1:Sim.nL_loop
247
248     for i = 1:Sim.Np                    % Loop over prediction horizon
249         for j = 1:height(sys.seqm)      % Loop over all possible mode sequences
250             if Sim.lineariseMIQP
251                 for k = 1:height(sys.seqv)
252                     Sim.objective_out = Sim.objective_out + optimvar.tj{(j-1)*height
                            (sys.seqv)+k}(i)*sys.Ps_tilde(prev_mode, j, k);
253                 end
254             else
255                 Sim.objective_out = Sim.objective_out + optimvar.t{j}(i)*reshape(
                        sys.Ps_tilde(prev_mode, j, :), ...
256                     [1 length(optimvar.v)])*optimvar.v;
257             end
258         end
259     end
260
261     for i = 1:Sim.Np                    % Loop over prediction horizon
262         Sim.objective_in = Sim.objective_in - Sim.beta*optimvar.u(i);
263         for j = 1:height(sys.seqv)      % Loop over all v(k) sequences
```

```
264             Sim.objective_in = Sim.objective_in + Sim.beta*optimvar.v(j)*Sim.alpha
                    (i)*sys.seqv(j,i);
265         end
266     end
267
268
269 %%
270 %
271 %
272 % *Construct controller per mode*
273
274     Sim.controller = optimizer(Sim.constraints, Sim.objective_out+Sim.
            objective_in, Sim.options, ...
275         Sim.input_var, Sim.output_var);
276
277     % Give controller a mode-specific name
278     Sim.nameController = ['controller' num2str(prev_mode)];
279     assignin('base', Sim.nameController, Sim.controller);
280 end
281
282 %%
283 %
284 %% Run simulation
285 %
286 %
287 % *Start loop for timing purposes*
288
289 % for z = 1:10; tic
290 %
291 % *Initialise control loop*
292
293 lpvar.U_imp       = zeros(Sim.kmax+1, 1);      % Implemented inputs
294 lpvar.V_imp       = zeros(Sim.kmax+1, 1);      % Implemented binary inputs
295 lpvar.mode        = zeros(Sim.kmax+1, 1);      % System mode
296 lpvar.diagnostics = zeros(Sim.kmax+1, 1);      % Yalmip diagnostics
297 lpvar.output_det  = zeros(Sim.kmax+1, 1);      % Output determined by which
        state
298 lpvar.state_det   = zeros(Sim.kmax+1, sys.nx);    % State determined by output
        (logical)
299 lpvar.state       = zeros(Sim.kmax+1, sys.nx);    % State vector
300 lpvar.output      = zeros(Sim.kmax+1, sys.ny);    % Output vector
301
302 rng(1);
303 lpvar.num = rand(Sim.kmax+1,1);
304
305 lpvar.state(1,:)      = zeros(1, sys.nx);      % Initial state
306 lpvar.output(1)       = mpMulti(sys.C(:,:,1), lpvar.state(1,:)');
307 lpvar.mode(1)         = 1;                  % Initial mode
308 lpvar.output_det(1)   = 0;                  % Which state determines output
309 lpvar.state_det(1,:)  = zeros(1, sys.nx);      % State determined by output
310
311 lpvar.state_FSM = zeros(sys.nL*Sim.maxnummode, Sim.kmax+1);
312 lpvar.state_FSM(Sim.maxnummode*lpvar.mode(1)-(Sim.maxnummode-1),1)  = 1;
            % First FSM state
313
314 Sim.c_ref = -2;
```

```
315 Sim.ref    = Sim.rho_r*(0:Sim.kmax+sys.ny*Sim.Np)'+Sim.c_ref;    % Reference
    output signal
316
317 lpvar.differenceY(1,:) = lpvar.output(1,:) - Sim.ref(1);     % Difference output
     and reference signals
318 lpvar.differenceU(1,:) = lpvar.U_imp(1) - Sim.ref(1);        % Difference input
     and reference signals
319
320 Sim.exp_conv_steps = ceil((lpvar.output(1)-Sim.c_ref)/(Sim.rho_r-growth.
    rho_dubar_Np));
321
322 % No errors have yet occured
323 lpvar.warnvar = false;
324
325 showprogress = zeros(1, 10);
326 progressval = linspace(0, 1, length(showprogress)+1);
327 %%
328 %
329 %
330 % *Start loop*
331
332 for k = 2:Sim.kmax+1
333
334    % determine admissible discrete control inputs as a function of the FSM
335    % state
336    lpvar.vseqposloop = sys.adm_cont_seq{lpvar.state_FSM(:,k-1)==1}';
337
338    % Check if due to stochastics the maximum number of consecutive modes
339    % is exceeded
340    if ~any(lpvar.vseqposloop)
341       warning('Maximum number of consecutive modes exceeded')
342       for j = 1:sys.nL
343          ind = strfind([lpvar.mode(k-min(Sim.maxnummode+1, k-1):k-1)' sys.seqv(
               i,:)], ...
344             j*ones(1,Sim.maxnummode+1));
345          if ~isempty(ind) && ind(1) == 1
346             for i = 1:length(sys.seqv)
347                if sys.seqv(i, 1) ~= j
348                   lpvar.vseqposloop(i) = 1;
349                end
350             end
351          end
352       end
353    end
354
355    % Run optimiser based on previous mode
356    if sys.modedependent
357       lpvar.nameController = ['controller' num2str(lpvar.mode(k-1))];
358    else
359       lpvar.nameController = 'controller1';
360    end
361    eval(['[U, lpvar.diagnostics(k)] = ', lpvar.nameController, ...
362       '{transpose(lpvar.state(k-1,:)), Sim.ref(k:k+Sim.Np-1), lpvar.vseqposloop
           };']);
363
364    % Check for errors
365    checkError(lpvar.diagnostics(k), lpvar.warnvar);
```

```
366
367     % Read output
368     lpvar.U_imp(k)     = U(1);                                  % Save implemented
            control input
369     lpvar.V_imp(k)     = sys.seqv(find(U((Sim.Np+1)+1:end)), 1);    % Save
            implemented control input
370
371     % Advance to next mode based on probabilities in Ps and v(k)
372     for i = 1:sys.nL
373        if lpvar.num(k) < sum(sys.Ps(lpvar.mode(k-1), 1:i, lpvar.V_imp(k)))
374           lpvar.mode(k) = i;
375           break
376        end
377     end
378
379     % Advance FSM state
380     for i = 1:sys.nL
381        if lpvar.mode(k) == i
382           if ismember(find(lpvar.state_FSM(:,k-1)), 1+(i-1)*Sim.maxnummode:Sim.
                   maxnummode*i-1)
383              lpvar.state_FSM(find(lpvar.state_FSM(:,k-1))+1,k) = 1;
384           elseif find(lpvar.state_FSM(:,k-1)) == Sim.maxnummode*i
385              lpvar.state_FSM(:,k) = lpvar.state_FSM(:,k-1);
386           else
387              lpvar.state_FSM(1+(i-1)*Sim.maxnummode, k) = 1;
388           end
389        end
390     end
391
392     % Update state
393     lpvar.state(k, :) = mpAdd(mpMulti(sys.A(:,:,lpvar.mode(k)), lpvar.state(k
            -1,:)'), ...
394        mpMulti(sys.B(:,:,lpvar.mode(k)), lpvar.U_imp(k)));
395     lpvar.output(k,:) = mpMulti(sys.C(:,:,lpvar.mode(k)), lpvar.state(k,:)');
396
397     % State determined by input?
398     lpvar.state_det(k,:) = mpMulti(sys.B(:,:,lpvar.mode(k)), lpvar.U_imp(k)) >=
            ...
399        mpMulti(sys.A(:,:,lpvar.mode(k)), lpvar.state(k-1,:)');
400
401     % Check which state determines output
402     [~, ind_os] = max(sys.C(:,:,lpvar.mode(k))+lpvar.state(k,:));
403     lpvar.output_det(k) = ind_os;
404
405     % Define error signals
406     lpvar.differenceY(k,:) = lpvar.output(k,:) - Sim.ref(k);     % Difference
            output and reference signals
407     lpvar.differenceU(k,:) = lpvar.U_imp(k) - Sim.ref(k);        % Difference
            input  and reference signals
408
409     % Display progress
410     for i = length(showprogress):-1:1
411        if k/(Sim.kmax+1) >= progressval(i+1) && showprogress(i) == 0
412           disp("progress: " + num2str(progressval(i+1)*100) + "%")
413           showprogress(i) = 1;
414           break
415        end
```

```
416     end
417
418 end
419 %%
420 %
421 %
422 % *Check which modes are not according to plan*
423
424 lpvar.logical_mode = lpvar.V_imp(1:end)==lpvar.mode(1:end);
425
426 lpvar.xvec_c  = nonzeros(double(lpvar.logical_mode).*(0:Sim.kmax)');
427 lpvar.mode_c  = lpvar.mode(lpvar.logical_mode);
428
429 lpvar.xvec_nc = double(~lpvar.logical_mode).*(0:Sim.kmax)';
430 lpvar.xvec_nc = [lpvar.xvec_nc(1); nonzeros(lpvar.xvec_nc(2:end))];
431 lpvar.mode_nc = lpvar.mode(~lpvar.logical_mode);
432 %%
433 %
434 %
435 % *Value objective functions*
436
437 lpvar.val_obj_out = sum(max(lpvar.differenceY, 0));
438 lpvar.val_obj_in  = Sim.beta*sum(-lpvar.U_imp(1:end-1));
439 lpvar.val_obj     = lpvar.val_obj_in + lpvar.val_obj_out;
440 %%
441 %
442 %
443 % *Store runtime information*
444
445 % lpvar.runtime(z) = toc
446 % end
447 %%
448 %
449 %% Display results
450 %
451 %
452 % *Data table*
453
454 VarNames = {'Event step', 'Mode', 'Control input u(k)','Control input v(k)', '
        Output', 'Reference', ...
455     'Output determined by state', 'State determined by input?'};
456 lpvar.dataTable = table(transpose(1:Sim.kmax), lpvar.mode(2:Sim.kmax+1), lpvar.
        U_imp(2:Sim.kmax+1), lpvar.V_imp(2:Sim.kmax+1), lpvar.output(2:Sim.kmax+1),
        ...
457     Sim.ref(2:Sim.kmax+1), lpvar.output_det(2:Sim.kmax+1),mat2cell(lpvar.
            state_det(2:Sim.kmax+1,:), ...
458     ones(1, Sim.kmax), sys.nx), 'VariableNames', VarNames);
459 disp(lpvar.dataTable)
460 %%
461 %
462 %
463 % *System signals*
464
465 % ---- Output vs reference ----
466 figure
467 ax1 = subplot(4,1,1); hold on
468 % Plot growth rates
```

```
469 v = [0 0; Sim.kmax growth.rho_dubar_Np_con*Sim.kmax; Sim.kmax growth.rho_bar*
        Sim.kmax];
470 f = [1 2 3];
471 patch('Faces',f,'Vertices',v,'FaceColor',other.colourvec{4},'FaceAlpha',.5, '
        EdgeColor', 'none');
472 % Plot data
473 stairs(0:Sim.kmax, Sim.ref(1:Sim.kmax+1), 'Color', other.colourvec{5})
474 stairs(0:Sim.kmax, lpvar.output(1:Sim.kmax+1), 'Color', other.colourvec{1})
475 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
        {2}, 'LineStyle', '--')
476 % Other settings
477 legend('Region of achievable growth rates', "$r(k)$ with $\rho_\mathrm{r}=$ "
        ...
478    + num2str(Sim.rho_r, 2), '$y(k)$', 'Expected first convergence', '
           Interpreter', ...
479    'latex', 'Location', 'northwest')
480 xlim([0 Sim.kmax+1]); grid on
481 title('Output $y(k)$ vs reference $r(k)$')
482 ylabel('Time')
483 hold off
484
485 % ---- Difference signals ----
486 ax2 = subplot(4,1,2); hold on
487 p = plot(0:Sim.kmax, [lpvar.differenceY(1:Sim.kmax+1) lpvar.differenceU(1:Sim.
        kmax+1)], '-o');
488 p(1).Color = other.colourvec{1}; p(2).Color = other.colourvec{5};
489 plot(lpvar.xvec_nc, lpvar.differenceY(lpvar.xvec_nc+1), 'o', 'Color', other.
        colourvec{2})
490 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
        {2}, 'LineStyle', '--')
491 legend('$y(k)-r(k)$', '$u(k)-r(k)$', '$\ell(k) \neq v(k)$', 'Interpreter', '
        latex')
492 xlim([0 Sim.kmax+1]); grid on
493 title('Difference signals')
494 ylabel('Time')
495 hold off
496
497 % ---- Mode sequence ----
498 ax3 = subplot(4,1,3); hold on
499 plot(lpvar.xvec_c, lpvar.mode_c, '*', 'Color', other.colourvec{1})
500 plot(lpvar.xvec_nc, lpvar.mode_nc, '*', 'Color', other.colourvec{2})
501 line([Sim.exp_conv_steps Sim.exp_conv_steps], [0 sys.nL+1], 'Color', other.
        colourvec{2}, 'LineStyle', '--')
502 legend('$\ell(k) = v(k)$', '$\ell(k) \neq v(k)$', 'Interpreter', 'latex')
503 set(gca, 'YTick', 0:(sys.nL+1))
504 ylim([0 sys.nL+1]); xlim([0 Sim.kmax+1]);
505 grid on
506 title('System mode $\ell(k)$')
507 ylabel('System mode $\ell(k)$')
508
509 % ---- State differences ----
510 ax4 = subplot(4,1,4); hold on
511 p = plot(0:Sim.kmax, lpvar.state(1:Sim.kmax+1,:)-Sim.ref(1:Sim.kmax+1),'*');
512 if sys.nx == 3
513    p(1).Color = other.colourvec{1}; p(2).Color = other.colourvec{5}; p(3).Color
         = other.colourvec{3};
514 end
```

```
515 line([Sim.exp_conv_steps Sim.exp_conv_steps], ylim, 'Color', other.colourvec
        {2}, 'LineStyle', '--')
516 legend('$x_1(k) - r(k)$', '$x_2(k) - r(k)$', '$x_3(k) - r(k)$', 'Interpreter',
        'latex')
517 title('Difference between states $x_i(k)$ and reference signal $r(k)$')
518 xlabel('Event step $k$');
519 ylabel('Time')
520 xlim([0 Sim.kmax+1]); grid on
521 hold off
522
523 % ---- Link axes ----
524 set(gca, 'TickLabelInterpreter','latex')
525 linkaxes([ax1, ax2, ax3, ax4], 'x')
526 %%
527 %
528 %
529 % *Clear meaningless variables*
530
531 clear ax1 ax2 ax3 ax4 vec a growth_append v f check_vec i j k p showprogress
532 clear progressval num_conseq_mode U VarNames vseqpos last_modes ind_os
        val_conseq_
533 clear mode mat Nsim maxit offsetRange prev_mode controller1 controller2
        controller3
534 %%
535 %
```

A-2-4 allSequences.m

```
1  function [seqm, seqv, Ps_tilde] = allSequences(nL, Np, Ps)
2
3  % Find all possible mode sequences over the prediction horizon and put them in
       a matrix
4  seqm = zeros(nL^Np, Np);    % Matrix containing all possible mode sequences
5  for i = 1:Np                % Loop over prediction horizon
6     vec = imresize((1:nL)', [nL^i 1], 'nearest');
7     seqm(:,Np+1-i) = repmat(vec, nL^(Np-i), 1);
8  end
9
10 % Find all possible sequences of  over the prediction horizon and put them in a
        matrix
11 seqv = seqm;
12
13 % Ps_tilde(i,m,v): Probability of advancing from mode i to mode sequence m
14 % given v_tilde(k)
15 Ps_tilde = zeros(nL, height(seqm), height(seqv));
16 for v = 1:height(seqv)                  % Loop over all possible sequences of v
17    for i = 1:nL                 % Loop over all possible previous modes
18       for j = 1:height(seqm)          % Loop over all mode sequences
19          Ps_tilde(i,j,v) = Ps(i, seqm(j, 1), seqv(v, 1));
20          for k = 1:Np-1       % Loop over all modes in a sequence
21             Ps_tilde(i,j,v) = Ps_tilde(i,j,v) * Ps(seqm(j, k), seqm(j,k+1),
                   seqv(v, k+1));
22          end
23       end
24    end
25 end
```

```
26
27
28 end
```

A-2-5 checkError.m

```
1  function [] = checkError(diagnostics,warnvar)
2  %UNTITLED Summary of this function goes here
3  %   Detailed explanation goes here
4      if diagnostics ~= 0
5          warning(['Yalmip error - ', yalmiperror(diagnostics)]);
6          if ~warnvar
7              warndlg('Yalmip error, check warning messages')
8              warnvar = true;
9              assignin("caller", "warnvar", warnvar)
10         end
11     end
12 end
```

A-2-6 findOffset.m

```
1  function [offsetPoP] = findOffset(A, C, Ps, Nsim_vec, rdub_np, rdb_np, seqm,
       maxit, ProbOfPerf, offsetRange)
2
3  % Define parameters
4  nx = size(A, 1);  % Number of system states
5  ny = 1;           % Number of system outputs
6  nL = size(A, 3);  % Number of system modes
7
8  numhor  = length(Nsim_vec);   % Number of simulation horizons
9
10 clrs = {'#00A6D6', '#E03C31', '#FFB81C', '#6CC24A', '#EC6842'};
11
12 % Define accuracy and confidence
13 delta   = 0.005;                          % Confidence 1-delta
14 epsilon = sqrt(1/(maxit/log(2/delta))/2); % Accuracy
15
16 % Define parameters for growth rate calculation
17 Nt = 10;    % Transient behaviour parameter
18
19 % Find minimum growth rate and sequence
20 [~, indPoP] = min(abs(rdb_np-rdub_np));   % Find index of control sequence
21 ctrl_seq = seqm(indPoP, :);               % Store corresponding control
       sequence
22
23 % Extend control sequence to full horizon
24 ctrl_seq = repmat(ctrl_seq, [1, ceil(max(Nsim_vec)/length(ctrl_seq))]);
25 ctrl_seq = ctrl_seq(1:max(Nsim_vec));
26
27 % Initialise growth rate vector
28 GR = zeros(maxit, numhor);
29
30 % Start parallel pool
31 p = gcp;
32 if isempty(p)
33    parpool;
```

```
34      p = gcp;
35  end
36
37  % Perform multisample
38  for n = 1:numhor
39      Nsim = Nsim_vec(n);
40      parfor j = 1:maxit
41
42          % Initialise vectors
43          state       = zeros(Nsim, nx);    % State vector
44          state(1,:)  = zeros(1, nx);       % Initial state
45          mode        = zeros(Nsim, 1);     % Mode vector
46          mode(1)     = 1;                  % Initial mode
47          output      = zeros(Nsim, ny);    % Output vector
48          output(1)   = 0;                  % Initial output
49
50          % Construct random vector to support mode switching
51          rng('shuffle');
52          num = rand(Nsim,1);
53
54          % Simulate over simulation horizon
55          for k = 2:Nsim
56
57              % Advance to next mode based on probabilities in Ps and v(k)
58              for i = 1:nL
59                  if num(k) < sum(Ps(mode(k-1), 1:i, ctrl_seq(k-1)))
60                      mode(k) = i;
61                      break
62                  end
63              end
64
65              % Update state and output
66              state(k, :) = mpMulti(A(:,:,mode(k)), state(k-1,:)');
67              output(k,:) = mpMulti(C(:,:,mode(k)), state(k,:)');
68
69          end
70
71          % Calculate growth rate for each sample
72          GR(j, n) = (output(Nsim)-output(1+Nt))/(Nsim-1-Nt);
73
74      end
75  end
76
77  % Close parallel pool
78  delete(p);
79
80  % Define linearly spaced offset parameter
81  offset = linspace(-offsetRange, offsetRange, 1e4);
82
83  % Plot probability of performance against offset
84  figure;
85  for i = 1:size(GR, 2)
86      subplot(size(GR,2),1,i)
87      hold on
88
89      % Calculate indices
90      frac = sum(GR(:,i)<=rdub_np+offset)/length(GR(:,i));
```

```
91      indPoP = find(frac>=ProbOfPerf, 1);
92      [~,ind0] = min(abs(offset));
93
94      % Construct error region
95      xconf = [offset offset(end:-1:1)];
96      yconf = [frac+epsilon frac(end:-1:1)-epsilon];
97      yconf(yconf>1) = 1;
98      yconf(yconf<0) = 0;
99      p = fill(xconf,yconf, 'red');
100     p.FaceColor = clrs{2};
101     p.FaceAlpha = 0.3;
102     p.EdgeColor = 'none';
103
104     % Plot data
105     plot(offset, frac, 'Color',clrs{1})
106     yl = ylim; xl = xlim;
107     line([offset(indPoP) offset(indPoP)], [yl(1) frac(indPoP)], 'LineStyle', '--
            ', 'Color', clrs{2})
108     line([offset(ind0) offset(ind0)], [yl(1) frac(ind0)], 'LineStyle', '--', '
            Color', 'black')
109     line([xl(1) offset(indPoP)], [frac(indPoP) frac(indPoP)], 'LineStyle', '--',
             'Color', clrs{2})
110     line([xl(1) offset(ind0)], [frac(ind0) frac(ind0)], 'LineStyle', '--', '
            Color', 'black')
111
112     % Add text
113     marY = 0.015;
114     marX = range(offset)*marY;
115     txt = "$o \approx $" + " " + sprintf('%.3f', offset(indPoP));
116     text(offset(indPoP)+marX, 0+marY, txt, "HorizontalAlignment","left", "
            VerticalAlignment","bottom", "Color", clrs{2})
117
118     txt = "$\widehat{p}_N(0.5) \approx $" + " " + sprintf('%.2f', frac(ind0));
119     text(min(offset)+marX, frac(ind0)+marY, txt, "HorizontalAlignment","left","
            VerticalAlignment","bottom")
120
121     txt = "$\widehat{p}_N(0.5) \approx $" + " " + sprintf('%.2f', frac(indPoP));
122     text(min(offset)+marX, frac(indPoP)-marY, txt, "HorizontalAlignment","left
            ","VerticalAlignment","top", "Color", clrs{2})
123
124     txt = "$\epsilon \approx $" + " " + sprintf('%.2f', epsilon);
125     text(min(offset)+marX, 0+marY, txt, "HorizontalAlignment","left","
            VerticalAlignment","bottom")
126
127     ylabel("$\widehat{p}_N(0.5)$", 'Interpreter', 'latex')
128     if i == size(GR, 2)
129        xlabel('Offset $o$', 'Interpreter', 'latex')
130     end
131     title("$\widehat{p}_N(0.5)$ as function of $o$ for $N_\mathrm{sim}=$ "+
            string(Nsim_vec(i)) + " and $1-\delta=$ " + sprintf('%.3f', 1-delta),'
            Interpreter', 'latex')
132     if i == 1
133        legend('$\widehat{p}_N(0.5)\pm \epsilon$', '$\widehat{p}_N(0.5)$', "$\
              widehat{p}_N(0.5)=$ "+string(ProbOfPerf), '$o=0$', 'Interpreter', '
              latex', 'location', 'east')
134     end
135     xlim([min(offset) max(offset)])
```

```
136     yticks(0:0.2:1)
137     grid on
138     hold off
139 
140 end
141 
142 % Find minimum offset
143 offsetPoP = offset(indPoP);
144 
145 % Plot histogram
146 figure;
147 
148 for i = 1:length(Nsim_vec)
149 
150     subplot(size(GR,2),1,i); hold on
151 
152     histogram(GR(:,i), 75, 'Normalization', 'probability', 'FaceColor', clrs{1})
153     ylim([0 0.05])
154     line(repmat(mean(GR(:,i)), [2 1]), ylim, 'Color', clrs{2}, 'LineWidth', 2, '
          LineStyle', ':')
155     line(repmat(rdub_np, [2 1]), ylim, 'Color', clrs{2}, 'LineWidth', 2)
156     if i == 1
157         legend('$\rho_{N_\mathrm{sim}}$', '$E\left[\rho_{N_\mathrm{sim}}\right]$'
              , '$\underline{\underline{\rho}}_{N_\mathrm{p}}$', 'Interpreter', '
             latex')
158     end
159     xlim([0.95*rdub_np 1.05*rdub_np])
160     grid on
161     ylabel('Relative probability', 'Interpreter', 'latex')
162     if i == length(Nsim_vec)
163         xlabel('Growth rate')
164     end
165     title("Distribution of sampled $\rho_{N_\mathrm{sim}}$ for $N_\mathrm{sim}=$
           "+string(Nsim_vec(i)),'Interpreter', 'latex')
166 
167 end
168 
169 end
```

A-2-7 generateSemigroup.m

```
1  function varargout = generateSemigroup(n1,n2,m,Red,Fix,magn,vec,check_irr)
2  % Code by Abhimanyu Gupta
3  
4  varargout = cell(1,nargout);
5  
6  if vec
7      A = zeros(n1*n2,m);
8      if ~Fix
9          Supp = zeros(n1*n2,m);
10     end
11 else
12     A = zeros(n1,n2,m);
13     if ~Fix
14         Supp = zeros(n1,n2,m);
15     end
16 end
```

```
17
18 if Fix
19     Supp1 = generateSupport(n1,n2,Red,check_irr);
20 end
21
22 for i = 1:m
23     B = zeros(n1*n2,1);
24     if ~Fix
25         Supp1 = generateSupport(n1,n2,Red,check_irr);        %Generate support
26     end
27     nz = nnz(Supp1);
28     ai = rand(nz,1);              %Uniformly dist. matrix elements
29 %      ai = randi(5,nz,1);
30     ai = ai/sum(ai);
31
32     ind = logical(Supp1);
33     B(ind,:) = ai*magn;
34     B(~ind,:) = -inf;
35
36     if ~vec
37         A(:,:,i) = reshape(B,[n1,n2]).';
38         Supp(:,:,i) = reshape(Supp1,[n1,n2]).';
39     else
40         A(:,i) = B;
41         Supp(:,i) = Supp1;
42     end
43     clear ind
44 end
45 varargout{1} = A;
46
47 if nargout == 2
48     varargout{2} = Supp;
49 end
50
51 end
```

A-2-8 generateSupport.m

```
1  function Supp = generateSupport(n1,n2,Red,check_irr)
2  % Code by Abhimanyu Gupta
3
4  %Assumption: all matrices are regular!
5
6  % n                                    % # Nodes
7  % l                                    % # Support matrices
8  if Red==1
9      % Reducible case
10     Supp = zeros(n1,n2,1);
11     tmp = 0;                                %Counter
12     while tmp~=1
13         P = randi([0 1],n1*n2,1);
14         P1 = reshape(P,[n1,n2]);
15         reg = min([sum(P1),sum(P1,2)']);
16         if check_irr
17             I = (eye(n1)+P1)^(n1-1);       %Irreducibility => I>0 (elementwise)
18             if nnz(I)<n1^2&&reg~=0         %Reducibility and regularity check
19                 tmp = tmp+1;
```

```
20                  Supp = P;
21              end
22          else
23              tmp = tmp+1;
24          end
25      end
26
27  elseif Red==0
28      % Irreducible case
29      Supp = zeros(n1*n2,1);
30      tmp = 0;                                    %Counter
31      while tmp~=1
32          P = randi([0 1],n1*n2,1);
33          P1 = reshape(P,[n1,n2]);
34
35          if check_irr
36              reg = min([sum(P1),sum(P1,2)']);
37              I = (eye(n1)+P1)^(n1-1);        %Irreducibility => I>0 (elementwise)
38              if nnz(I)==n1^2&&reg~=0          %Irreducibility and regularity check
39                  tmp = tmp+1;
40                  Supp = P;
41              end
42          else
43              tmp = tmp+1;
44          end
45      end
46
47  else
48      % Random case
49      Supp = zeros(n1*n2,1);
50      tmp = 0;                                    %Counter
51      while tmp~=1
52          P = randi([0 1],n1*n2,1);
53          P1 = reshape(P,[n1,n2]);
54          if check_irr
55              reg = min([sum(P1),sum(P1,2)']);
56              if reg~=0          %Regularity check
57                  tmp = tmp+1;
58                  Supp = P;
59              end
60          else
61              reg = max(P1);
62              reg2 = min(P1);
63              if reg-reg2~=0
64                  tmp = tmp+1;
65                  Supp = P;
66              end
67          end
68      end
69
70  end
71
72  end
```

A-2-9 generateSystem.m

```
function [A, B, C, Ps] = generateSystem(nx, nu, ny, nL, magn)
% Based on code by Abhimanyu Gupta

if nargin == 4
    magn = 10;
end

% Settings
% magn = 1000;  % Scaling of matrix elements
Red  = 0;   % Reducible => Red = 1; Irreducible => Red = 0; Else random
Fix  = 0;   % Fixed support => Fix = 1, otherwise pseudorandom spport using
    randi()
vec  = 0;   % Output as vectors => vec = 1 else as 3D arrays, keep it as it is

% Construct random semigroup
[A,~] = generateSemigroup(nx,nx,nL,Red,Fix,magn,vec,1);     % Generate A-
    matrices
[B,~] = generateSemigroup(nx,nu,nL,0,0,magn,vec,1);         % Generate B-
    matrices

C = zeros(1, nx);
C = repmat(C, 1, 1, nL);                                % Define for each mode

Ps = zeros(nL,nL,nL);

for i = 1:nL
   for j = 1:nL
      for k = 1:nL
         if i == j
            Ps(k, j, i) = 0.8;
         else
            Ps(k, j, i) = 0.2/(nL-1);
         end
      end
   end
end
```

A-2-10 growthRate.m

```
function [growth] = growthRate(A, C, seqm, Ps_tilde, varargin)

% Read inputs
names  = varargin(1:2:end);
values = varargin(2:2:end);

% (default) Parameters
nL = size(A, 3);
nx = size(A, 1);
Np = size(seqm,2);
Nt = 10;
maxnummode = 3;
seqv = seqm;
scale = ceil(100/Np);

% Switch calculations on/off for efficiency
CalcGRPerMode  = 1;
CalcRB         = 1;
```

```
CalcRDB         = 1;
CalcConst       = 1;

% Process inputs
for k = 1:numel(names)
    switch names{k}
        case "Nt"
            Nt = values{k};
        case "maxnummode"
            maxnummode = values{k};
        case "seqv"
            seqv = values{k};
        case "CalcGRPerMode"
            CalcGRPerMode = values{k};
        case "CalcRB"
            CalcRB = values{k};
        case "CalcRDB"
            CalcRDB = values{k};
        case "CalcConst"
            CalcConst = values{k};
        case "scale"
            scale = values{k};
    end
end

% Check if Nt > Nt
if Nt >= Np*scale
    Nt = Np*scale-1;
    warning("Nt set to " + num2str(Nt))
end

% Scale matrices
seqm = repmat(seqm, [1 scale]);
seqv = repmat(seqv, [1 scale]);
Np = scale*Np;

if CalcConst
    % Mark rows that violate constraint
    for i = 1:size(seqm,1)
        if strfind(diff(seqm(i,:), [], 2)==0, ones(1, maxnummode))
            violate_con(i) = 1;
        else
            violate_con(i) = 0;
        end
    end
end

% -------------------------------------------------
% Growth rate per mode
% -------------------------------------------------
if CalcGRPerMode
    for i = 1:nL
        growth.rho_mode(i) = mp_mcm(A(:,:,i));
    end
end

% -------------------------------------------------
```

```
76 % Maximum unconstrained growth rate
77 % -------------------------------------------------
78 if CalcRB
79    S_A = eps*ones(nx);              % Initialise max-plus addition of all A-
          matrices
80    for ell = 1:nL
81       S_A = mpAdd(S_A, A(:,:,ell));
82    end
83    growth.rho_bar = mp_mcm(S_A);
84 end
85
86 % Check if switching probabilities depend on initial mode
87 % If not, don't loop over all initial modes
88 if sum(diff(Ps_tilde, 1, 1), 'all') < 1e5
89    modeindependent = true;
90 else
91    modeindependent = false;
92 end
93
94 % -------------------------------------------------
95 % Finite-horizon unconstrained expected growth rate
96 % -------------------------------------------------
97 if CalcRDB
98    x0   = zeros(nx, 1);    % Initial state
99    rate = zeros(size(seqm,1), 1);
100   EGR_mat = zeros(nL, size(seqv,1));
101   for i = 1:size(seqm,1)  % Loop over all mode sequences
102
103      x_loop(:,1) = x0;
104      y_loop = []; y_loop(1) = mpMulti(C(:,:,1), x_loop(:, 1));
105      for j = 1:Np         % Loop over prediction horizon
106         x_loop(:,j+1) = mpMulti(A(:,:,seqm(i,j)), x_loop(:,j));
107         y_loop(j+1)   = mpMulti(C(:,:,seqm(i,j)), x_loop(:,j+1));
108      end
109
110      % Growth rate of a certain mode sequence
111      rate(i) = (y_loop(end)-y_loop(1+Nt))/(Np-Nt);
112
113   end
114   for k = 1:nL-(nL-1)*modeindependent    % Loop over all initial modes
115      for l = 1:size(seqv,1)                % Loop over all control sequences
116         % Expected growth rate for every initial mode and every control
              sequence
117         EGR_mat(k, l) = reshape(Ps_tilde(k, :, l), [1 size(seqv,1)])*rate;
118      end
119   end
120
121   if modeindependent
122      EGR_mat = repmat(EGR_mat(1,:), [nL, 1]);
123   end
124
125   % Store values
126   growth.rho_dbar_Np = mean(EGR_mat, 1)';
127   growth.rho_dubar_Np = min(growth.rho_dbar_Np);
128   growth.rho_ubar_Np = min(rate, [], 'all');
129 end
130
```

```
% ------------------------------------------------
% Finite-horizon constrained expected growth rate
% ------------------------------------------------
if CalcConst
    growth.rho_dbar_Np_con = mean(EGR_mat(:,~violate_con), 1)';
    growth.rho_dubar_Np_con = min(growth.rho_dbar_Np_con);
    growth.rho_ubar_Np_con = min(rate(~violate_con), [], 'all');
    growth.violate_con = violate_con;
end

growth.seqm = seqm;

end
```

A-2-11 modeConstraints.m

```
function [adm_cont, adm_cont_seq] = modeConstraints(maxnummode, nv, seqv)

adm_cont = cell(maxnummode*nv, 1);

% Define admissible control inputs for next event step
for i = 1:length(adm_cont)
    adm_cont{i} = 1:nv;
    if ismember(i, maxnummode:maxnummode:nv*maxnummode)
        vec = 1:nv;
        vec(vec==i/maxnummode) = [];
        adm_cont{i} = vec;
    end
end

% Define admissible control sequence over prediction horizon
adm_cont_seq = cell(maxnummode*nv, 1);

for i = 1:length(adm_cont_seq)
    % Number of equal consecutive modes in a row
    num_conseq_mode = rem(i-1, maxnummode)+1;
    % Value of the mode
    val_conseq_mode = floor((i-1)/maxnummode)+1;

    % Find for each control action the max number of modes in a row
    last_modes = val_conseq_mode*ones(length(seqv), num_conseq_mode);
    vec = [last_modes seqv];
    vseqpos = ones(1,length(seqv));


    for j = 1:length(seqv)
        for k = 1:nv
            check_vec = k*ones(1, maxnummode+1);
            if ~isempty(strfind(vec(j,:), check_vec))
                vseqpos(j) = 0;
            end
        end
    end

    adm_cont_seq{i} = vseqpos;
end

```

```
42 end
```

A-2-12 mpAdd.m

```
1  function [Z] = mpAdd(X, Y)
2
3  [n1,n2]=size(X);
4  [~,~]=size(Y);
5  Z=zeros(n1,n2);
6
7  for i=1:n1
8     Z(i,:) = max([X(i,:);Y(i,:)]);
9  end
10
11 end
```

A-2-13 mpMulti.m

```
1  function [Z] = mpMulti(X, Y)
2
3  [n1,~]=size(X);
4  [~,n4]=size(Y);
5  C=zeros(n1,n4);
6  oo=ones(1,n4);
7
8  Z = zeros(n1, n4);
9  for i=1:n1
10    Z(i,:) = max((X(i,:)'*oo)+Y);
11 end
12
13 end
```

A-2-14 predictionModel.m

```
1  function [A_tilde, C_tilde, B_tilde, D_tilde] = predictionModel(A, B, C, seqm)
2
3  [nx, ~, nL] = size(A);
4  Np = size(seqm, 2);
5  nu = size(B,2);
6  ny = size(C,1);
7
8  % Construct A_tilde and C_tilde
9  A_tilde = repmat(mp_eye(nx, nx), 1, 1, nL^Np);  % Initialise with max-plus
       identity matrix
10 C_tilde = zeros(ny*Np, nx, nL^Np);              % No initialisation necessary
11 for j = 1:nL^Np     % loop over all mode sequences
12    for i = 1:Np    % Loop over the prediction horizon
13       A_tilde(:,:,j) = mpMulti(A(:,:,seqm(j,i)), A_tilde(:,:,j));
14       C_tilde(i,:,j) = mpMulti(C(:,:,seqm(j,i)), A_tilde(:,:,j));
15    end
16 end
17
18 % Construct B_tilde and D_tilde
19 B_tilde = zeros(nx, nu, Np, Np);
20 D_tilde = zeros(ny*Np, nu*Np, nL^Np);
21 for p = 1:nL^Np          % Loop over all mode sequences
```

```
22      for i = 1:Np          % Loop over rows of B_tilde and D_tilde
23          for k = 1:Np      % Loop over columns of B_tilde and D_tilde
24
25              if i > k              % Lower triangular entries of D_tilde
26                  mat = B(:,:,seqm(p, k));
27                  for j = 1:i-k
28                      mat = mpMulti(A(:,:,seqm(p,k+j)), mat);
29                  end
30                  B_tilde(:,:,i,k) = mat;
31              elseif i == k         % Diagonal entries of D_tilde
32                  B_tilde(:,:,i,k) = B(:,:,seqm(p, i));
33              elseif i < k          % Upper triangular entries of D_tilde
34                  B_tilde(:,:,i,k) = eps*ones(nx, nu);
35              end
36
37              D_tilde(i,k,p) = mpMulti(C(:,:, seqm(p,i)), B_tilde(:,:,i,k));
38
39          end
40      end
41  end
42
43  end
```

Bibliography

[1] M. Pesselse, *Modelling and Optimal Scheduling of Inland Waterway Transport Systems (MSc Thesis)*. PhD thesis, Delft University of Technology, 2022.

[2] J. C. Maxwell, "I. On governors," *Proceedings of the Royal Society of London*, vol. 16, pp. 270–283, 12 1868.

[3] R. Cuninghame-Green, "Minimax algebra," *Lecture notes in economics and mathematical systems*, vol. 166, 1979.

[4] T. J. van den Boom and B. De Schutter, "Modeling and control of switching max-plus-linear systems with random and deterministic switching," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 22, no. 3, pp. 293–332, 2012.

[5] A. Gupta, T. J. van den Boom, J. van der Woude, and B. De Schutter, "Framework for Studying Stability of Switching Max-Plus Linear Systems," *IFAC-PapersOnLine*, vol. 53, pp. 68–74, 7 2020.

[6] A. Gupta, T. J. van den Boom, J. van der Woude, and B. De Schutter, "Structural Controllability of Switching Max-Plus Linear Systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1936–1942, 2020.

[7] A. M. Liapounoff, "Problème général de la stabilité du mouvement," in *Annales de la Faculté des sciences de Toulouse: Mathématiques*, vol. 9, pp. 203–474, 1907.

[8] W. Hahn, *Stability of Motion*, vol. 138. Springer, 1967.

[9] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Science & Business Media, 2008.

[10] B. Heidergott, G.-J. Olsder, and J. van der Woude, *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton, New Jersey: Princeton University Press, 2005.

[11] R. Cuninghame-Green, "Describing industrial processes with interference and approximating their steady-state behaviour," *Journal of the Operational Research Society*, vol. 13, no. 1, pp. 95–100, 1962.

[12] F. Baccelli, G. Cohen, G.-J. Olsder, and J.-P. Quadrat, *Synchronization and Linearity: an Algebra for Discrete Event Systems*. John Wiley & Sons Ltd, 1992.

[13] J. Gunawardena, *An Introduction to Idempotency*. Cambridge, U.K.: Publications of the Newton Institute, 1998.

[14] J. Komenda, S. Lahaye, J.-L. Boimond, and T. J. van den Boom, "Max-plus algebra in the history of discrete event systems," *Annual Reviews in Control*, vol. 45, pp. 240–249, 2018.

[15] J. Braker, "Max-algebra modelling and analysis of time-dependent transportation networks," in *Proceedings of First European Control Conference*, (Grenoble, France), pp. 1831–1836, 1991.

[16] T. J. 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, no. 3, pp. 302–309, 2004.

[17] S. Masuda, H. Goto, T. Amemiya, and K. Takeyasu, "An Inverse System for Linear Parameter-Varying Max-Plus-Linear Systems," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 4, pp. 4549–4554, IEEE, 2002.

[18] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, p. 415, 2011.

[19] C. R. Cutler and B. L. Ramaker, "Dynamic matrix control: A computer control algorithm," *IEEE Transactions on Automatic Control*, vol. 17, p. 72, 1979.

[20] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[21] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, and J. Orwant, "Quantitative analysis of culture using millions of digitized books," *science*, vol. 331, no. 6014, pp. 176–182, 2011.

[22] M. Morari, C. E. Garcia, and D. M. Prett, "Model predictive control: Theory and practice - A survey," in *IFAC Proceedings Volumes*, vol. 25, pp. 335–348, 1989.

[23] E. Fernandez-Camacho and C. Bordons-Alba, *Model Predictive Control in the Process Industry*. Springer, 1995.

[24] D. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control—Part I: The basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, 1987.

[25] L. T. Biegler, "Efficient Solution of Dynamic Optimization and NMPC Problems," in *Allgöwer F., Zheng A. (eds) Nonlinear Model Predictive Control*, vol. 26, pp. 219–243, Basel: Birkhäuser Basel, 2000.

[26] B. De Schutter and T. J. van den Boom, "Model predictive control for max-plus-linear discrete event systems," *Automatica*, vol. 37, no. 7, pp. 1049–1056, 2000.

[27] S. P. Boyd and C. H. Barratt, *Linear Controller Design: Limits of Performance*. Prentice Hall Englewood Cliffs, NJ, 1991.

[28] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

[29] N. Guglielmi, O. Mason, and F. Wirth, "Barabanov norms, Lipschitz continuity and monotonicity for the max algebraic joint spectral radius," *Linear Algebra and its Applications*, vol. 550, pp. 37–58, 2018.

[30] B. De Schutter and W. Heemels, *Lecture Notes on Modeling and Control of Hybrid Systems*. Delft Center for Systems and Control, Delft University of Technology, The Netherlands, 2015.

[31] F. Dercole and F. Della Rossa, "Tree-based algorithms for the stability of discrete-time switched linear systems under arbitrary and constrained switching," *IEEE Transactions on Automatic Control*, vol. 64, no. 9, pp. 3823–3830, 2018.

[32] F. D. Torrisi and A. Bemporad, "HYSDEL - A tool for generating computational hybrid models for analysis and synthesis problems," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 2, pp. 235–249, 2004.

[33] A. Bemporad and S. di Cairano, "Model-Predictive Control of Discrete Hybrid Stochastic Automata," *IEEE Transactions on Automatic Control*, vol. 56, pp. 1307–1321, 6 2011.

[34] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized algorithms for analysis and control of uncertain systems: with applications*. Springer, 2013.

[35] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.

[36] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, *A Modern Introduction to Probability and Statistics: Understanding why and how*, vol. 488. Springer, 2005.

[37] J. Bernoulli, "Ars Conjectandi," 1713.

[38] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, pp. 493–507, 1952.

[39] L. R. Ray and R. F. Stengel, "A Monte Carlo approach to the analysis of control system robustness," *Automatica*, vol. 29, no. 1, pp. 229–236, 1993.

[40] T. J. 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, no. 10, pp. 1199–1211, 2006.

[41] T. J. van den Boom and B. De Schutter, "Stabilizing model predictive controllers for randomly switching max-plus-linear systems," in *2007 European Control Conference (ECC)*, pp. 4952–4959, IEEE, 2007.

[42] T. J. van den Boom and B. De Schutter, "Model predictive control for switching max-plus-linear systems with random and deterministic switching," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 7660–7665, 2008.

[43] T. J. van den Boom and B. De Schutter, "Randomly switching max-plus linear systems and equivalent classes of discrete event systems," in *2008 9th International Workshop on Discrete Event Systems*, pp. 242–247, IEEE, 2008.

[44] B. De Schutter and T. J. van den Boom, "On model predictive control for max-min-plus-scaling discrete event systems," *Technical Report Bds 00-04: Control Systems Engineering, Faculty of Information Technology and Systems*, 2000.

[45] E. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Transactions on automatic control*, vol. 26, no. 2, pp. 346–358, 1981.

[46] V. Kozyakin, "The Berger-Wang formula for the Markovian joint spectral radius," *Linear Algebra and Its Applications*, vol. 448, pp. 315–328, 5 2014.

[47] X. Dai, "A Gel'fand-type spectral radius formula and stability of linear constrained switching systems," *Linear Algebra and its Applications*, vol. 436, pp. 1099–1113, 3 2012.

[48] Y. Wang, N. Roohi, G. E. Dullerud, and M. Viswanathan, "Stability analysis of switched linear systems defined by regular languages," *IEEE Transactions on Automatic Control*, vol. 62, no. 5, pp. 2568–2575, 2017.

[49] K. M. Passino and K. L. Burgess, *Stability Analysis of Discrete Event Systems*, vol. 16. Wiley-Interscience, 1998.

[50] C. Commault, "Feedback stabilization of some event graph models," *IEEE transactions on Automatic Control*, vol. 43, no. 10, pp. 1419–1423, 1998.

[51] G.-C. Rota and W. Strang, "A note on the joint spectral radius," in *Proceedings of the Netherlands Academy*, p. 379–381, 1960.

[52] J. N. Tsitsiklis and V. D. Blondel, "The Lyapunov exponent and joint spectral radius of pairs of matrices are hard—when not impossible—to compute and to approximate," *Mathematics of Control, Signals and Systems*, vol. 10, no. 1, pp. 31–40, 1997.

[53] S. Gaubert, "Performance evaluation of (max,+) automata," *IEEE transactions on automatic Control*, vol. 40, no. 12, pp. 2014–2025, 1995.

[54] Y. Chitour, G. Mazanti, and M. Sigalotti, "On the gap between deterministic and probabilistic joint spectral radii for discrete-time linear systems," *Linear Algebra and Its Applications*, vol. 613, pp. 24–45, 3 2021.

[55] I. Daubechies and J. C. Lagarias, "Sets of matrices all infinite products of which converge," *Linear algebra and its applications*, vol. 161, pp. 227–263, 1992.

[56] P. Butkovič, "On tropical supereigenvectors," *Linear Algebra and Its Applications*, vol. 498, pp. 574–591, 6 2016.

[57] M. Philippe, R. Essick, G. E. Dullerud, and R. M. Jungers, "Stability of discrete-time switching systems with constrained switching sequences," *Automatica*, vol. 72, pp. 242–250, 10 2016.

[58] R. D. Katz, "Max-plus (A, B)-invariant spaces and control of timed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 2, pp. 229–241, 2007.

[59] E. Menguy, J.-L. Boimond, L. Hardouin, and J.-L. Ferrier, "Just-in-time control of timed event graphs: update of reference input, presence of uncontrollable input," *IEEE Transactions on Automatic Control*, vol. 45, no. 11, pp. 2155–2159, 2000.

[60] T. J. van den Boom and B. De Schutter, "Properties of MPC for max-plus-linear systems," *European Journal of Control*, vol. 8, pp. 453–462, 1 2002.

[61] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear model predictive control*, pp. 345–369, Springer, 2009.

[62] M. Johansson, "Piecewise linear control systems," 1999.

[63] B. De Schutter and T. J. van den Boom, "MPC for continuous piecewise-affine systems," *Systems and Control Letters*, vol. 52, pp. 179–192, 7 2004.

[64] H. P. Williams, *Model Building in Mathematical Programming.* New York: Wiley, 3rd editio ed., 1993.

[65] J. Stanczyk, "Max-Plus Algebra Toolbox for MATLAB," 2016.

[66] G.-J. Olsder, K. Roos, and R.-J. van Egmond, "An efficient algorithm for critical circuits and finite eigenvectors in the max-plus algebra," *Linear Algebra and its Applications*, vol. 295, no. 1-3, pp. 231–240, 1999.

[67] "MATLAB R2022a," 2022.

[68] J. Löfberg, "YALMIP," 2021.

[69] R. Bixby, Z. Gu, and E. Rothberg, "Gurobi," 2008.

Glossary

List of Acronyms

DCSC	Delft Center for Systems and Control
DES	discrete-event system
MPC	model predictive control
PTL	plus-times linear
MPA	max-plus algebra
MPL	max-plus linear
MP-LPV	max-plus linear parameter-varying
SMPL	switching max-plus linear
RSMPL	randomly switching max-plus linear
MMPS	max-min-plus-scaling
PWA	piecewise-affine
MLD	mixed logical dynamical
LP	linear programming
MILP	mixed-integer linear programming
MIQP	mixed-integer quadratic programming
LQG	linear-quadratic-gaussian
HA	hybrid automaton
DFA	deterministic finite automaton
nDFA	nondeterministic finite automaton
DHA	discrete hybrid automaton
DHSA	discrete hybrid stochastic automaton
FSM	finite state machine
sFSM	stochastic finite state machine

SAS	switched affine system
EG	event generator
MS	mode selector
JSR	joint spectral radius
LSR	lower spectral radius
CJSR	constrained joint spectral radius
SISO	single-input single-output
LMI	linear matrix inequality
MC	Monte Carlo
PDF	probability density function
MSE	mean square error

List of Symbols

Δ	Difference operator; $\Delta x(k) = x(k) - x(k-1)$
Γ_i	A polyhedral partition of the space $\mathbb{R}^{n_w}$
ρ_{o}	Asymptotic output growth rate
ρ_{s}	Asymptotic state growth rate
$\overline{\overline{\rho}}$	Expected growth rate of a switching max-plus linear (SMPL) system based on controlled stochastic evolution, given a discrete control sequence (pronounced 'rho double bar')
$\overline{\overline{\rho}}_{N_{\mathrm{p}}}(k)$	Expected growth rate over the prediction horizon N_{p}
$\overline{\rho}$	Maximum growth rate of an SMPL system under autonomous evolution (pronounced 'rho bar')
$\overline{\rho}_{N_{\mathrm{p}}}(k)$	Upper bound on the finite-horizon growth rate $\mu_{N_{\mathrm{p}}}$
$\rho_{N_{\mathrm{p}}}(k)$	(Expected) finite-horizon growth rate over the prediction horizon N_{p}
$\underline{\rho}$	Best case minimum growth rate of an SMPL system based on controlled deterministic switching (pronounced 'rho underbar')
$\underline{\rho}_{N_{\mathrm{p}}}(k)$	Lower bound on the finite-horizon growth rate $\mu_{N_{\mathrm{p}}}(k)$
$\underline{\underline{\rho}}$	Minimum expected growth rate of an SMPL system based on controlled stochastic evolution (pronounced 'rho double underbar')
$\underline{\underline{\rho}}_{N_{\mathrm{p}}}(k)$	Minimum expected growth rate over the prediction horizon N_{p}
ε	Zero element in a dioid
ξ_i	Cycle time vector representing the asymptotic average time between event occurrences
ℓ	Mode of an SMPL system
$\mathbb{N}$	The set of natural numbers
$\mathbb{R}$	Real numbers

$\mathbb{R}^q$	Vector space of q-tuples of real numbers
$\mathbb{R}^{q\times q}$	Vector space of $q \times q$ real matrices
$\mathbb{R}_\varepsilon$	Equal to $\mathbb{R} \cup \{\varepsilon\}$
$\mathbb{1}_n$	Max-plus algebraic unity vector of all zeros of dimension n
$\mathbb{1}_n$	Vector of all zeros of dimension n
$\mathcal{E}$	Max-plus zero matrix
$\mathcal{G}(\mathcal{A})$	A directed graph associated with a matrix $A \in \mathbb{R}_\varepsilon^{n\times n}$, consisting of the pair $(V(A), E(A)$
$\mathcal{L}$	Set of all modes of an SMPL system $\{1, \ldots, n_L\}$
$\mathcal{R}$	Semiring
$\mathcal{R}_{\max}$	The max-plus algebra $(\mathbb{R}_\varepsilon, \oplus, \otimes, \varepsilon, e)$
$\underline{n}$	Set of all positive integers up to n
E	Max-plus identity matrix
e	Unit element in a dioid
N_{c}	Control horizon MPC
N_{p}	Prediction horizon MPC
N_{t}	Parameter used to reduce influence from transients in the calculation of finite-horizon growth rate $\mu_{N_{\mathrm{p}}}(k)$
n_L	Number of modes of an SMPL system
n_u	Number of continuous system inputs
n_v	Number of discrete system inputs
n_x	Number of system states
n_y	Number of system outputs
$u(k)$	Input vector with length n_u
$w(k)$	Full state vector with length n_w of a type-1 SMPL system
$x(k)$	State vector with length n_x
$y(k)$	Output vector with length n_y
$z(k)$	Switching variable with length n_z in an SMPL system description
$^\top$	Symbol to denote transpose
$\setminus$	Relative complement, i.e., $A\setminus B$ is the set of objects that belong to A and not B
$\|A\|_{\max}$	The maximal finite entry of matrix A
$\|A\|_{\min}$	The minimal finite entry of matrix A
$\|x\|_\infty$	The supremum norm of vector x
$\|x\|_{\mathbb{P}}$	The projective norm of vector x
$\neg$	Logical inverse
$\odot$	Max-plus Schur product
$\oplus$	Addition in a dioid (pronounced 'oplus')
$\otimes$	Multiplication in a dioid (pronounced 'otimes')
$\preceq_i$	A vector operator where the entries stand for $\leq$ or $<$

Index