Guaranteed Set-based Controller Design for Hybrid Dynamical Systems

Dissertation
zur Erlangung des akademischen Grades

Doktoringenieur
(Dr.-Ing.)

von **M.Sc. Ing. Petar Andonov**
geb. am 10. Januar 1987 in Varna, Bulgarien

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke-Universität Magdeburg

Gutachter: Prof. Dr.-Ing. Rolf Findeisen
Prof. Dr. rer. nat. Frank Ortmeier
Prof. Dr.-Ing. Sergio Lucia

eingereicht am 25. März 2021
Promotionskolloquium am 30. Juli 2021

"If we knew what we were doing, we wouldn't call it research."

Albert Einstein

Contents

Abstract

Control systems lie at the heart of many technical achievements. Often overall system and thus technological progress is achieved by increasing the quantity and complexity of the imposed system requirements. Achieving the requirements typically involves that controller satisfies safety limits, improves the performance, and provides efficient operation. This, in turn, poses the challenge in systematically formulating and guaranteeing these often diverse requirements.

This work presents a set-based framework for guaranteed controller tuning and analysis of hybrid dynamical systems. The three main pillars of this work are: first, the discussion and formulation of quantitative and qualitative requirements; second, providing guarantees for the system behaviour in spite of the uncertainties; and third, taking the dynamical phenomena typical for hybrid systems into account.

With respect to the considered uncertainties, we consider an unknown-but-bounded uncertainties, where the uncertainties are specified by semi-algebraic sets. To accommodate the uncertainties and still provide guarantees, we consider a set-based formulation in the form of a feasibility problem. This set-up allows to obtain in an easy way approximation sets of the controller parameters that satisfy all conditions. Furthermore, the set-up allows including system performance specifications and robust control requirements, which we divide into quantitative and qualitative requirements.

Quantitative requirements enforce primarily fixed admissible ranges for the variables' values at specified times, despite uncertainties. They are motivated by, for example, the requirements on the transient response characteristics. They can also be used to formulate control scenarios like disturbance rejection, path following, and trajectory tracking. In contrast, qualitative requirements focus on temporal uncertainty and conditional constraints. Through qualitative requirements, we can take into account logical conditions and include temporal operators inspired by temporal logic. Moreover, we show how to handle discontinuity phenomena directly in the controller design. As a result, the quantitative and qualitative requirements provide the foundation to consider and enforce a wide range of requirements when designing and tuning a controller. For example, the set-based approach allows to design the control response relative to the initial conditions or to derive reference values for controller tuning. It furthermore allows to deal with the problem of robust error-free steady-state control. Moreover, we are interested in the controller parameters that provide specific behaviours for all initial conditions and all references from a desired set of values.

Throughout this work, we provide simulation examples, as well as experimental validation results. These examples demonstrate the presented approach's applicability

starting from a single control loop up to the operation validation and at a plant-wide scale. The applications vary from level control in tanks, through magnetic levitation and battery charging, to discrete-manufacturing systems.

Deutsche Kurzfassung

Regelsysteme sind das Herzstück vieler technischer Innovationen. Häufig erfordert dieser technologische Fortschritt eine Erhöhung notwendigen, oftmals komplexe Systemanforderungen. Die Erfüllung der Anforderung involviert oftmals die Entwurf von Regler, die unter aller Umständen Sicherheitsgrenzen einhält, die Güte verbessert und einen effizienten Betrieb ermöglichen. Dies wiederum führt die Frage, wie sich die unterschiedlichen Systemanforderungen systematisch formulieren und unter Unsicherheiten mit Garantien erfüllen lassen.

Dieser Arbeit stellt einen mengenbasierten Ansatz für die garantierte Reglertunen und für die Analyse hybrider dynamischer Systeme vor. Die drei Eckpfeiler der Arbeit bilden: die Diskussion und Formulierung quantitativer und qualitativer Anforderungen und den Regelkreis; zweitens die Gewährleistung des Systemverhaltens trotz unvermeidlichen Unsicherheiten und drittens die Berücksichtigung für die hybride Systeme dynamische Phänomenen.

Hinsichtlich der betrachtenden Unsicherheiten berücksichtigen wir unbekannte, aber begrenzte Unsicherheit, welche durch semi-algebraische Mengen spezifiziert werden können. Um diesen Unsicherheiten Rechnung zu tragen und trotzdem Garantien zu leisten, greifen wir auf einen mengenbasierten Formulierung des Regelungsproblem zurück. Diese erlaubt es auf einfache Weise Reglerparametermengen zulassen zu bestimmen. Daneben kommen Anforderungen an die Güte und Robustheit in Form von quantitative und qualitative Anforderungen berücksichtigen werden.

Quantitativen Anforderungen betreffen in erster Linie räumliche Forderungen in Form von festen zulässigen Bereichen von (Regel-)Größen zu bestimmten Zeiten. Sie sind durch Anforderungen an das transiente Antwortverhalten motiviert, erlauben aber auch die Berücksichtigung anderen Regelungsszenarien wie Störungsunterdrückung, Pfadverfolgung und Trajektorienverfolgung. Im Gegensatz hierzu erlauben qualitativen Anforderungen die Berücksichtigung logischbedingte Einschränkungen. Qualitativen Anforderungen erlauben es z. B. logische Anforderungen zu berücksichtigen und zeitliche Verknüpfungen zwischen Signale einzubeziehen. Daneben zeigen wir auf, wie Erreignisgebunden Phänomene in Reglerentwurf einbezogen werden können. Die quantitativen und qualitativen Anforderungen legen die Basis, um bei der Auswahl und Einstellung eines Reglers eine breite Palette an Anforderung einzubeziehen. Die gewählte mengenbasierten Ansatz liefern Antworten ein breites Spektrum an Fragen, wie z. B. die Berücksichtigung von Anfangsbedingungen oder der Referenzwerten bei der Reglereinstellen oder der robusten fehlerfreien stationären Regelung. Darüber erlaubt der Ansatz Reglerparameter, die das gewünschte Verhalten für alle Anfangsbedingungen

und alle Sollwerte garantieren, zu bestimmen.

Die Ergebnisse und Methoden werden anhand von Simulationen und Experimente validiert. Die Beispiele untermauern die Anwendbarkeit des vorgestellten Ansatzes von einem einzelnen Regelkreis bis hin zur Validierung des Verhaltens einer komplexen Anlage im anlagenweiten Maßstab. Die vorgestellten Beispiele reichen von der Füllstandsregelung, eine Magnetschweberegelung, der Regelung einer Batterieladung bis hin zu diskreten Fertigungssystemen.

1 Introduction

> Uncertainty is the only certainty there is, and knowing how to live with insecurity is the only security.
>
> The father of John Allen Paulos

Automated processes support more and more of our daily private and professional life. Each autonomous system poses a challenge, stemming, for example, from the size or complexity of the system. Respectively, the requirements that need to be fulfilled by the system grow in number and become more elaborate. Such a rise in complexity is noticeable even for well-known and understood engineering systems, e.g. by the need to take into account dependencies to other systems or to include complex temporal and conditional behaviours.

Often, some parts of the dynamics are neglected or considered phenomenologically due to either the lack of a systematic approach to handle them or the resulting complexity. These cases are commonly handled by imposing limiting assumptions, e.g. the system is allowed to operate only inside a narrow range where it behaves linearly and does not exhibit challenging phenomena. Another possible strategy is switching between multiple linear operational ranges, i.e. instead of finding a global solution, numerous local ones are tediously developed and connected. These simplifications are often valid approaches that are successfully used in practice. Although choosing to work with simple models leads to an easier control design, which could be, however, insufficient in some cases. Therefore, systematic approaches that handle complex dynamical phenomena and situations are a field of active research.

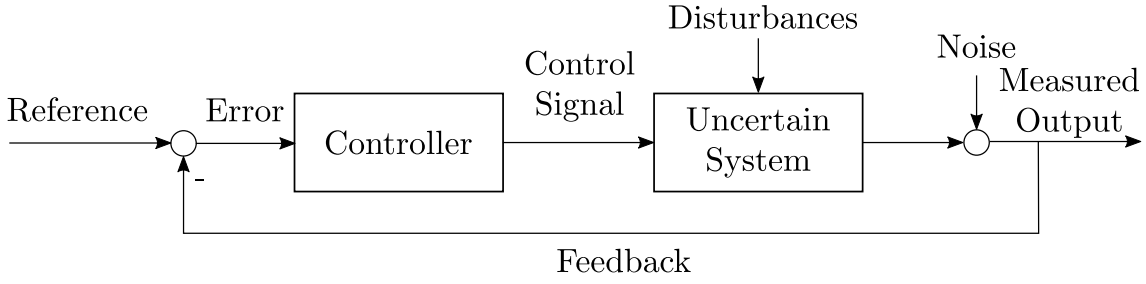


Figure 1.1: Basic block scheme of negative closed-loop control. The controller computes a control signal to steer the measured outputs to the desired references. Typically, the controller is based on the error between the outputs and references. In practice, disturbances influence the often uncertain system dynamics directly and noise influences the system outputs and thus the measurements.

From a systems point of view, the correct, safe and efficient operation of a system involves, first, defining and, second, guaranteeing the desired requirements. From a control point of view, the requirements need to be taken into account and achieved by the controller. One of the most distinctive characteristics of a control system is whether it operates in open- or closed-loop. Open-loop control is useful for some problems, such as manufacturing and low-level automation. In contrast, closed-loop control takes advantage of the output information through a feedback loop to fight the disturbances and the model errors, see Figure 1.1. In open-loop control, the feedback is missing. Key advantages of closed-loop control are the abilities to compensate disturbances, handle plant-model mismatch, the ability to reduce steady-state error, improve system performance, provide stability of the system and to react to unexpected changes. In this work, we consider systems from a simple control loop (Chapter 3 & 5), up to a plant-wide validation (Chapter 6).

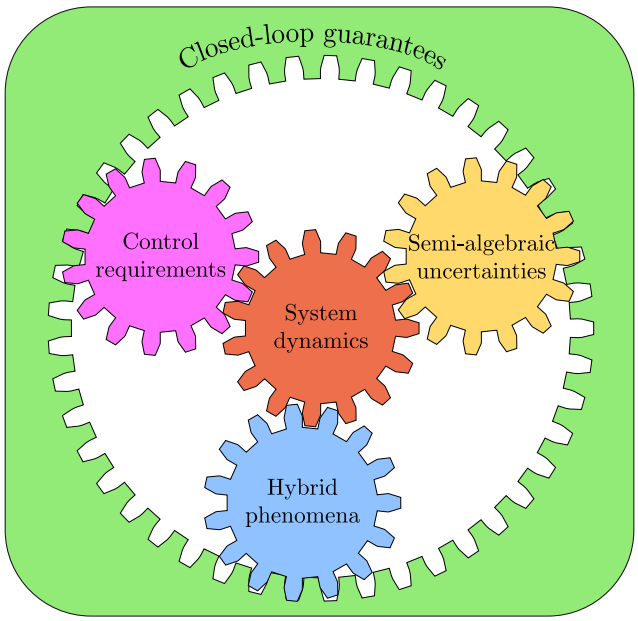


Figure 1.2: Considered key elements to achieve closed-loop guarantees.

The primary focuses of this work are the question of how to guarantee control requirements and how can they be formulated and validated for closed-loop system? We focus on uncertainties in the form of disturbances, noises, and model uncertainties. The complexity of the requirements and the examined system class led us to consider a hybrid system set-up.

Figure 1.2 shows the key elements of the proposed approach: the control requirements, the consideration of uncertainties described by semi-algebraic expressions, and hybrid dynamics. Each of these needs to be carefully considered and formulated. We aim to guarantee the performance taking all elements into account. The following sections introduce and discuss the main elements.

1.1 Uncertain Systems

"... Not doubt, certainty is what drives one insane...." [76]

Motivation

It is important to consider uncertainties explicitly when tuning a controller. Many practically relevant system phenomenon involve uncertainties such as measurement noise, disturbances, or plant-model mismatch. While it might be possible to construct models that incorporate these phenomena, it is, in general, challenging and laborious. Another arising challenge is that'..., the best ... model for a cat is another, or preferably the same cat.' [182]. We consider in this work that all system elements can be uncertain. Considering uncertainties in the parameters, states, outputs, inputs, references, initial conditions pose challenges but also provides opportunities. For example, by supporting the modelling effort by capturing some of the complex dynamics into uncertain auxiliary variables, allowing to reduce the model size significantly, see e.g. [193]. By explicitly considering these uncertainties, we can encompass behavioural variations and design robust controllers and even enforce resilience against unmodelled dynamics [217]. We adopt the notion that a controller is robust when it is able to handle uncertain systems [98], [149], [219].

Given a set of requirements, uncertainties and a fixed control structure, often, a set of controller parameters can provide the desired behaviour. The shape and volume of the feasible controller parameters set can be important information. In particular, one can analyse the feasible range of a particular control parameter and evaluate the sensitivity with respect to the performance and stability [196]. A narrow range of admissible controller values can point out the importance and sensitivity of the closed-loop with respect to a tuning parameter. Also, being able to determine the set of feasible parameters provides the advantage of having many controller parameterisations to choose from that deliver the desired behaviour.

Additionally, considering uncertainties, allows to take into account time-variant parameters. This allows to tune the controller not only for the nominal case but also for all possible uncertainties. Designing a controller robustly for the complete ranges of uncertain parameters avoids the need to re-design it if the system parameters deviate in the future. Moreover, one is often interested in providing guarantees of the tuning in spite of the system uncertainties.

Considered Uncertainties

Depending on the system at hand, the uncertainties can originate from various sources and have different influences on the system. We do not consider stochastic uncertainties and thus avoid the need for first obtaining and second dealing with probabilistic distributions. Indeed, such information is often neither common nor directly

obtainable from industrial sensors [86]. We consider uncertainties of the unknown-but-bounded type. In other words, each variable is considered bounded inside a set, and the so-called 'true value' or 'the measurement without noise' lies inside it.

A common approach to deal with set-bounded uncertain systems is through interval analysis [225]. We consider semi-algebraic sets, which allow for describing dynamical relations with both uncertain time-variant and time-invariant variables. Example sources for parametric uncertainties are confidence intervals from set-based observers, or from a system identification procedure [36], [50], [141], [142], [146], [153]. To understand how interval and semi-algebraic uncertainties differ, we present the following example:

Example 1. *Let us consider a two-dimensional example [20], shown in Figure 1.3. The non-convex black set illustrates a detailed shape that can be described through a semi-algebraic set, i.e.* $(\alpha, \beta) \in \mathcal{B}$*. In contrast, the smallest interval that includes this shape is the orange box, i.e.* $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, $\beta \in [\underline{\beta}, \overline{\beta}]$.

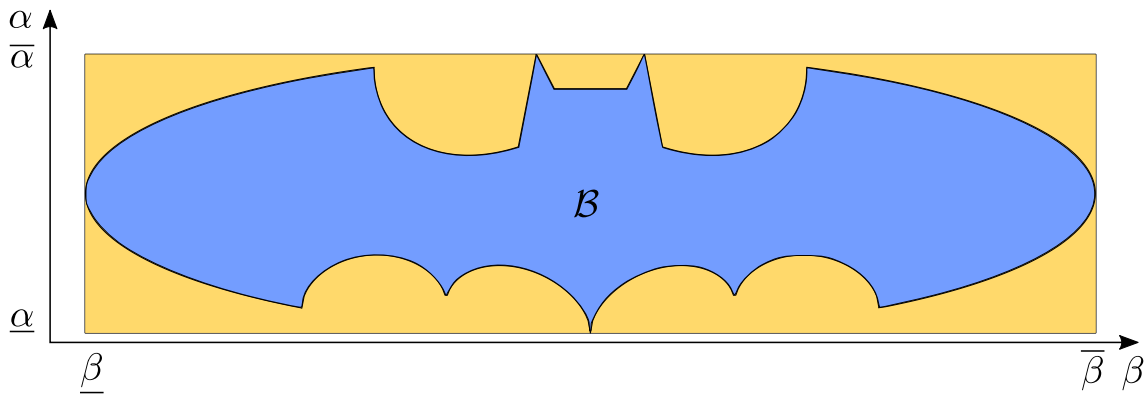


Figure 1.3: Geometrical comparison between a semi-algebraic, in blue, and the smallest interval box surrounding it, in orange.

In this work, we distinguish between spatial and temporal uncertainties. Spatial uncertainties relate to uncertainty in the amplitude of signals or parameters. In controller tuning, spatial uncertainty of the parameters is addressed in robust control strategies [239]. Requirements with temporal uncertainties allow to specify the desired behaviour to be achieved without fixing the exact execution time. Also, it allows to consider time-varying disturbances. Conditional requirements might occur during a control horizon multiple times or not at all. In contrast to the spatial uncertainties, the temporal uncertainties are considered less often when tuning a controller [24].

Set-based Controller Tuning

We consider so-called set-based controller tuning, which differs fundamentally from classical tuning. Classical tuning results in one specific value for each controller parameter, e.g. [161], [240]. Furthermore, in advanced control methods, like optimal

control, a controller is not tuned but a series of control values for each control input at each time is computed [179], [183]. In set-based tuning, we aim to obtain a set of controller parameters that guarantee the satisfaction of the process requirements [8], [9]. This set can be of two kinds, depending on the guarantees that it provides. The set can be a feasible set, which guarantees that all elements satisfy the system dynamics, constraints and uncertainties. The second type of a set of controller parameterisations, which we refer to as for-all set, guarantees that the controlled performance is achieved by any combination of uncertainties.

When tuning a controller for a system with uncertainties, there are many possible control scenarios. The ones discussed in this work can be crudely summarised into four categories. The first one is finding which of the uncertain values are feasible considering the system dynamics, initial conditions and constraints [18], [185]. The second one is tuning a controller such that it performs as desired despite all uncertainties, i.e. guaranteed robust control [8]. The third one is a mixture of the previous two. One example is controller tuning considering uncertain initial conditions, and another one is to determine how the choice of a reference influences the controller tuning [7], [194]. The fourth category deals with system monitoring and validation of the correct operation. These four categories are tackled by an invalidation approach for system performance.

These four scenarios can be seen as a robust form of control. We continue with an overview of robust control methods that are relevant to the system class we consider. Some of the classical robust control techniques are $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control, μ synthesis, Youla-Kucera parametrizations, small-gain theorem approaches [68], [79], [239].

Among the robust control approaches, there are different strategies based on the considered uncertainty type. In this work, we consider the unknown-but-bounded type of uncertainties. Their key characteristic is the strong differentiation between the valid (or safe) and invalid (or unsafe) regions of the variables. Such differentiation relates to bounded-error estimation and we refer the interested reader for more information on the basics and the development of this field to [18], [32], [141], [146], [160], [198], [227] and the references therein. Often bounded estimation, set-based methods, and set membership refer to the same class of approaches [142]. Here, we use set-based methods developed in, among others, [36], [186], [211].

One approach that deals with uncertainties and constrained control are target sets and target tubes [25]. They allow very similarly to the quantitative constraints considered in this work to specify a corridor of sets in time that needs to be visited. Another set-based approach that guarantees a constrained behaviour in the closed loop is funnel control [94], [95]. It defines a performance funnel in error coordinates by offsetting the output by the reference. Both target tubes and funnel control do not consider requirements with temporal or conditional uncertainties.

Specifying the system dynamics at each step is also possible through model predictive control (MPC). MPC can also take into account system uncertainties or hybrid

dynamics and even qualitative requirements [24], [23], [42], [170]. An MPC formulations that allow to specify a margin around the future evolution of the system are Tube-based MPC [139], [174], [235]. In contrast to MPC, we are interested in tuning a given controller structure and obtaining controller parameterisations that fulfil certain qualitative and quantitative requirements.

Another well-established approach to handle bounded uncertainties is through the use of interval analysis techniques [98], [147]. There are approaches that compute a stabilising controller through interval analysis, e.g. in the case of single-input single-output linear systems [131]. Interval arithmetic allows to compute the reachable space [121]. Interval methods have been used to tune PID controllers for linear systems, open-loop nonlinear control, and path planning [100]. By mapping the reachable set forward and backwards through inclusion functions, together with a set-inversion strategy, can lead to an inner approximation of the desired space, such that the constraints are held [97], [99]. However, interval methods might be challenged by the wrapping effects and dependency problems [98]. Another group of methods that can provide set-based guarantees are differential inclusions [14], [15], [16], [17], [90].

Due to the inherent uncertainties in the system, we need to look for suitable formulations of the system requirements.

1.2 Quantitative & Qualitative Robust Requirements

Engineering systems require guarantees with respect to physical limitations, safety considerations, and performance. All these constraints are forms of process requirements. We are interested in including in the controller tuning information that originates in a verbal form. For example, a requirement needs to be satisfied with respect to an action that is not specified precisely in time. Furthermore, one of the key requirements, guaranteeing the system stability, can be formulated in the form of keeping the system dynamics inside a safe region.

Figure 1.4 depicts some of the considered requirement types. Considering uncertain system renders exact enforcement of specific values non-meaningful. Due to uncertainties, constraints become set relations. To be able to ensure system requirements under uncertainties, we need to pose them in a suitable form. We organise them into quantitative and qualitative requirements.

Quantitative requirements express spatial conditions, i.e. the variables lie in specific ranges. They are often motivated by the classical transient response characteristics. We consider formulations on the overshoot, undershoot, settling time, steady-state error, rise time, delay time, and peak time. For each of these, we provide a set-based formulation.

In contrast to the quantitative requirements, the qualitative ones can be conditional, relative and are not fixed to specific time instances. They might furthermore contain temporal uncertainty. These requirements stem from different sources. On one side,

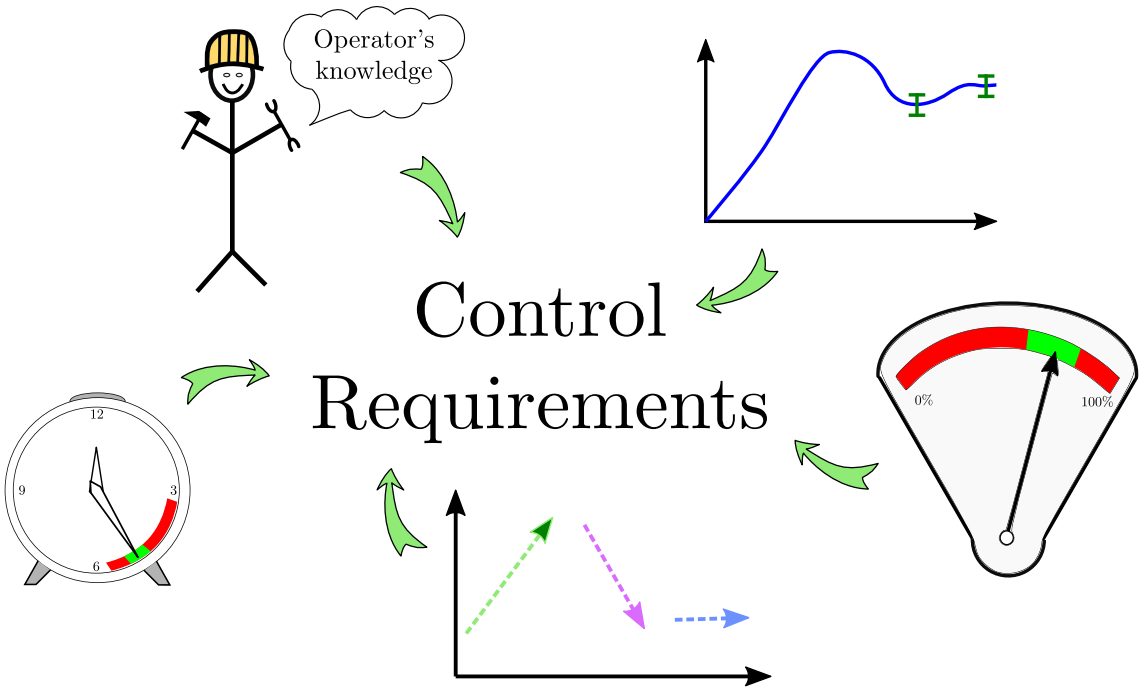


Figure 1.4: Different sources of requirements considered: verbal operator's knowledge, uncertain performance trajectory, safe amplitude ranges, logical and relative conditional requirements.

some systems might need to follow a desired behaviour that is not fixed in time, e.g. 'After inserting an agent in a chemical system, a particular marker needs to increase and after a period to decrease.'. When and how much it needs to increase and then to decrease is not fixed or important. The second source comes from the need to incorporate verbal operators like eventually, henceforth, next, until, and others. Such operators are not common in classical controller tuning. Examples are: 'Until one rotates this knob here and the other system is not working, first nothing will happen, but then a signal will go up, and eventually it will start to decrease.' To formulate such qualitative requirements, we consider Boolean algebra and operators from temporal logics.

Temporal logic has a long-standing history. A classical representative for temporal logic is the Linear Temporal Logic, proposed in 1977 by Pnueli in [171]. It considers the following four temporal operators *next*, *eventually*, *until* and *henceforth* [134]. Many other logics use similar operators or provide additional theoretical capabilities, e.g. Time Window Temporal Logic can specify the beginning and end time of the validity of a statement [224]. Signal Temporal Logic works directly with the signal value as a real number and does not require the segmentation of the state space [177]. Co-safe Linear Temporal Logic allows to check safety properties [108], [111]. Metric Temporal Logic provides a real-time framework [106]. Metric Interval Temporal Logic adds a bounded period of validity of the temporal operators to the Metric Temporal Logic [132]. Timed Propositional Temporal Logic adds a measure between two events [38].

Parametric Metric Interval Temporal Logic adds the ability to consider the beginning and end of the time intervals as parameters [58]. Computational Tree Logic and CTL* are branching temporal logics that can consider alternative time executions [52], [62]. Quantified Propositional Logic provides the ability to quantify over propositional atoms [89], [69]. The mentioned logics are just some of the more common temporal logics that exist, and many more are currently further developed and expanded [65].

All in all, quantitative and qualitative requirements can express and deal with spatial and temporal uncertainties, conditional and absolute amplitudes of the variables. Working with, in general, discrete-valued variables and, in particular, binary variables enables not only the formulation of a broader class of requirements but also provide the ability to express discrete-valued system description or hybrid dynamical systems.

1.3 Validation of Controlled Hybrid Systems

"...hybrids systems is such a wide notion that sticking to a single definition shall be too restrictive..."[223]

Considered Hybrid System Phenomena

An overarching characteristic of the hybrid dynamical system is that they can change their dynamical behaviour during operation [41], [120]. One example is the different growth stages in a controlled bio-reactor [148] or the different dynamics of an aeroplane during take-off in contrast to in-flight [48]. The resulting changes in the dynamics often require a change in the controller structure, controller settings, or both. Such systems are commonly referred to in literature either as **switched/switching** [120] or multi-mode systems [43]. Switching and switched system differ in whether we control the change or not. Switching allows to influence when to switch, and one can design rules based on state thresholds, timers, or a combination of both [120]. To do so, we propose a system description that allows to handle switched and switching systems, and also allows to tackle state- and time-dependent rules into account. We desire also to allow for **discontinuities** in the design [206].

One way to model switched dynamics is through the use of **discrete-valued variables** [24], [232]. Moreover, we can use discrete-valued variables to denote single entities or objects, e.g. in production systems, i.e. in discrete manufacturing, where separate items are processed or produced. A sub-class of the discrete-valued variables are the **binary variables**, which allow to model Boolean algebraic decisions. We use binary variables to formulate **process requirements** and special control scenarios, see Chapter 3. A switching behaviour can also be modelled by **time-variant parameters** [101]. This allows the explicit consideration of physical phenomena that can occur during longer times of operation, e.g. change of the values of friction coefficients in mechanical systems or conductivity properties in electrical systems [82]. One can

also work with a gain-scheduled set-up to obtain a switched system formulation [202].

Hybrid systems can also show **cyclic behaviour** [72], [138], [238], which a common scenario in the manufacturing systems, where many identical pieces need to be produced. Cyclic behaviour is not limited to manufacturing, e.g. pumping trajectories for a rigid-wing rotary kite [56]. Cyclic operation can be modelled, e.g. through Petri nets [55], [145].

While **constraints** are not a distinct characteristic of hybrid systems, constrained systems can be modelled as hybrid systems with different operating modes: one mode corresponds to the system operating inside the allowed zone - unconstrained, while the others would correspond to the system operating at the constraints. In this case, the switching conditions are state-dependent.

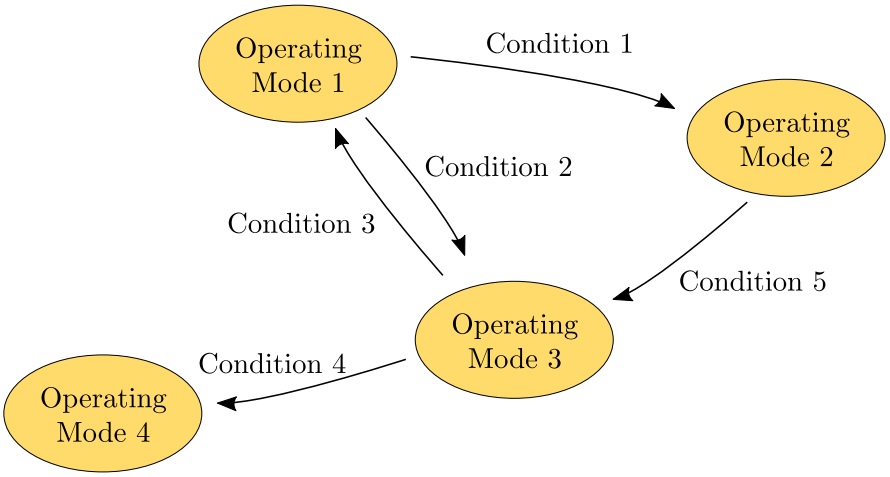


Figure 1.5: The behaviour of a hybrid dynamical system changes and it can be depicted as different operation modes. The switching between modes is governed by conditions and typically only certain transitions are possible.

Figure 1.5 shows the operation of a hybrid system due to different modes. A particular example of a hybrid system with multiple modes is autonomous driving [19]. Autonomous cars have modes in which dynamics, constraints, references and control strategies change. One can split the driving into different operation modes that switch from one to another, e.g. turn left, overtake, keep lane, and so on [26], [63], [67]. For more information on the topic of controlled hybrid systems, we refer the interested reader to the following sources and the references therein [12], [77], [125].

Behaviour validation

The two main requirements to the tuning process itself are to provide guaranteed results with respect to the requirements and to work with controller structures fitting to the considered system class. The guarantees need to hold despite the system uncertainties and be valid for the discussed hybrid phenomena, quantitative and qualitative

requirements. Moreover, the approach needs to be able to certify the desired requirements for all initial conditions and all references from given sets. We provide a short overview of validation methods for hybrid systems.

One class of validation approaches for hybrid systems are called formal methods for dynamical systems [233]. These methods are rather general and thus suffer from computational burden [21], [22]. 'Semi-formal' approaches allow verification but typically do not allow for controller design [181]. To validate systems, one can also search for barrier certificates. Barrier certificates allow to provide validity statements for hybrid systems with constraints [6], [64], [172], [234].

The so-called correct-by-design methods aim to design controllers which provide guarantees [215]. One way to do so is by Binary Decision Diagrams [140]. To find the control action such that the specifications are met, the input space and possibly the state or output space is quantised. The partitions are a finite amount of elements and approximate the dynamics. For guarantees, one can use a reachability projection - the space is over-approximated. Such abstraction of the space is a form of bisimulation, i.e. substituting the original system with another that serves as a surrogate one in the form of state transition system [188]. The two systems share behavioural similarity [143]. One specific similar abstraction technique is symbolic control [214], [216], [237]. Symbolic control experiences challenges with unmodelled disturbances and time-variant parameters [75], [122]. Similarly, approaches from computer science are used to verify the correctness of the programs and protocols, e.g. bounded model checking [29], binary decision diagrams [44]. One idea in model checking is to enumerate over the states and to check the correctness of the requirements based on the reachable states [61].

1.4 Contributions

In this work, we consider the task of tuning a controller. For this purpose, we use guaranteed set-based estimation methods and expand them for controller tuning of hybrid dynamical systems. The work is based and expands the set-estimation procedures that can be found [36], [186], [187], and [211]. A major focus of this work is the interplay of spatial and temporal specifications, uncertainties in the formulation, and validation of the control requirements. The main contributions of this work are:

- A systematic approach for guaranteed tuning for uncertain hybrid dynamical systems is provided. Some of the characteristics of the considered class are multi-input multi-output, constrained, discrete-valued variables, time-variant and time-invariant parameters, and allowing for uncertainties in each system element - states, inputs, references, outputs, parameters, disturbances, noises. The uncertainties can be spatial and temporal. The system can be multi-rate, multiplexed, event-based, includes preview information, and operates on a finite-time horizon.

Moreover, the dynamics can be nonlinear through the consideration of polynomial and rational expressions, and can also include algebraic dependencies. The approach allows controllers to be posed in polynomial and rational form.

- To address controller tuning requirement of the transient response of a system, a set-based formulation and discussion for each of the characteristics are provided. These requirements are referred to as quantitative.
- Qualitative control requirements that express relative dependencies and temporal uncertainties are considered. On one side, these requirements allow to include in the controller design information containing temporal operators. On the other side, they allow to express logical and relative amplitude conditions.
- Through extensions and further re-formulations of the quantitative and qualitative requirement, several control scenarios are presented. The major ones are disturbance rejection, bumpless control, deadbeat control, path following, and trajectory tracking.
- Simulation and experimental results demonstrate the applicability and performance of the tuning procedure. The applicability of the approach spans from a single control loop up to the validation of production plant behaviour. An implementation demonstrates the real-time capabilities of the approach.
- Set-based formulation and estimation results are provided for the robust error-free steady-state control under an endlessness requirement. Through this requirement, the system performance is guaranteed for an infinitely-long time through a finite-time approach. It is solved by tackling two control tasks. The first task provides robust set-point regulation for any reference from a reference set, where the system starts from any initial condition from a specified set. The second one delivers a form of robust controlled periodic set invariance.

1.5 Thesis Outline

In **Chapter 2**, we present the system class elements and their properties. In the end, we combine them in the controlled system set-up and outline some of the major challenges that we tackle.

In **Chapter 3**, we start by introducing the quantitative requirements and their inspiration - the transient response characteristics. We explain the behaviour and provide a set-based formulation for each of them. Based on the capabilities of the quantitative requirements, we demonstrate how to tune a controller for disturbance rejection, path following, or trajectory tracking. Afterwards, we introduce qualitative requirements and their characteristics. They express Boolean algebra expressions and can include temporal uncertainty operators that are inspired by the temporal logics.

With the help of the qualitative requirements, we bring two control scenarios - bumpless control and deadbeat control. We conclude the chapter with the formulation of discontinuity phenomena.

In **Chapter 4**, we outline the set-based estimation method that provides the guarantees of the obtained results. We start by posing the control problem in a feasibility problem formulation and deal with finding the feasible sets. This formulation aids the design process by excluding the infeasible solutions. Afterwards, we look at the constraint inversion formulation and how to use it to find a set of controller parameterisations despite all uncertainties.

In **Chapter 5**, we introduce the robust error-free steady-state control under endlessness requirement scenario, which we solve in three steps. The first one re-formulates the problem into error-coordinates to remove the layered uncertainty around the references. The second one tunes a controller to perform robust set-point regulation. The third one tunes possibly another controller to keep the system around the achieved reference by providing a form of robust periodic controlled set invariance.

In **Chapter 6**, we demonstrate how the presented approach can be used to validate the performance of large systems, e.g. the transportation system in discrete manufacturing. We abstract the system into sub-system elements and provide an approach to validate the performance without the need to consider the complete plant.

In **Chapter 7**, we summarise the presented work and bring some concluding remarks. Furthermore, we outline several next-step extensions.

2 Control of Uncertain Hybrid Systems

> Inside of every problem lies an opportunity.
>
> Robert Kiyosaki

In the systems and control literature, various processes are encompassed by the term 'hybrid dynamical systems' [78], [81], [105], [127], [176], [223]. For this reason, we present a formulation that covers a broad spectrum of hybrid systems and phenomena. For the system elements, we provide the description together with basic examples. Furthermore, we discuss the structure of switching strategies in controlled hybrid dynamical systems. We end the chapter with a summary of the system set-up and comment on the applicability of the presented system class.

2.1 System Class Description

We consider systems with the general description:

$$\begin{aligned} x(k^+) &= f(x(k), u(k), p(k)), \\ y(k) &= g(x(k), p(k)), \end{aligned} \tag{2.1}$$

where f and g are mixed-integer polynomial or rational functions. This structure encompasses a broad class of nonlinear dynamics. For approximating other types of nonlinearities and for multivariate systems, we refer the reader to [169]. The variables x are system states, p are system parameters, u are system inputs, y are system outputs, and k^+ is the subsequent time instance to $k \in \mathbb{N}$. This formulation allows for time-variant and time-invariant system parameters. We assume that the variables are contained within sets:

$$\begin{aligned} x(k) &\in \mathcal{X}^k \subset \mathbb{R}^{n_x} \times \mathbb{Z}^{m_x}, \\ y(k) &\in \mathcal{Y}^k \subset \mathbb{R}^{n_y} \times \mathbb{Z}^{m_y}, \\ u(k) &\in \mathcal{U}^k \subset \mathbb{R}^{n_u} \times \mathbb{Z}^{m_u}, \\ p(k) &\in \mathcal{P}^k \subset \mathbb{R}^{n_p} \times \mathbb{Z}^{m_p}, \end{aligned} \tag{2.2}$$

where, n_x, n_p, n_u, and n_y are the dimensions of real-valued and m_x, m_p, m_u, and m_y of the integer-valued variables and $n, m \in \mathbb{N}_{\geq 0}$. The integer-valued variables allow the consideration of a broader system class and accommodate the formulation of advanced process requirements, see Chapter 3, and hybrid phenomena, as discussed

in Section 1.3. We aim to achieve these requirements by tuning a controller with a fixed controller structure as described next.

Controller structure We consider that the controller structure is given in polynomial form

$$u(k^+) = h(y(k), r(k), u(k), c). \tag{2.3}$$

The controller has as an input both the reference signal $r \in \mathcal{R}^k \subset \mathbb{R}^{n_r} \times \mathbb{Z}^{m_r}$ and the system output y. Moreover, the controller states $u(k)$ are the system inputs from (2.1). The goal is to find controller parameters $c \in \mathcal{C} \subset \mathbb{R}^{n_c} \times \mathbb{Z}^{m_c}$ such that system (2.1) performs as desired. Both closed- and open-loop control structures are considered in this work. In the case where we have a controller in a non-polynomial or non-rational form, we can approximate it up to the desired precision. The resulting approximation error can be taken into account, e.g. either through an additive error term or by adjusting the uncertainty ranges of the variables.

Control horizon We focus on controller tuning for finite-time specifications. In the cases of a higher complexity on the requirements or with various execution stages of the process, we can consider splitting them into a series of pre-defined tasks with shorter finite-time periods. The example in Figure 2.1 illustrates how a process with different operating points can be split into a series of finite-time control tasks. Each of

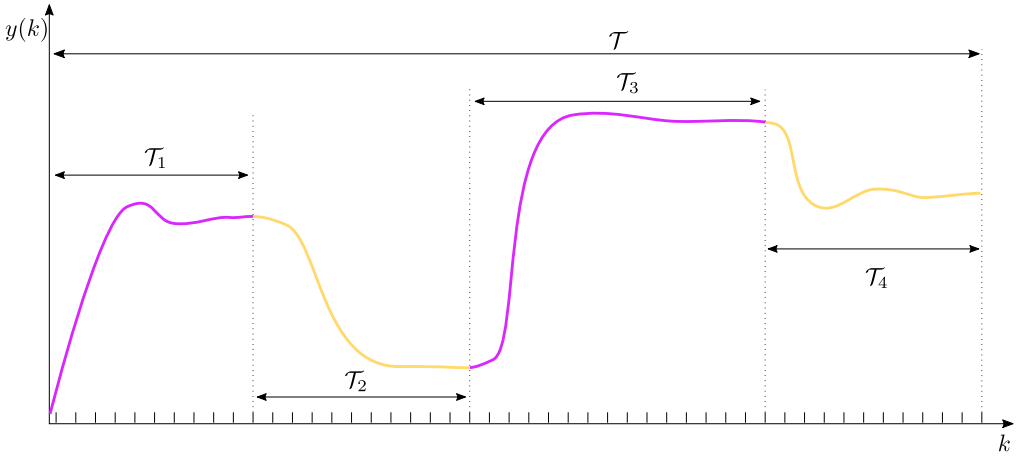


Figure 2.1: Requirements that are complex or span long or multiple horizons can be split into several shorter ones. The coloured segments of the output signal depict the four shorter finite-time control horizons. Each of the horizons $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$ corresponds to a single task, e.g. a reference change, or a period with expected high disturbance load.

the control horizons $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, $\mathcal{T}_4$ corresponds to a single set-point control regulation problem. Consider the following example:

Example 2. *The environment conditions inside (bio-)chemical batch reactors need to be controlled such that a desired yield is achieved. This is done by adjusting the*

chamber temperature, reactant feed rate, oxygen levels, or container pressure. Commonly, there is a known recipe that consists of several steps, each with a different range and duration of the control signals, e.g. the oxygen is switched between high and low concentrations to create aerobic or anaerobic conditions. Such recipe approaches are conventional in the process control industry [73] and can be seen in our context as several requirements over time.

Finite-time horizon set-ups can also be used to consider the control of periodic systems. In such systems, the initial and end conditions of the system have the same values, and during the control horizon, the states or outputs need to fulfil the process requirements. Examples for such systems are cyclic surveillance of an area [158], or maintaining the optimal trajectory of rigid-wing kites for producing electrical energy [56].

Often the system behaviour should change once certain conditions are met and thus at irregular time periods. To accommodate for such an event-based nature and the corresponding requirements, we consider that control horizon $\mathcal{T}$ spans a collection of time instances k_i:

$$\mathcal{T} \coloneqq \{k_0, k_1, ..., k_{n_t}\},$$

where $n_t \in \mathbb{N}_{>0}$. Additionally, we introduce $\mathcal{T}^- := \mathcal{T} \setminus \{k_{n_t}\}$, where the symbol $\setminus$ denotes set difference. The horizon $\mathcal{T}^-$ is used for those expressions that formulate the system dynamics. Constructing the horizon as a collection of instances allows to consider non-equidistant spaced sampling time as well as the classical equidistant discrete-time control. Unless stated otherwise, we introduce the next time instance as $k^+ \coloneqq k_{i+1}$, the previous time instance as $k^- \coloneqq k_{i-1}$, and remove the subscript index for the current time $k \coloneqq k_i$, $i = \{0, .., n_t\}$. In this sense, the order of these instances is $k^- \to k \to k^+$. This relative notion, compared to using absolute indices, is used later to express some qualitative requirements that have temporal uncertainty characteristics.

Considered Uncertainties We considered unknown-but-bounded uncertainties [141]. They are modelled as follows. A measurement instrument acquires the measured signal with a particular class of accuracy that certifies that the actual value lies within a specified range [152]. These uncertainty ranges can also come from system identification and regression models [133], [167], such as Gaussian kernels [173], support vectors [60], or artificial neural networks [54], [180]. In first-principle models, the parameters are related often to underlying physical laws that govern a considered system and thus, the bounded-uncertainty assumption is justified for many engineering systems.

Furthermore, we categorise the uncertainties into two different types, those that are present in the amplitude of the variables, i.e. spatial uncertainties, and those with an uncertain execution time, i.e. temporal uncertainties. For both types, we consider unknown-but-bounded uncertainties, c.f. Chapter 3. We refer to unknown-

but-bounded uncertainty as

Definition 1. Unknown-but-bounded (UBB) *uncertainties are drawn from closed sets containing all admissible values for a variable.*

Sources of uncertainties besides parameters can be system disturbances d and measurement noises s. State disturbances $d(k) \in \mathcal{D}^k \subset \mathbb{R}^{n_x} \times \mathbb{R}^{m_x}$ influence directly the system dynamics and thus the system states x. They can be either explicitly modelled and included additively to the system description, like

$$x(k^+) = f(x(k), u(k), p(k)) + w(x(k), d(k))$$

To model them explicitly, we need knowledge about the source of the disturbances. Depending on the particular system, it is possible to perform system identification and obtain a disturbance model $w(x(k), d(k))$ [155]. Alternatively, the influence of the disturbances can be added as time-variant variables $d(k) \in \mathcal{D}^k$ to each state.

Analogously, measurement noise $s \in \mathcal{S}^k \subset \mathbb{R}^{n_y} \times \mathbb{R}^{m_y}$ can be included explicitly to the dynamics through a model v or implicitly by adjusting $\mathcal{Y}^k$. The noise model v can be a random additive signal to the output or can have a state-dependent formulation [208]. Including v to the output formulation leads to

$$y(k) = g(x(k), p(k)) + v(s(k), x(k)).$$

We redefine f and g to include disturbances and noises:

$$\begin{aligned} x(k^+) &= f(x(k), u(k), p(k), d(k)), \\ y(k) &= g(x(k), p(k), s(k)). \end{aligned} \tag{2.4}$$

Additional to the signal uncertainties, we are interested in temporal uncertainties. We consider expressing them through either imprecise execution times or time-variant parameters, see Section 3.2 for more details. Moreover, time-variant parameters can be used to model the operating mode of the controller and system.

Requirements Every practically relevant system has constraints and requirements. They often originate from safety considerations, physical limitations, or preferred operating ranges. We formulate them as polynomial inequalities of the following form:

$$l(x(k), u(k), y(k), z(k), r(k), k) \geq 0, k \in \mathcal{T}, \tag{2.5}$$

where $z(k) \in \mathcal{Z}^k := \{0, 1\}^k$ are auxiliary binary variables used to express qualitative requirements. Formulation (2.5) allows to express control requirements like input/output constraints, system performance, input energy limits, and others. For brevity of notation, we refer to the i-th requirement as l_i. Due to the explicit consideration of uncertainties, control requirements in the form of equalities that demand achieving exactly a specific value are often questionable.

The formulation of the system requirements is one of the major contributions of this work. Accordingly, they are being discussed in detail, depending on the context, throughout the next chapters.

2.2 Control of Hybrid Dynamical System with Multiple Modes

Combining (2.4) and (2.3), we end up with a controlled hybrid dynamical system. In the hybrid system literature, a differentiation exists between switched and switching systems [120]. In the former, we have influence over the switching, e.g. an aeroplane or car gearbox. In the latter, we cannot manipulate the transition from one mode to another, e.g. a cell mutation or a system fault. The discussed system description is suitable for both cases, and whether the change in modes is intentional depends on the task at hand.

The information from the controllers or the active governing system can be included in aggregated models, as seen next. The aggregate controller model Ξ consists of controllers Ξ_i. Each Ξ_i defines a combination of a controller structure and a particular parameterisation. We express this structure by posing it as a (weighted) sum of the outputs for each controller Ξ_i, i.e.

$$\Xi = \sum_{i=1}^{n_a} a_i \Xi_i,$$
$$\sum_{i=1}^{n_a} a_i = 1.$$

Depending on the choice of a_i, one can describe two switching scenarios: abrupt and smooth switching. When the switching is done abruptly, only one of the controllers is responsible for the input at each time step, i.e. $a_i = \{0, 1\}$. We present an example of such a case in Section 4.3.3, which deals with the constant-current constant-voltage charging cycle of a Li-ion battery [203]. Figure 2.2 illustrates an abrupt switching scheme.

In contrast to the abrupt switching, smooth switching blends two or more controller output signals, and thus the weighting coefficients are chosen as $a_i = \mathbb{R}_{[0,1]}$, e.g. control of biotechnological processes when switching from one nutrient to another, or from one environmental condition to another [148]. Analogously, the same switching strategies can be used to formulate the system dynamics:

$$\Omega = \sum_{i=1}^{n_b} b_i \Omega_i,$$
$$\sum_{i=1}^{n_b} b_i = 1,$$

where Ω_i contains of system description for the particular mode.

A straightforward way of modelling switching behaviour is combining the modes

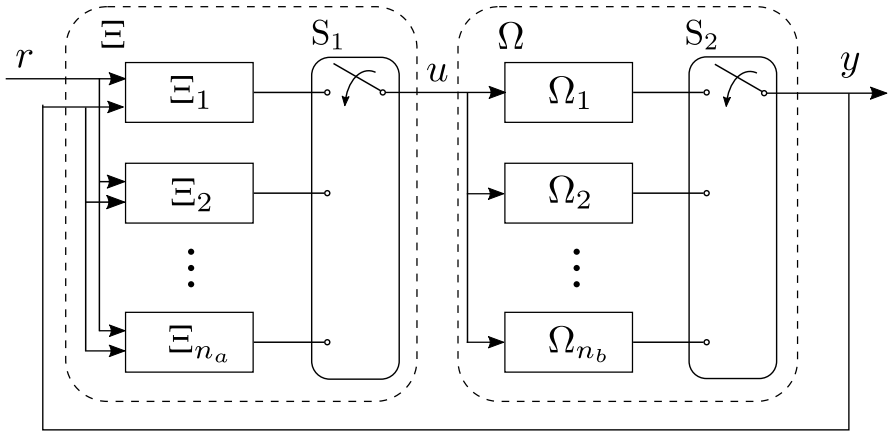


Figure 2.2: A sketch of a closed-loop controlled switched system. Both the controller Ξ and the system Ω have more than one operation modes. The change between them is done abruptly and depicted by the two switching blocks S_1 and S_2.

by introducing auxiliary binary variables that enable the current mode and disable the rest [195]. Doing so, we end up with an aggregated description for the system dynamics and the control law. In some cases, the switching is dependent only on time [120]. In other cases, we have to include additional information to form a decision logic. In this work, we assume that the switching rules are given, and the design of the switching block is outside the scope of this work. We continue with combining the presented system elements into the control parameterisation set-up and pose the key problems we are interested in.

2.3 Overall Set-up

Combining the system dynamics (2.4), controller structure (2.3), system uncertainty description (2.2), and control system requirements (2.5), we obtain the following total set-up:

$$\begin{aligned}
&x(k^+) = f(x(k), u(k), p(k), d(k)), k \in \mathcal{T}^-,\\
&y(k) = g(x(k), p(k), s(k)), k \in \mathcal{T},\\
&u(k^+) = h(y(k), r(k), u(k), c), k \in \mathcal{T}^-,\\
&l(x(k), u(k), y(k), z(k), r(k), k) \geq 0,\\
&x(k) \in \mathcal{X}^k, p(k) \in \mathcal{P}^k, r(k) \in \mathcal{R}^k,\\
&u(k) \in \mathcal{U}^k, y(k) \in \mathcal{Y}^k, s(k) \in \mathcal{S}^k,\\
&d(k) \in \mathcal{D}^k, z(k) \in \mathcal{Z}^k, c \in \mathcal{C}.
\end{aligned} \tag{2.6}$$

This set-up allows to consider a broad class of dynamical behaviours. We use (2.6) as a general formulation, which allows to cover a wide range of control scenarios. We aim to solve the following task:

Problem 1. *(Controller requirement formulation) For the system described in* (2.6)*, derive a systematic approach to express the control requirements in the form of* (2.5)*.*

The requirements in Problem 1 encompass both classical tuning criteria, e.g. transient response characteristics, and empirically obtained engineering knowledge, e.g. from oral descriptions like 'eventually', 'if', 'until', and others. To answer Problem 1, we outline the descriptions of the control requirements in Chapter 3.

Due to the broad system class, obtaining the set of controller parameters analytically is a non-trivial task. We propose to obtain a set-based approximation. Basically, we are interested in the following problem:

Problem 2. *(Guaranteed controller parameterisation) Obtain an estimation set* $\hat{\mathcal{C}}$ *of the set of consistent controller parameter values* c^* *that satisfy* (2.6)*.*

Selecting particular values $c^* \in \hat{\mathcal{C}}$ the controlled system performs as desired. In Chapter 4, we provide details on a method that obtains $\hat{\mathcal{C}}$ and present different approximation formulations to address the task at hand. Because of the general set-up and the chosen set-based estimation approach, we can formulate various control scenarios. These scenarios are created by adding new or modifying existing requirements. Next, we continue with the discussion of the types of requirements and their formulation based on whether their uncertainty aspect is in the spatial or in the temporal domain. To illustrate the applicability of the constraints and their construction, we provide examples.

3 Control Requirements - A Set-based Perspective

> A theory that you can't explain to a bartender is probably no damn good.
>
> Sir Ernest Rutherford

Requirements on the control system can originate from various sources, e.g. technical specifications, experience from working with the system, energy efficiency, pollution reduction, etc. In this chapter, we discuss different types of controller requirements and how to formulate them in a structured way. They need to be applicable to the considered class of hybrid dynamical systems. On the other side, they need to respect and take into account the present uncertainties, due to noises, disturbances, plant-model mismatch, etc. Thus, we aim for a 'robust' formulation in which to postulate the requirements.

We consider the following two types of requirements: quantitative and qualitative. Quantitative requirements enforce variables to take special values, possibly in a certain range. Examples are set points that should be tracked precisely. In contrast to the quantitative, the qualitative requirements do not enforce specific values at specific times. Rather, they demand certain behaviour, such as being error-free after some time. Moreover, we use them to formulate relative or conditional value dependencies. Qualitative requirements allow to take logical conditions or even verbally formulated relevant information into account, e.g. statements that contain operators like *eventually*, *next*, etc.

We begin by focusing on quantitative requirements, concentrating on enforcing transient response characteristics. Secondly, we focus on Boolean logic expressions and temporal logic predicates that are motivated by the consideration of qualitative requirements. The results in this chapter build upon our works [7], [8], [9]. In addition, we discuss disturbance rejection, bumpless control, trajectory tracking, path following, deadbeat control and how to handle discontinuities in the variables.

3.1 Quantitative Requirements

We consider quantitative requirements, i.e. a range of admissible values at fixed time instances. We consider the following definition for quantitative requirements:

Definition 2. *Quantitative requirements are system requirements that bound the admissible values of a time-variant variable ψ to a specified semi-algebraic set Ψ at a time-instance k, i.e. $\psi(k) \in \Psi^k$. In the case that ψ is time-invariant, then $\psi \in \Psi$.*

Quantitative requirements relate to classical controller tuning by analysing the dynamical properties of the output system response after a step input. Those properties are referred to as transient response characteristics [103].

3.1.1 Robust Transient Response Characteristics

A classic approach to evaluate the dynamical performance of a system is by the application of a step input to the system and analysing the resulting system output. Based on the resulting output profile, there are common dynamical characteristics that characterise the system behaviour [59], [117], [162]. Figure 3.1 illustrates several step responses and depicts some of the transient response characteristics.

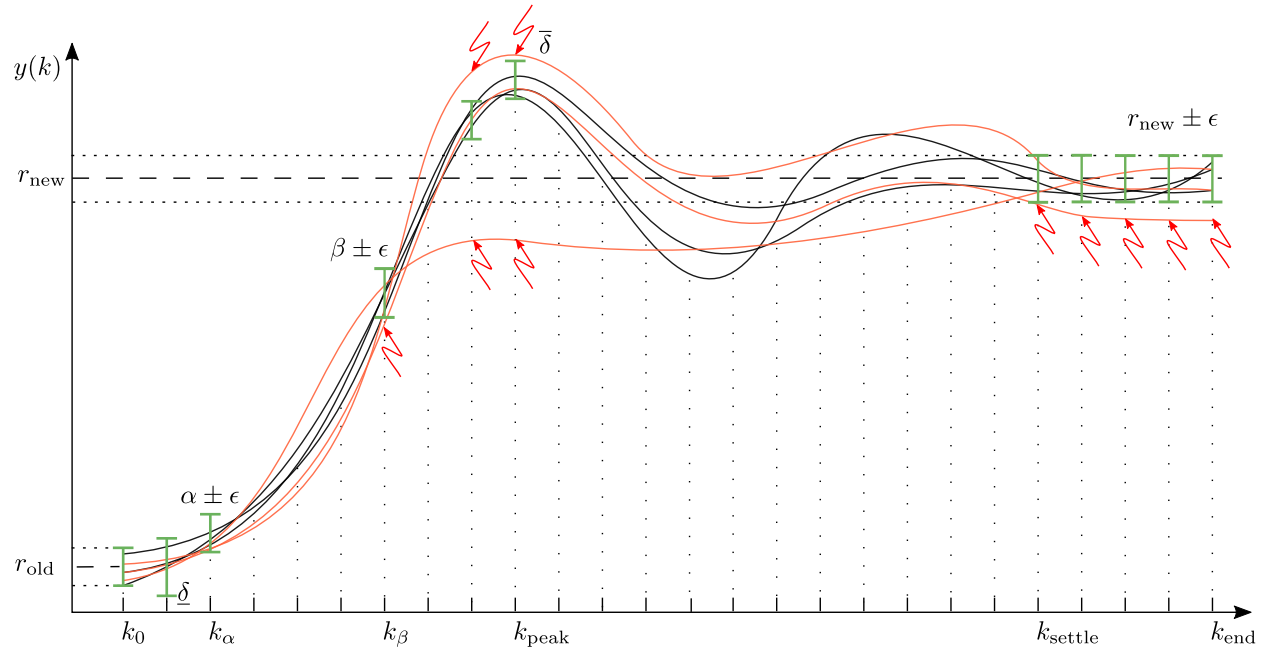


Figure 3.1: System step responses for various initial conditions and different control parameterisations. The double T-bars (in green) define the desired regions that the system's response needs to satisfy. The trajectories in orange fail at satisfying at least one of them, where the black trajectories satisfy all of them. The red lightning symbols point to where the constraints are violated.

Transient response requirements have been studied widely, and analytical tools exist for linear systems [168]. In contrast to the linear case, achieving transient response properties for nonlinear systems remains non-trivial. Therefore, to take into account the uncertainties, e.g. from noise, one can not aim to achieve exact values. One needs to rather relax this expectation towards achieving set bounds, considering the UBB uncertainty.

Although the set-up (2.6) allows for multi-output systems, for illustrative purposes, each of the characteristics is presented for a single-output system. We take a separate look into the basic description and the robust form for each of the following transient response characteristics: overshoot, peak time, rise time, delay time, settling time, and steady-state error. Often, two or more characteristics are imposed simultaneously [59]. Formulating the requirements separately is not limiting, as we can impose later on a combination of them onto the corresponding outputs.

Overshoot is the behaviour when the output signal exceeds a desired reference value r_{new}. As a specific example, consider the control of cranes, where while moving a suspended cargo, significant sways, for safety and performance considerations, must be contained below a certain level [163]. The overshoot can be characterised as the difference between the absolute maximum value of the output signal $y_{\max}$ and the new reference r_{new} over a time $\mathcal{T} = \{k_0, ..., k_{\text{end}}\} \subset \mathbb{N}_{\geq 0}$, where

$$y_{\max} = \max\{y(k)\}, y(k) \in \mathcal{Y}^k, k \in \mathcal{T}.$$

We present two mathematical formulations of overshoot requirements. The first one sets a global constraint for the maximum output value at all time instances:

$$y(k) < r_{\text{new}} + \overline{\delta}, k \in \mathcal{T}, \quad \overline{\delta} \in \mathbb{R}_{\geq 0}.$$

The second formulation imposes the restriction locally in time:

$$y(k) < r_{\text{new}} + \overline{\delta}, k \in \{k_j, ..., k_{j+l}\},$$

where $l \in \mathbb{N}_{\geq 0}$, $0 \leq l \leq k_{\text{end}} - k_j$ defines the period during which the output is constrained from above. In short, we formulate the overshoot requirement either as a global bound, i.e. imposed for the entire horizon or locally. The overshoot parameter $\overline{\delta}$ needs to be chosen such that it is coherent with the present uncertainties. Alternatively to an absolute value formulation, a relative formulation, depending on other process variables exists, which can be formulated as a qualitative requirement, see Section 3.2. Note that there could also be the requirement for a minimum overshoot, e.g. is in some deadbeat response formulations [59].

In summary, the design parameters for the overshoot requirement are $\overline{\delta}$, k_j and l. Additional to the maximum value of the overshoot, the time when the maximum is achieved can also be specified, which is referred to as the peak time and presented next.

Peak time is the instance at which the output y has its maximum value. Examples are the design of an autopilot of a jet fighter and the distance control between vehicles [59].

The peak time is defined as:

$$k_{\text{peak}} = \underset{k}{\text{argmax}}\, y(k), k \in \mathcal{T}.$$

One can specify time k_{peak}, as follows

$$\begin{aligned} &y(k_{\text{peak}}) \in \mathcal{Y}^{k_{\text{peak}}}, \\ &y(k) \in \mathcal{Y}^{k}, k \in \mathcal{T} \backslash \{k_{\text{peak}}\}, \\ &y(k) < y(k_{\text{peak}}), k \in \mathcal{T} \backslash \{k_{\text{peak}}\}. \end{aligned}$$

Peak time requirements should be cautiously considered, as it might lead to unsatisfactory results. For example, the output noise could be responsible for producing a false signal peak and thus a false peak time. This is even more critical when two sampling instances are close to each other, which can result in wrongly interpreting the temporary variation of the signal. In such cases, a more suitable approach is to impose the requirement for a desired peak time over a series of instances $k_j, ..., k_{j+l}$:

$$y(k_{\text{peak}}) > y(k), k_{\text{peak}} \in \{k_j, ..., k_{j+l}\}, k \in \mathcal{T} \backslash \{k_j, ..., k_{j+l}\}.$$

Considering the period $\{k_j, ..., k_{j+l}\}$, one can counteract the risk that the peak value is a result of short disturbance. The design parameters for the peak time requirement are k_{peak}, k_j, k_{j+l}, and $\mathcal{Y}^{k_{\text{peak}}}$.

Rise time expresses the time period the system needs to reach a threshold β at a time k_β, starting from a value α at time k_α, i.e. $\alpha < \beta$ and $k_\alpha < k_\beta$. Application examples are the control of wind turbines, and the shaping of the response of robotic arms [59], pressure level control [236], and automatic voltage regulation [159]. Common pairs for $\{\alpha, \beta\}$ are $\{10\%, 90\%\}$ and $\{0\%, 100\%\}$ on a relative scale [59]. To accommodate the inherent uncertainties, we introduce a coefficient ϵ. The rise time can be thus be formulated by:

$$\begin{aligned} y(k_\alpha) &\geq \alpha - \epsilon, \\ y(k_\alpha) &\leq \alpha + \epsilon, \\ y(k_\beta) &\geq \beta - \epsilon, \\ y(k_\beta) &\leq \beta + \epsilon. \end{aligned}$$

Note that, $\beta - \epsilon > \alpha + \epsilon$. We can use this information and formulate it in the controller design by defining the duration of the rise time itself by specifying both times k_α and k_β:

$$k_{\text{rise}} = k_\beta - k_\alpha$$

The design parameters for the rise time requirement are k_{rise}, k_α, k_β, ϵ, α, and β.

Delay describes the time until a system starts noticeably increasing its output amplitude after a change in reference. Examples are the tiltrotor control of an unmanned

aerial vehicle [113]. Common threshold values α for the system output y are 10% and 50% of the absolute reference value r [162]. To formulate the delay time requirement, we can impose a lower boundary on all time instances after the desired time k_α.

$$y(k) \geq \alpha, k \in \{k_\alpha .., k_{\text{end}}\}, \alpha \in \mathbb{R}_{\geq 0}.$$

In this case, the output $y(k)$ must be above the precise threshold α after k_α. Alternatively, for a smoother start, we can impose, e.g. an upper limit during the initial period of $\{k_0, .., k_\alpha\}$:

$$y(k) \leq \alpha, k \in \{k_0, ..., k_{\alpha-1}\}.$$

In general, this requirement is present for systems that need time until they start a more rapid increase in the output amplitude, e.g. higher-order systems or systems with delay. The design parameters for this requirement are k_α and α.

Settling time & steady-state error are two characteristics that are closely associated. The settling time k_{settle} marks the period the system output $y(k)$ takes to reach and afterwards stays inside a bounded region around a new reference r_{new}. This bounded region is specified with a desired precision ϵ. This precision determines the amplitude of the steady-state error. Typical values that are considered for the steady-state error ϵ are either 2% or 5% of the reference change [59]. The steady-state error requirement can be posed through the following constraints:

$$\begin{aligned} y(k) &\leq r_{\text{new}} + \epsilon, \quad k \in \{k_{\text{settle}}, .., k_{\text{end}}\}, \\ y(k) &\geq r_{\text{new}} - \epsilon, \quad k \in \{k_{\text{settle}}, .., k_{\text{end}}\}. \end{aligned}$$

In general, by increasing the allowed steady-state error ϵ, more controller values can be able to satisfy the relaxed precision and achieve the new reference faster. In summary, the design parameters for settling time and steady-state error requirements are the k_{settle} and ϵ.

3.1.2 Remarks on the Transient Characteristics

In addition to the more common characteristics, one can also consider an *undershoot* requirement. Undershoot describes the behaviour where the output changes in the opposite direction of the step input, and later on, it achieves steady-state [40]. For example, if the new reference r_{new} is bigger than the current one r_{curr}, i.e. $r_{\text{new}} > r_{\text{curr}}$, then the output $y(k)$ decreases before rising, i.e. $y(k_0) > y(k)$ where k is among the first time steps after the change occurs. Many systems, such as servomechanisms exhibit this behaviour [144]. For linear systems, this corresponds to having zeros in the right half s-plane [116].

Combining the settling time and the steady-state error leads to an often desired dynamical behaviour of steady-state error-free control [59]. We extend this behaviour

for all initial conditions and any reference from a desired set for an infinitely long time in Chapter 5. In linear control, it is common knowledge that having an integral controller eliminates the steady-state error in the long run [231]. For nonlinear systems, as considered here, guaranteeing small steady-state errors is a non-trivial task [119] and becomes even more challenging for multi-input multi-output systems [46], [212].

Using the presented formalism, we continue with three control scenarios that can be expressed through quantitative statements, namely, trajectory tracking, path following, and disturbance rejection.

3.1.3 Formulating Quantitative Control Scenarios

Set-based trajectory tracking

One control scenario that follows directly from the capability of enforcing constraints at the time steps is trajectory tracking. We consider the problem of specifying a trajectory tracking corridor for a single-output system, imposing the desired behaviour by a series of sets $\mathcal{R}^k$, c.f. Figure 3.2. Alternatively, the trajectory can be given as a time-variant reference $r(k)$ and a desired precision $\epsilon(k) \in \mathbb{R}^+$. Typical applications of trajectory tracking are quadrotor helicopter control [88] or synchronisation problem [213].

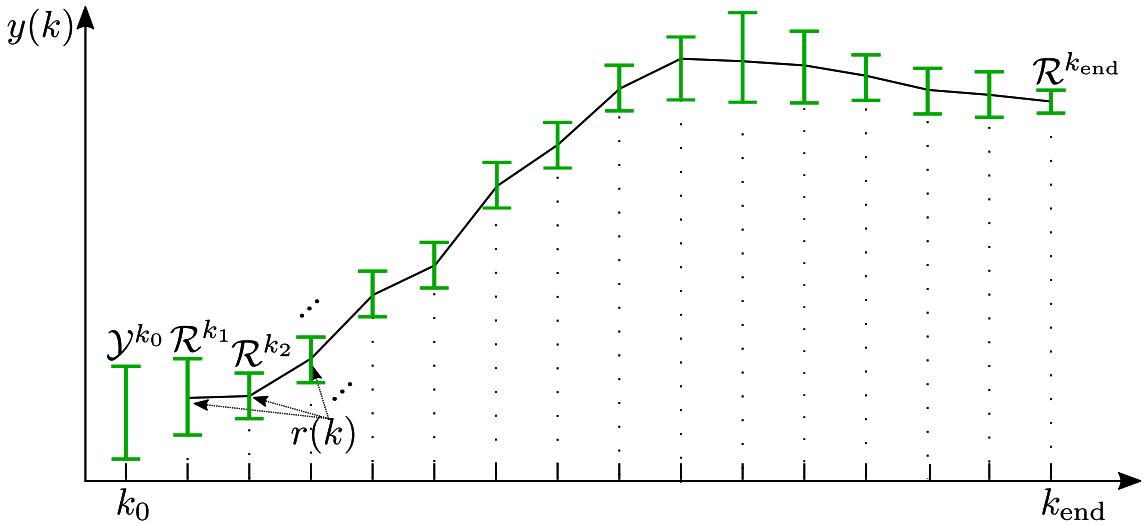


Figure 3.2: Trajectory tracking example, where the admissible trajectories are limited ot the ranges given by the green bars $\mathcal{R}^k = \{r(k) - \epsilon(k), r(k) + \epsilon(k)\}$.

Trajectory tracking can be expressed as bounds on admissible values at each time instance. They can be posed through the following constraints:

$$\begin{aligned} y(k) &\geq r(k) - \epsilon(k), k \in \mathcal{T} \backslash k_0, \\ y(k) &\leq r(k) + \epsilon(k), k \in \mathcal{T} \backslash k_0. \end{aligned} \tag{3.1}$$

Here k_0 is excluded, as it provides the initial conditions. We are interested in generalising this formulation following the definitions provided in [4], and therefore we pose

the following robust form of the trajectory tracking problem:

Problem 3. *(Set-based trajectory tracking) For the controlled system* (2.6) *with initial conditions* $\mathcal{Y}^{k_0}$*, obtain controller parameterisation set* $\hat{\mathcal{C}}$*, such that the system outputs* $y(k)$ *track a given trajectory* $\mathscr{T}^k$*, i.e.* $y(k) \in \mathscr{T}^k$ *,* $k \in \mathcal{T}\backslash\{k_0\}$*, for all parameters of their corresponding sets.*

Basically, the trajectory $\mathscr{T}^k$ constraints are added to (2.6). To satisfy the above constraint, we require $\mathscr{T}^k \subseteq \mathcal{Y}^k$, $k \in \mathcal{T}\backslash\{k_0\}$. At the initial time k_0, the output is within the range $y(k_0) \in \mathcal{Y}^{k_0}$. Overall, Problem 3 leads to the following set constraints for the outputs:

$$\begin{aligned} &y(k_0) \in \mathcal{Y}^{k_0}, \\ &y(k) \in \mathscr{T}^k, k \in \mathcal{T}\backslash\{k_0\}. \end{aligned} \tag{3.2}$$

Formulation (3.2) generalises (3.1), as it allows for a general semi-algebraic multi-output formulation.

To alleviate the computation and allow for parallel processing, one can split the trajectory into segments, e.g. each with a length of $d+1$ time instances. To do so, the last instance of the previous segment is part of the next segment. The obtained set of controller parameters $\tilde{\mathcal{C}} \subseteq \mathcal{C}_{\text{init}}$ for each segment are intersected, to obtain $\hat{\mathcal{C}}$. The resulting set $\hat{\mathcal{C}}$ contains controller parameterisations that achieve the behaviour in Problem 3. The final approach using segments is formalised in Algorithm 1.

Algorithm 1 Obtaining controller parameterisation for robust trajectory tracking by segmentation

Input: $\mathscr{T}^k, \Omega, \Xi, \mathcal{C}_{\text{init}}, d, \mathcal{Y}^{k_0}$
Output: $\hat{\mathcal{C}}$

set $\tilde{\mathcal{Y}} = \mathcal{Y}^{k_0}$, $\hat{\mathcal{C}} = \mathcal{C}_{\text{init}}$
set $n =$ size of $\mathscr{T}$
set $i = 1$
while $i + d \leq n$ **do**
  based on Ω, Ξ, $\tilde{\mathcal{Y}}$ obtain $\tilde{\mathcal{C}}$, s.t. $\mathcal{Y}^l \subseteq \mathscr{T}^l$, $l = \{i, .., i+d\}$
  update $\hat{\mathcal{C}} = \hat{\mathcal{C}} \bigcap \tilde{\mathcal{C}}$
  set $\tilde{\mathcal{Y}} = \mathscr{T}^{i+d-1}$
  set $i = i + d$
  if $i + d > n$ AND $n - i \geq 1$ **then**
    set $d = n - i$
  end if
end while

In Algorithm 1, $d \in \{1, .., n\}$ denotes how many steps from the trajectory should be computed at once. Obtaining a guaranteed set $\mathcal{Y}^l$ is the focus of Chapter 4. Note

that, Algorithm 1 does not guarantee that one obtains a non-empty set $\hat{\mathcal{C}}$. Obtaining an empty set could indicate too tight precision, specified by the sets $\mathscr{T}^k$, which can be an indication of a challenging trajectory to track. In such cases, the strategy can be used iteratively, i.e. starting with a narrow trajectory corridor $\mathscr{T}^k$, and if no feasible controller parameters are found, then one can relax the allowed regions until valid parameterisations are found.

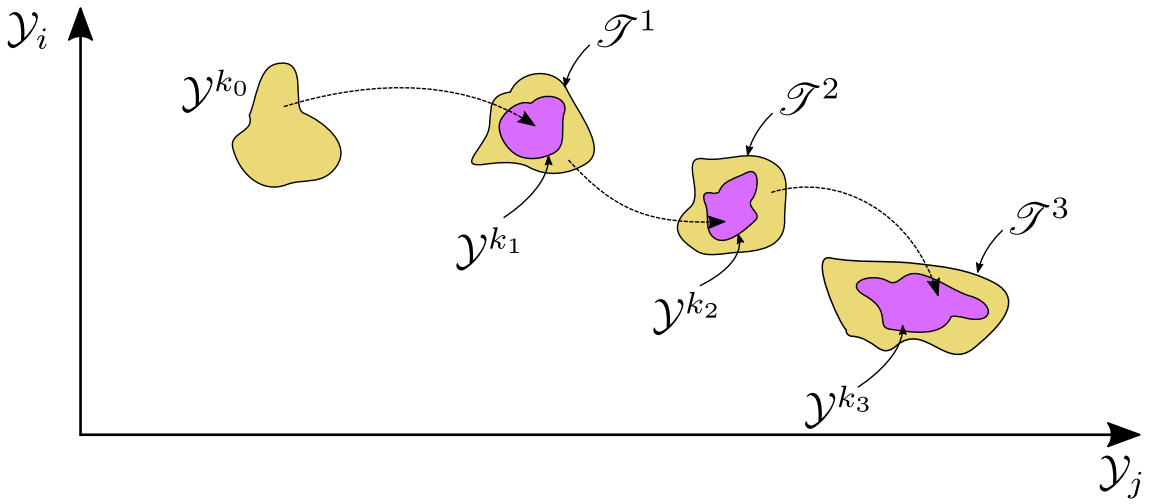


Figure 3.3: Illustration of trajectory tracking, employing the segmentation approach given in Algorithm 1.

Two other control problems that can be expressed as robust trajectory tracking are contract-based control and plug and play control. In the former, each system communicates the predicted bounds of the coupling signals to the connected systems, such that a control objective is guaranteed [189]. While in the latter, the goal is to substitute a system with another one while guaranteeing performance. Thus, the behaviour of the to-be-exchanged-system can be specified by the bounds at each step and later provided by plugged in system [124], [209].

Set-based path following

A closely related scenario to trajectory tracking is the task of path following. In it, a curve $\mathscr{P} \in \mathcal{Y}^k$ referred to as a path, and it should be followed precisely by a system Ξ. In contrast to trajectory tracking, the geometric path is not fixed in time. The focus lies on following the path precisely. An application for path following is the control of autonomous underwater vehicles for precise ocean surveillance [109].

Path following can be expressed as follows:

Problem 4. *(set-based path following) For the system* (2.6) *and initial conditions* $\mathcal{Y}^{k_0}$*, obtain the set of controller parameters* $\hat{\mathcal{C}}$*, such that the system outputs* $y(k)$ *follows the desired path* $\mathscr{P} \in \mathcal{Y}^k$ *within a specifiable precision.*

Problem 4 is formulated in the output space, which also allows a state-space formulation. Note that the path is not time parameterised, i.e. the speed with which the system follows it is not prespecified. Therefore, path following puts the focus on keeping the system close to the path, independent from the time. This could lead to the pathological case where the system halts at a point on the path, i.e. it does not move along it any longer. A common way to overcome this hurdle is by adding a virtual 'time state' to the problem, which can be considered as a requirement on the states [137]. To illustrate this, consider the following example:

Example 3. *Imagine greyhounds racing along a track. The position of greyhounds symbolise the system output, and the track is the path along which the system needs to follow. Once the trap doors are opened, the dogs would just stay there, but if there is an artificial lure that moves along the track, the hounds start chasing it. This artificial lure lets the system output to move forward along the path.*

To reduce the number of requirements, one can, as in the trajectory tracking case, split the path $\mathscr{P}$ into η segments. Except for the first segment, the beginning point of each segment is the end of the previous. We denote these waypoints with $\mathscr{P}^i$. In doing so, the path $\mathscr{P}$ is abstracted by a sequence of waypoints $\{\mathscr{P}^1, \mathscr{P}^2, .., \mathscr{P}^\eta\}$. To each of the path points, one can add a safety region $\mathscr{B}$, i.e. $\mathscr{R}^i = \mathscr{P}^i \oplus \mathscr{B}$.

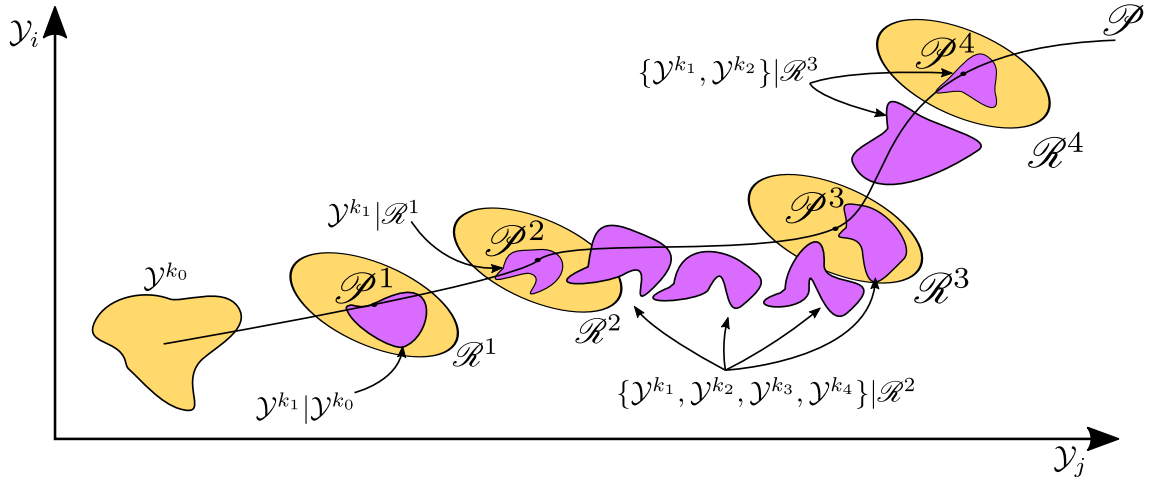


Figure 3.4: Segmenting the requirements for path following. The path $\mathscr{P}$ is segmented in a series of waypoints $\mathscr{P}^i$, which with their safety region $\mathscr{B}$ result in the sets $\mathscr{R}^i$. The time needed to reach two consecutive waypoints $\mathscr{P}^i$ can differ, e.g. $\mathscr{P}^2$ is reached in one step, where between $\mathscr{P}^2$ and $\mathscr{P}^3$ four instances are needed.

Keeping in mind the challenge of moving forward along the path, we approach this problem by looking for controller parameterisations $\tilde{\mathcal{C}}$ that steer the system outputs $y(k)$ from one waypoint $\mathscr{P}^i$ to the next one $\mathscr{P}^{i+1}$. We leave as design freedom the choice of the maximum time $k_{\max}$ that the system needs to achieve the next consecutive waypoint. We start by checking the existence of controller parameterisations such that one step, i.e. $k = 1$, set inclusion is guaranteed, i.e. $\mathcal{Y}^k|\mathscr{R}^{i-1} \subseteq \mathscr{R}^i$, where the symbol $|$

should be read as *starting from*. If no controller parameterisations are found, then the control horizon k is increased until a parameterisation is obtained. This local increase leads to time dilation, i.e. a different amount of time is needed in the different path segments, see Figure 3.4. This solution approach is structured in Algorithm 2.

Algorithm 2 Controller parameterisation and segmentation for path following

Input: $\mathscr{P}$, Ω, Ξ, η, $\mathcal{Y}^{k_o}$ $\mathcal{C}_{\text{init}}$, $k_{\max}$, $\mathscr{B}$
Output: $\hat{\mathcal{C}}$

segment $\mathscr{P}$ into η sequential points
for i = 1:1:η **do**
  $\mathscr{R}^i = \mathscr{P}^i \oplus \mathscr{B}$
end for
for i = 1:1:η **do**
  for j = 2:1:k_{end} **do**
    set $\mathcal{Y}^0 = \mathscr{R}^{j-1}$
    obtain $\tilde{\mathcal{C}}$ s.t. $\mathcal{Y}^j | \mathscr{R}^{i-1} \subseteq \mathscr{R}^i$
    if $\tilde{\mathcal{C}} \neq \emptyset$ **then**
      break
    end if
  end for
  if $\tilde{\mathcal{C}} = \emptyset$ **then**
    set $\hat{\mathcal{C}} = \emptyset$
    break
  end if
  update $\tilde{\mathcal{C}} = \mathcal{C}_{\text{init}} \bigcap \tilde{\mathcal{C}}$
end for

Obtaining a guaranteed estimated set $\tilde{C}$ is the focus of the next Chapter 4. If consecutive waypoints cannot be reached with one time step, additional constraints of the form (2.5) can be added to the problem set-up (2.6). Achieving path following is dependent not only on the controller, the system constraints but also on the path itself, the segmentation into waypoints and the safety set $\mathcal{B}$ [3].

Set-based disturbance rejection

Disturbances are undesired but not changeable influences on the system dynamics. Examples of disturbances are side wind gust to a vehicle that pushes it outside the lane or opening the door of a refrigerator and thus influencing the internal chamber temperature.

The disturbances $d(k)$ leads to a change of the states $x(k)$, which, in turn, leads to

a change of the output $y(k)$. The control task is to return the system to the old or some acceptable levels of the reference, which can be posed as follows:

Problem 5. *(Set-based disturbance rejection) For a controlled system* (2.6) *and initial output conditions* $\mathcal{Y}^{k_0}$ *and disturbance* $d(k) \in \mathcal{D}^*$*, obtain a set of controller parameterisations* $\hat{\mathcal{C}}$*, such that the closed-loop outputs are bounded by the desired output profile* $\bar{\mathcal{Y}}^k$*.*

Figure 3.5 presents an example of a profile $\bar{\mathcal{Y}}^k$ for the admissible output values $y(k)$, following a linear decay.

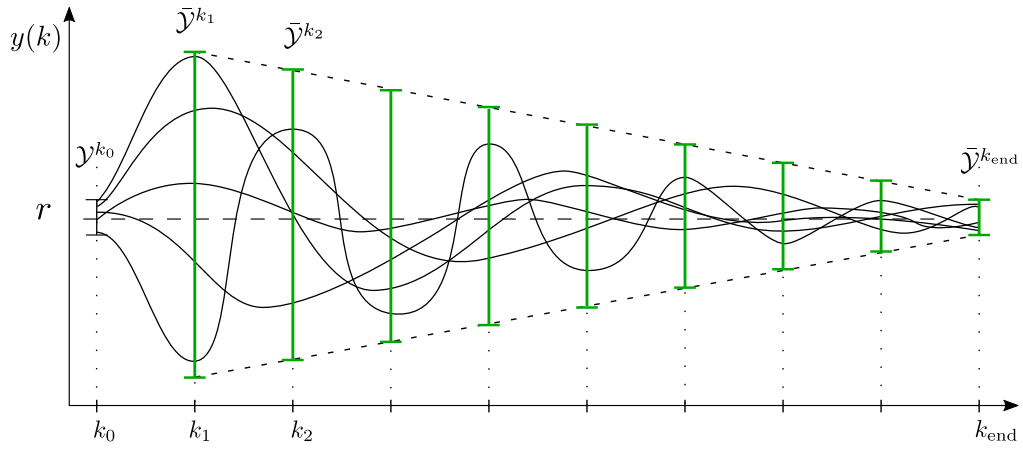


Figure 3.5: An example of a desired output behaviour for a disturbance rejection scenario. The green bars are the admissible output values for each time instance. The black lines are example output trajectories. The dashed lines illustrate the imposed linear decay on the range of the set $\mathcal{Y}^k$ that contain the admissible output signal values.

Based on the amplitude and type of the disturbance d, the target set $\bar{\mathcal{Y}}^{k_{\text{end}}}$ at the end of the control horizon k_{end} might pose an offset compared to $\mathcal{Y}^{k_0}$. Solving Problem 5 can be done straightforwardly by adding the desired output profile $\bar{\mathcal{Y}}^k$ to the problem set-up as a requirement and estimating the admissible controller parameters, i.e.

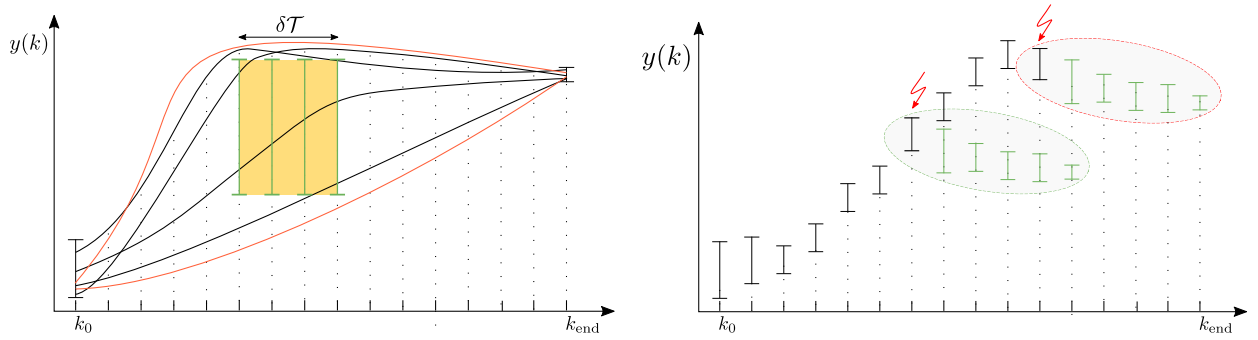
$$y(k) \in \bar{\mathcal{Y}}^k, k \in \mathcal{T} \backslash \{k_0\}.$$

The bounding profile of the desired output sets $\overline{\mathcal{Y}}^k$ can be considered as design freedom. Examples are linear decay rate or exponential decay [222]. Information that could aid the disturbance rejection is to model the disturbance. Including identification information of the disturbance is critical in aiding the disturbance rejection task, e.g. information about the model of the noise or the precise amplitude of the disturbance [115], [228]. To adjust the volumes of the sets, an online estimator of the amplitude and time of occurrence of the disturbance can also be beneficial [126]. Compared to the presented scenarios so far, we acknowledge the fact that other requirements involve logical statements and are not given with specific time instances. This describes a more

complex behaviour, including conditional dependencies and temporal uncertainty. We refer to them as qualitative requirements.

3.2 Qualitative Requirements

In many systems, like manufacturing, biological or chemical processes, the controller requirements are neither fixed in time nor having a precisely defined amplitude. They contain and are characterised often through qualitative system behaviour. Examples of such behaviour include conditional requirements like 'if Phenomenon A happens; then, it must be followed by either Phenomenon B or Phenomenon C'. One way to formulate each of these requirements is through Boolean algebra expressions. Moreover, the desired characteristics are often not fixed at a particular time but need to be valid at all times. This leads to the inability to pose a quantitative requirement at a fixed time. We include temporal operators, like *eventually* in our formulation, and to express them, we considered temporal logic. Temporal uncertainty can be due to a period during which this requirement is active, or it can be conditional and thus only occur after a particular event. Those two scenarios are illustrated in Figure 3.6.



(a) The yellow region and $\delta\mathcal{T}$ symbolise the temporal uncertainty of the constraint depicted by the black bars.

(b) The red lightnings illustrate the enforcement of a condition, depicted by the green bars in the grey bubble, at different stages.

Figure 3.6: A depiction of requirements with temporal (on the left) or conditional (on the right) uncertainty.

In addition to the discussed qualitative requirements, we consider discontinuities. They allow to capture behaviours like saturation, dead zone, rate limit. We present their formulation in a suitable mixed-integer formulation in agreement with the problem set-up.

3.2.1 Basic Qualitative Requirements

In many systems, the performance or the safety of the system is expressed by basic logical expressions that can be formulated using Boolean algebra operators. The three basic operators are: AND - conjunction, depicted by $\wedge$; OR - disjunction, depicted

by $\vee$; and NOT - negation, depicted by $\neg$. In Boolean algebra, the variables and the expressions can be either TRUE, denoted by the symbol $\top$ and have the numerical value of 1, or FALSE, denoted by the symbol $\bot$ and having the value of 0. Moreover, 0 and 1 are the only two possible values for a variable or an expression in Boolean algebra. For a succinct presentation, we present the three basic Boolean operators by the following truth table with the help of the two binary variables $z_1, z_2 \in \{0, 1\}$, cf. Table 3.1.

z_1	z_2	$\neg z_1$	$z_1 \wedge z_2$	$z_1 \vee z_2$
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Table 3.1: Truth table for the operators: NOT($\neg$), AND($\wedge$), and OR($\vee$) through enumeration of the binary variables z_1 and z_2.

Expressions based on these operators are commonly used in control, e.g. 'switch to an automatic mode once a specified temperature is reached, AND the pressure alarm is NOT active'. To use logical constraints in the problem formulation (2.6), we need to formulate them in polynomial form (2.5). Expressing the logical operators through polynomial expressions with binary variables can be done through the standard conversion method [37]. To illustrate the re-formulation of a logical requirement, we take a Boolean expression $l_b(z)$ and pose it as the mixed-integer polynomial $l(z)$, e.g.

$$\begin{aligned} l_b(z_1, z_2) = z_1 \wedge z_2 &\leftrightarrows l(z_1, z_2) = z_1 z_2, \\ l_b(z_1, z_2) = z_1 \vee z_2 &\leftrightarrows l(z_1, z_2) = z_1 + z_2 - z_1 z_2, \\ l_b(z_1) = \neg z_1 &\leftrightarrows l(z_1) = 1 - z_1, \end{aligned} \tag{3.3}$$

where the symbol $\leftrightarrows$ should be read as 'is equivalent to the expression'. Using (3.3) allows to formulate Boolean logic expressions into the requirement's form (2.5). We are interested in enforcing a general controller requirement (2.5) that is not only of logical variables but an expression of the variables in the problem formulation. The validity of a requirement constraint can be expressed as a binary result. Therefore, we can bind the value of a binary variable z_i to the validity of a constraint $l(a) \geq 0, a \in \mathcal{A}$. Here a is a placeholder variable and $\mathcal{A}$ is the set of admissible values for a. For brevity, we refer to the validity of a constraint $l_i(a) \geq 0$ as the statement L_i. For a detailed tutorial on the topic of model building of systems with mixed integers, we refer the reader to [24], [49], [232]. We point out the following basic relations that are commonly

used to simplify or formulate logical expressions:

$$\begin{aligned}
&\neg\top = \bot,\\
&\neg\neg z_1 = z_1,\\
&\neg(z_1 \wedge z_2) = \neg z_1 \vee \neg z_2,\\
&\neg(z_1 \vee z_2) = \neg z_1 \wedge \neg z_1.
\end{aligned}$$

Besides the basic operators, we also use the equivalence operator - if and only if - to connect the validity of requirement L to a binary variable z. In addition to the presented three basic logical operators, we discuss three more commonly used logical operators: IFF is the equivalence operator, denoted by $\Leftrightarrow$; IF...THEN is the material implication operator, denoted by $\Rightarrow$; and XOR is the exclusive disjunction operator, denoted by $\not\equiv$. The truth tables for these functions are given in Table 3.2.

z_1	z_2	$z_1 \Leftrightarrow z_2$	$z_1 \Rightarrow z_2$	$z_1 \not\equiv z_2$
0	0	1	1	0
0	1	0	1	1
1	0	0	0	1
1	1	1	1	0

Table 3.2: The truth table for the operators: IFF($\Leftrightarrow$), IF($\Rightarrow$), and XOR($\not\equiv$) through an enumeration of the binary variables z_1 and z_2.

In principle, one logical operator can be expressed through others. For example, exclusive disjunction can be formulated through negation, disjunction and conjunction. Nevertheless, working only with the basic elements leads to longer expressions that can cost more computation time. Also, longer expressions make the discussion more challenging to follow and thus more susceptible to errors. Therefore, the decision, whether the formulation should be limited to the basic operators, needs to be evaluated for each specific case. For completeness, we provide the following expressions that substitute the later three operators through the basic operators:

$$\begin{aligned}
z_1 \Rightarrow z_2 &\leftrightarrows \neg z_1 \vee z_2,\\
z_1 \Leftrightarrow z_2 &\leftrightarrows (\neg z_1 \vee z_2) \wedge (\neg z_2 \vee z_1),\\
z_1 \not\equiv z_2 &\leftrightarrows z_1 \wedge \neg z_2 \vee \neg z_1 \vee z_2.
\end{aligned}$$

In manufacturing, and in particular, in Industry 4.0 related examples, these operators are used to describe the decision-making of changing production routes in runtime, e.g. 'IF an item is heavier than A kilograms, THEN an item can be processed EITHER by Machine 1 OR by Machine 3' [2], [102].

In this work, we consider the control of hybrid dynamical systems, which means that the validity of a requirement, or generally speaking a constraint, is time-dependent, e.g. it is allowed that a threshold can be crossed only at certain times but not in others. As a consequence, each requirement l_i needs to be evaluated, whether it is

valid at each instance k. In turn, the corresponding binary variable z_i needs to be also time-variant. Considering two requirements l_1 and l_2, their corresponding binary variables z_1 and z_2 and their validity horizon $\mathcal{T}^* \subset \mathcal{T}$, we can derive the following expressions for the presented logical operations:

$$\begin{aligned}
L_1(k) \wedge L_2(k) = \top, k \in \mathcal{T}^* &\leftrightarrows z_1(k) + z_2(k) = 2, k \in \mathcal{T}^*, \\
L_1(k) \vee L_2(k) = \top, k \in \mathcal{T}^* &\leftrightarrows z_1(k) + z_2(k) \geq 1, k \in \mathcal{T}^*, \\
L_1(k) \Leftrightarrow L_2(k) = \top, k \in \mathcal{T}^* &\leftrightarrows z_1(k) - z_2(k) = 0, k \in \mathcal{T}^*, \\
L_1(k) \Rightarrow L_2(k) = \top, k \in \mathcal{T}^* &\leftrightarrows z_1(k) - z_2(k) \leq 0, k \in \mathcal{T}^*, \\
L_1(k) \not\equiv L_2(k) = \top, k \in \mathcal{T}^* &\leftrightarrows z_1(k) + z_2(k) = 1, k \in \mathcal{T}^*, \\
\neg L_1(k) = \top, k \in \mathcal{T}^* &\leftrightarrows z_1(k) = 0, k \in \mathcal{T}^*,
\end{aligned}$$

where $\neg L_1(k)$ means that we desire the requirement L_1 at time k to be violated. Negating a requirement is a critical tool that we use for the estimation of the controller parameters, see Section 4.4.

So far, we considered logical expressions consisting of either binary variables or requirements statements. Now we take a look at a mixed case, in which we combine the validity of a requirement and a binary variable. To connect a statement L with the state of a binary variable z, we introduce M and m. These two variables are correspondingly the upper and the lower boundary of a function $l(a)$, i.e.

$$\begin{aligned}
M &:= \max_{a \in \mathcal{A}} l(a), \\
m &:= \min_{a \in \mathcal{A}} l(a).
\end{aligned}$$

Additionally, we introduce a small positive constant $\epsilon \approx 0, \epsilon \in \mathbb{R}^+$, which is commonly chosen to be equal to the machine precision of the computation device [24]. Using these three additional variables, formulating the expression $l(a) \leq 0$ if and only if $z = 1$ can be done as:

$$l(a) \leq 0 \Leftrightarrow z = 1 \text{ is valid iff } \begin{aligned} l(a) &\leq M(1 - z) \\ l(a) &\geq \epsilon + (m - \epsilon)z \end{aligned}$$

Analogously, we can express disjunction, conjunction and implication between an inequality and a binary variable

$$\begin{aligned}
l(a) \leq 0 \wedge z = 1 &\quad \text{is valid iff} \quad l(a) - z \leq -1 + m(1 - z) \\
l(a) \leq 0 \vee z = 1 &\quad \text{is valid iff} \quad l(a) \leq Mz \\
l(a) \leq 0 \Rightarrow z = 1 &\quad \text{is valid iff} \quad l(a) \geq \epsilon + (m - \epsilon)z.
\end{aligned}$$

To enforce a negation on a constraint, we can easily just inverse the sign and thus avoid the introduction of an extra binary variable. So far, we looked at modelling inequalities, but the framework is valid also for equalities, as they are also natively supported in the problem formulation (2.6). Besides the obvious way of expressing an equality - through two inequalities intersecting at the boundary, we refer the reader

to [49] and [232] for more details on modelling through mixed-integer expressions.

The presented logical operators were discussed for two variables. Combining more of them, we can formulate more complex expressions. In Boolean algebra, each binary variable can be considered as a substitution for any Boolean formula. Nevertheless, we point out a simple way of formulating the enforcement of disjunction and conjunction for more than two variables through mixed-integer expressions. In the case of enforcing multiple controller requirements that are in conjunction with each other, adding and expressing them through binary variables could be skipped. They can be directly included in the set-up and thus skipping the additional binary variables and not increasing the problem size. Such formulation reduces the problem size and saves computation time. Nevertheless, if more than two variables are combined with conjunction inside a more complex logical expression, we can model the multiple conjunction as:

$$z_1 \wedge z_2 \wedge ... \wedge z_n = \top \leftrightarrows \sum_{i=1}^{n} z_i = n.$$

For a conjunction of multiple variables to be true, we can use the following expression:

$$z_1 \vee z_2 \vee ... \vee z_n = \top \leftrightarrows \sum_{i=1}^{n} z_i \geq 1,$$

and thus the inequality is satisfied. The corresponding statement is true as long as one of the binary variables is equal to 1.

In summary, the presented mixed-integer modelling techniques enable the consideration of requirements that can be formulated as Boolean algebra expressions. Therefore, we can address conditional system requirements directly in the controller tuning process.

3.2.2 Temporal Operators

In addition to the Boolean logic operators, we are interested in a method of describing requirements with temporal uncertainty. In such a case, the validity period that a particular requirement needs to be fulfilled is not fixed to specific time instances. In particular, examples for such behaviours are 'next after input A reaches a threshold B, then output C raises and eventually returns to the initial range D' and 'until flow A is smaller than flow B, valve C has to be henceforth opened'. From these two examples, four temporal operators can be identified: *next*, *eventually*, *until*, and *henceforth*. Such formulations are met in the verbal description of the operation of systems and contain system information that needs to be considered during the controller tuning. These operators are beyond the expressing capabilities of Boolean algebra and are not commonly considered in classical controller tuning methods but are conveying important information. In this work, we are interested in including the qualitative behaviour such operators describe.

Remark 1. *We neither perform formal verification nor utilise the analytical capabilities that the different temporal logics posses and their proving capabilities. In particular, some of these logics are correct only in the case of infinite time operation [171], which is not the case for the systems we consider, where others require fixed space segmentation, which we also do not consider directly [21]. Nevertheless, as stated in Problem 2, we are interested and thus provide a framework to include the qualitative behaviour in a mathematically solid framework, such that the obtained control results are guaranteed, see Chapter 4.*

In particular, we have a look at the aforementioned four temporal operators *next*, *eventually*, *henceforth* and *until*. To explain the behaviour of these operators, we introduce the term literal, which in propositional calculus is used to refer to a propositional variable or the negated form of the variable [45]. The operator *next*, denoted by $\bigcirc$, has a value 1 when the formula $\bigcirc a$ is valid at the next time instance. Therefore, this operator is also referred to as *nexttime*. Intuitively, *henceforth*, denoted by $\Box$, is equal to 1 if all upcoming time instances for the formula $\Box a$ are true. A more common name for this operator is *always*, but we find the latter misleading from a linguistic point of view. Two variables are needed when using *until*, denoted by $a \mathscr{U} b$, expresses that a holds true until b becomes true. The operator *eventually*, denoted by $\diamondsuit$, conveys that a literal A must be TRUE at some unspecified upcoming instance. An alternative naming for this operator is *sometimes*. The moment in which A has to be TRUE is unspecified, but it is required to happen in the future at least once. The effects these operators have over a literal are presented in Table 3.3.

$\mathcal{T}$	k_0	k_1	k_2	k_3	k_4	k_5	k_6
x	0	1	0	0	1	1	1
y	5.3	6.1	6.9	9.2	1.2	3.1	1.6
$z_1 \Leftrightarrow y < 6$	1	0	0	0	1	1	1
$z_2 \Leftrightarrow x = 0$	1	0	1	1	0	0	0
$\bigcirc z_1$	0	0	0	1	1	1	-
$\Box z_1$	0	0	0	0	1	1	1
$\diamondsuit z_1$	1	1	1	1	0	0	0
$z_2 \mathscr{U} z_1$	0	0	1	1	0	0	0

Table 3.3: Illustrative evaluation of several temporal expressions.

where $z_1, z_2, x \in \{0, 1\}$, $y \in \mathbb{R}$.

As seen from the definitions, these four operators are reflecting the meaning of the words that they have in human language. In this way, a verbal description can be expressed through a temporal logic formula, which in turn, we postulate as equalities and inequalities. We offer the following formulations:

$$
\begin{array}{lll}
\bigcirc z_1(k) & \rightarrow & z_1(k^+) = 1, \\
\Box z_1(k) & \rightarrow & z_1(k) = 1, k \in \{l, ..., k_{\text{end}}\}, \\
\Diamond z_1(k) & \rightarrow & \Sigma z_1(k) \geq 1, k \in \{l, ..., k_{\text{end}}\}, \\
z_1(k) \mathscr{U} z_2(k) & \rightarrow & \begin{cases} z_2(k) & \text{when } k = k_{\text{end}}, \\ z_2(l) \wedge (z_1(k) \vee (z_1(k^+) \mathscr{U} z_2(k^+))) & \text{otherwise.} \end{cases}
\end{array}
$$

where l is the current instance at the control horizon, and the formulation of *until* is based on the presented one in [30].

Analogously to the Boolean operators, the temporal operators contain a form of redundancy and can express one operator through others, e.g.:

$$
\begin{aligned}
&\Box a \equiv \neg \Diamond \neg a, \\
&\Box a \equiv a \mathscr{W} \bot, \\
&\Diamond a \equiv \neg \Box \neg a, \\
&a \mathscr{U} b \equiv (a \mathscr{W} b) \wedge \Diamond b.
\end{aligned}
$$

Note that intentionally avoiding some of the available operators can lead to unnecessary bloating of the expressions and thus of the problem size. To alleviate the simplification of temporal formulae and also for completeness, we would like to point several common equivalence relations of Linear Temporal Logic formulae:

$$
\begin{aligned}
&\neg \Diamond a \equiv \Box \neg a, \\
&\neg \Box a \equiv \Diamond \neg a, \\
&\neg \bigcirc a \equiv \bigcirc \neg a, \\
&\Diamond \Diamond a \equiv \Diamond a, \\
&\Diamond (a \vee b) \equiv \Diamond a \vee \Diamond b, \\
&\Box (a \wedge b) \equiv \Box a \wedge \Box b.
\end{aligned}
\tag{3.4}
$$

As seen from the examples, combining temporal logic operators is allowed. Nevertheless, due to the last two equivalence relations in (3.4), the following cases should be kept in mind:

$$
\begin{aligned}
&\Diamond (a \wedge b) \not\equiv \Diamond a \wedge \Diamond b \\
&\Box (a \vee b) \not\equiv \Box a \vee \Box b.
\end{aligned}
$$

To illustrate the construction of a temporal logic expression, let us consider the following example of controlling a drying chamber:

Example 4. *If the door is closed, keep the temperature higher than 40 degrees and less than 45 until the moisture in the air is higher than 20 % and henceforth keep the door open.*

We denote with z_1 the door being closed, with z_2 the temperature being more than 40 degrees, with z_3 the temperature being below 45 degrees, and with z_4 the moisture

being more than 20%. Therefore, we can formulate the temporal formula that describes Example 4 like

$$z_1 \Rightarrow (z_2 \wedge z_3) \mathscr{U} z_4 \bigcirc \Box \neg z_1$$

Two or more temporal operators can be combined, and such a combination is referred to as a modality. The order in which they are formulated is also defining their meaning. Some of the combinations can be irrelevant for the set-up considered here, e.g. $\Box\Diamond$ will henceforth eventually require fulfilment of the condition, which is not meaningful since we work with systems that operate in a finite-time control horizon. In contrast, the inverse combination $\Diamond\Box$ is relevant for the control context as it expresses system stability, i.e. reach the steady-state and stay there. The modality $\Diamond\Box$ can be expressed through Boolean operators:

$$\Diamond\Box z = \bigvee_{i=1}^{n_k} \bigwedge_{j=i}^{n_k} z(j)$$

In addition to the presented operators, there are two other common operators, namely *weak until* and *strong release* [178]. They do not bring significant new expressive capabilities to the control requirements, and thus we do not discuss them. Some temporal logics consider an extension through which the validity of the constraint is limited to a validity period, i.e. it has a beginning and an end time of the requirement. As we deal with finite-time horizon control, and we use the temporal operators as inspiration, this extension is inherently included. Still, when constructing a modality, the validity periods have to be the same.

Remark 2. *Temporal logics have also definitions for executions that relate to the past. The operators* next, henceforth, eventually, until, unless *have their corresponding past versions:* previously, has-always-been, once, since, back-to. *For the detailed semantics and proving axioms, we refer to [134]. There are also other operators* proceed, ensure, leads-to, entails. *The reason we do not discuss them here is they have a time-mirrored execution and do not bring new modelling capabilities for our set-up.*

The different logic operators and the ability to pose them as mixed-integer equalities and inequalities enables the framework to formulate requirements that are conditional and have temporal uncertainty. The introduction of binary variables increases the complexity of the problem. Therefore, shorter and simpler formulations are recommended. Another step to reduce the computational load is to check for redundant constraints and to eliminate them from the set-up as they might cause additional computational overhead.

3.2.3 Qualitative Control Scenarios

Set-based deadbeat control

Deadbeat control is a commonly desired control system behaviour that combines a fast step response during a step change of the reference with an overshoot limit. This requirement is also known as posicast control [128]. Often doing so, transitory oscillations can be observed in linear systems that are second or higher-order and also in nonlinear systems. Although for linear systems recommended strategies exist, this is still an open question for our system class [151], [229]. Regarding the damping qualitative behaviour, a transient response can be classified as being underdamped, overdamped, or critically damped, see Figure 3.7. An underdamped system reaches the reference r_{new} much quicker, compared to a critically damped system $k_{\text{ud}} < k_{\text{cd}}$. This fast response often leads to a substantial overshoot and significant oscillations around the reference. In contrast, the overdamped does not exhibit any overshoot, but it takes much longer to reach the reference and quite often achieves a poor steady-state error margin. There is a point beyond which the system is no longer underdamped and starts exhibiting oscillations, and the process is referred to as critically damped. A transient response example for each of the three behaviours is presented in Figure 3.7.

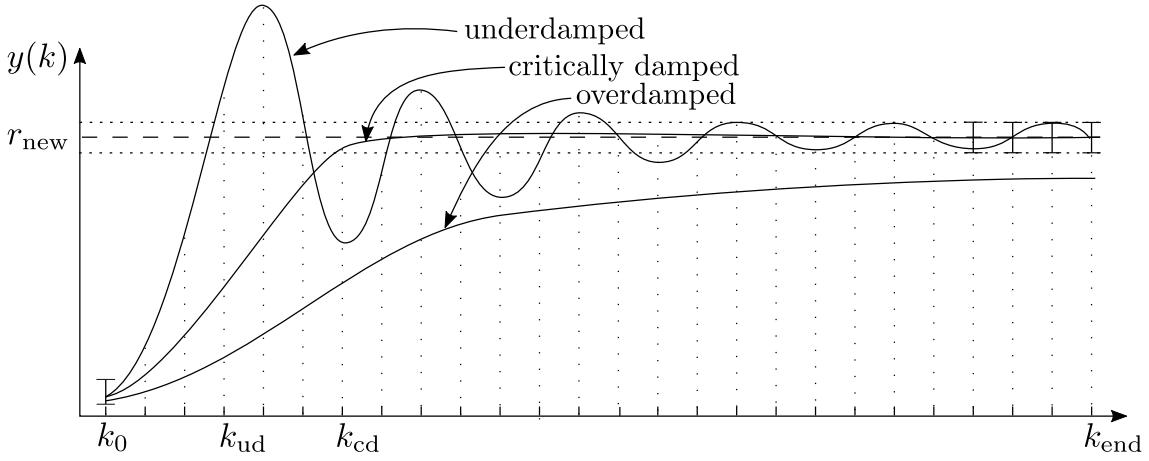


Figure 3.7: Qualitative comparison between an underdamped, overdamped and critically damped transient behaviour. Although the underdamped achieves the precision range much faster than the critically damped, it experiences significant oscillations, and the overdamped does not manage to achieve the precision range during the considered horizon.

Combining the fast rise time from the underdamped and the small or none overshoot from the underdamped characterise the so-called critically damped behaviour [59]. In the control engineering context, this behaviour is the main characteristic of deadbeat control. Following the design requirements for deadbeat control in [59], we outline them for systems with uncertainties:

- steady-state error smaller than ϵ,
- overshoot less than $\overline{\delta}\%$,

- undershoot less than $\underline{\delta}\%$,
- minimum settling time k_{settle} for all possible trajectories.

In some formulations of deadbeat control, there is also a specified minimum overshoot $\delta_{\min}$, such that the system is forced to improve its rise time [59]. As we deal with uncertain systems, it is no longer possible or desirable to require that the system achieves a singular value and stays at it. The minimum settling time can be achieved through an iterative procedure where the settling time is shifted to an earlier time instance as long as feasible controller parameters are found. The iterative solution approach leads to the inability to specify each of those characteristics with fixed times. Therefore, they have to be specified qualitatively in a conditional manner.

Algorithm 3 Set-based tuning for deadbeat control

Input: Ω, Ξ, $\underline{\delta}$, $\overline{\delta}$, δ, ϵ, $\mathcal{T}$
Output: $\hat{\mathcal{C}}$

---

set $n =$ size of $\mathcal{T}$
set $i = 1$
set $\hat{\mathcal{C}} = \emptyset$
while $i < n$ AND $\hat{\mathcal{C}} = \emptyset$ **do**
  set $k_{\text{settle}} = i$
  obtain $\hat{\mathcal{C}}$ s.t. the performance requirements: $\underline{\delta}$, $\overline{\delta}$, δ, ϵ, k_{settle}
  set $i = i + 1$
end while

Usually, the price to pay for a fast transient response is a higher amplitude and sometimes significant oscillations of the input signal, which can be addressed through additional requirements L_i. Two examples of this scenario are the control of wind rotors and X-Y plotters [59].

Set-based bumpless control

In this scenario, a system Ω is switched from one control strategy to another [51]. When the switching occurs at k_{switch}, a strong fluctuation, the so-called 'bump', in the output signal y is commonly observed, cf. Figure 3.8. Therefore, to avoid damaging the system, a smooth transition between the two control strategies is desired, e.g. the control of heat exchangers [118]. Classically, the switching between the two control regimes is from a manual mode to an automatic one. Another example can be the change from one control strategy that provides good reference change to a second one that has good disturbance rejection properties. Therefore, to ensure the bumpless control transition, the process requirement is a smooth switching transfer. Numerous examples for such tuning rules of PID controller can be found in [161], [240].

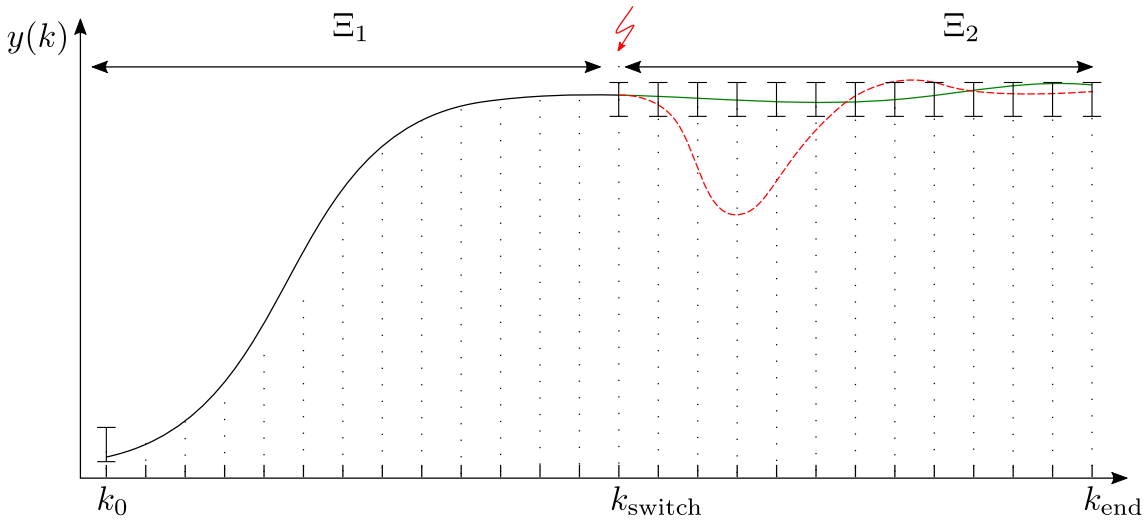


Figure 3.8: A depiction of a desired (in green) vs. undesired (in red) bumpless control. At k_{switch} controller Ξ_1 stops being active and Ξ_2 takes over the control of the system.

One of the reasons for this behaviour to occur is that the dynamical controller needs time to adjust its states such that the output is maintained. To address it, strategies that are based on mirroring the effect of the system exist to ensure a bidirectional transfer [59]. However, this becomes challenging for constrained hybrid nonlinear multi-input multi-output systems, and a resulting swinging output behaviour can lead to system instability, performance degradation, or permanent damage to the system. Therefore, depending on the system, managing the transition can be of critical importance. To achieve it, we can specify output constraints $\mathcal{Y}^k$ that are enforced after the switching. We are interested in the general case of switching between any two controllers Ξ_1 and Ξ_2 that can be posed as in (2.3).

Problem 6. *(Set-based bumpless control) For system* (2.6)*, achieve bumpless control by finding a set of controller parameters* $\mathcal{C}_2$*, such that when the switching between the two controllers* Ξ_1 *and* Ξ_2 *at* k_{switch} *occurs, the outputs* $y(k)$ *are bounded by* $\mathcal{Y}^k$*,* $k \in \{k_{switch+1}, k_{switch+2}, .., k_{end}\}$.

Problem 6 refers to two possible scenarios, depending on whether k_{switch} is fixed in time or conditional. When the switching time is known a priori, it describes a *switched* system behaviour. In this case, the first controller Ξ_1 is active during the first part of the control horizon, i.e. $\{k_0, k_1, ..., k_\alpha\}$ and the second controller Ξ_2 is active until the end, i.e. $\{k_{\alpha+1}, k_{\alpha+2}, .., k_{\text{end}}\}$. To tackle this scenario, the following controller structure and constraints are added to the system description (2.6):

$$
\begin{aligned}
&\Xi(k) = \alpha_1(k)\Xi_1(k) + \alpha_2(k)\Xi_2(k), k \in \mathcal{T},\\
&\alpha_1(k) = 1, \alpha_2(k) = 0, k \in \{k_0, k_1, .., k_{\text{switch}}\},\\
&\alpha_1(k) = 0, \alpha_2(k) = 1, k \in \{k_{\text{switch}+1}, k_{\text{switch}+2}, .., k_{\text{end}}\},\\
&y(k) \in \mathcal{Y}^k, k \in \{k_{\text{switch}+1}, k_{\text{switch}+2}, .., k_{\text{end}}\}.
\end{aligned}
$$

In contrast to working with a defined switching time k_{switch}, the case with an unknown switching time describes a *switching* type of system [120]. Let us denote with $z_1(k)$ and $z_2(k)$ the Boolean variables that are the bounding requirement for the output to switch, and $z_3(k)$ is the condition that limits the output inside the admissible region. For illustrative purposes, we consider the switching condition is the output reaching ϵ close to the reference r. This can be formulated through the following constraints:

$$
\begin{aligned}
&z_1(k) \Leftrightarrow y(k) \leq r + \epsilon, \\
&z_2(k) \Leftrightarrow y(k) \geq r + \epsilon, \\
&z_3(k) \Leftrightarrow y(k) \in \mathcal{Y}^k, \\
&(z_1(k) \wedge z_2(k)) \rightarrow \Box z_3(k).
\end{aligned}
$$

A reasonable addition to the switching condition is a requirement that the outputs spend several consecutive time instances close to the reference, and only then the switch occurs. This switching condition and any additional ones can be added before the material implication, in conjunction with $z_1(k)$ and $z_2(k)$. Although the bumpless transfer can be addressed through techniques from anti-windup design [84], here we presented a generic qualitative requirement formulation that tackles it directly and allows more capabilities. In the following section, we take a look at several discontinuities and, in particular, how to formulate and specify signal saturation, which is the culprit for the windup challenge that arises in control.

3.2.4 Discontinuity Phenomena

We take a look at a particular class of nonlinear dynamical phenomena that is often referred to as discontinuities [206]. These phenomena describe behaviours that are not fixed to a specific time and are observed as long as the conditions describing them occur. Therefore, we consider them as qualitative behaviours. Some of them are also dependent on values from previous instances and hence have a dynamical component. Moreover, the behaviour of each of these phenomena can be separated from the rest of the system, thus allowing us for a targeted discussion and custom modelling.

We take a look at the following discontinuities: saturation, dead zone, rate limiter, and relays with and without hysteresis. They can be formulated as a switching system with a mode per sub-behaviour or as system requirements. To be consistent with the approach here, we model these phenomena via mixed-integer polynomial expressions.

To illustrate the influence of the discontinuity elements, we look at their behaviour and evaluate them as input-output blocks, with $u(k)$ as the input signal and $y(k)$ as the output signal. This systematic view allows describing the nonlinear behaviour of a particular discontinuity separately from the rest of the dynamics. For each of them, we present a graphical input-output relation, the defining mathematical formulation, and a qualitative formulation through which they can be added to the problem formulation. To express the discontinuities, we introduce auxiliary binary variables $z_i(k) \in \{0, 1\}$

that are time-dependent, as the effect of the discontinuities is dependent on the values of time-variant signals.

Saturation expresses a behaviour in which inside a defined range, the input signal is passed through without modifying it, and outside this range, the signal is saturated. Depending on whether $u(k)$ is beyond $\underline{u} \in \mathbb{R}$ or $\overline{u} \in \mathbb{R}$, it is correspondingly limited to $\underline{y} \in \mathbb{R}$ or $\overline{y} \in \mathbb{R}$, where $\underline{u} < \overline{u}$ and $\underline{y} < \overline{y}$, see Figure 3.9 and (3.5). Practical examples are any system that operates from 0% to 100%, e.g. mechanical, biological, computer systems, etc. In particular, in any engineering system, some components exhibit such behaviour, e.g. a valve, a voltage transformer, heating/cooling capabilities of air-conditioning systems, practically speaking any actuator.

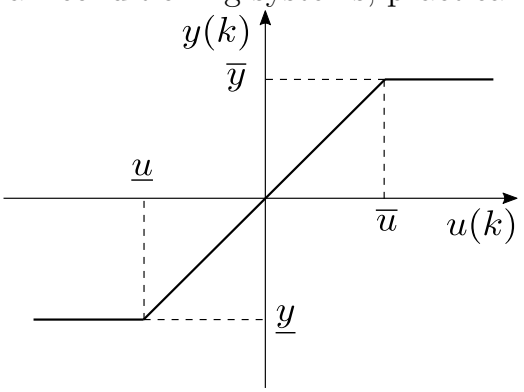


Figure 3.9: Signal transformation of a saturation discontinuity.

$$y(k) = \begin{cases} \overline{y}, & u(k) \geq \overline{u}, \\ \underline{y}, & u(k) \leq \underline{u}, \\ u(k), & otherwise. \end{cases} \tag{3.5}$$

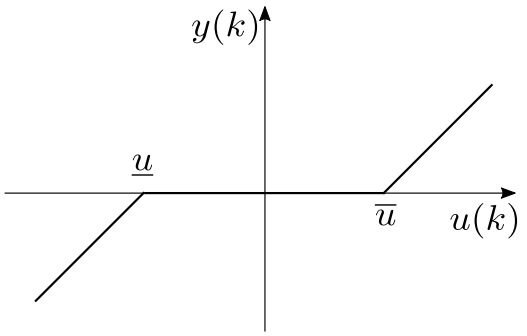
To express saturation qualitatively, we use two additional binary variables $z_1(k)$ and $z_2(k)$, where the first is equal to 1 when $u(k) \geq \overline{u}$ and the latter is equal to 1 if $u(k) \leq \underline{u}$. Between the two limits $\underline{u}$ and $\overline{u}$ the system output $y(k)$ is equal to $u(k)$, i.e.

$$\begin{aligned} &z_1(k) = 1 \iff u(k) \geq \overline{u}, \\ &z_2(k) = 1 \iff u(k) \leq \underline{u}, \\ &y(k) = z_1(k)\overline{u} + z_2(k)\underline{u} + (1 - z_1(k))(1 - z_2(k))u(k). \end{aligned} \tag{3.6}$$

In principle, the system can behave also non-linearly inside the saturation region $[\underline{u}, \overline{u}]$. Still, we consider the influence of each discontinuity separately, and additional phenomena can be added and analysed individually. In summary, the design parameters for a saturation in (3.6) are $\underline{u}$ and $\overline{u}$.

Dead zone phenomenon describes insensitivity to small signal amplitudes, see Figure 3.10. As long as the input signal $u(k)$ is within the dead zone region $(\underline{u}, \overline{u})$, $\overline{u} \in \mathbb{R}_{>0}$, $\underline{u} \in \mathbb{R}_{<0}$, the output $y(k)$ remains unchanged, typically equal to zero, see Figure 3.10 and (3.7). Outside this non-sensitive range, the system output behaves linearly. An example of this effect is the sinusoidal control input signal for controlling the position of a hydraulic piston [164]. In this case, a small amplitude of the applied voltage does not cause displacement of the piston, but only after certain amplitude

thresholds are reached.



$$y(k) = \begin{cases} u(k) - \overline{u}, & u(k) > \overline{u}, \\ u(k) + \underline{u}, & u(k) < \underline{u}, \\ 0, & otherwise. \end{cases} \tag{3.7}$$

Figure 3.10: Signal transformation of a dead zone discontinuity.

In this case, we use two additional binary variables $z_1(k)$ and $z_2(k)$ to signify whether the input $u(k)$ is outside the dead zone. The insensitive region between them is modelled by a conjunction of the negation of both auxiliary variables.

$$\begin{aligned} &z_1(k) = 1 \iff u(k) > \overline{u}, \\ &z_2(k) = 1 \iff u(k) < \underline{u}, \\ &y(k) = z_1(k)(u(k) - \overline{u}) + z_2(k)(u(k) - \underline{u}). \end{aligned} \tag{3.8}$$

In summary, the design parameters for a dead zone in (3.8) are $\overline{u}$ and $\underline{u}$.

Rate limiter describes how much a signal changes between two consecutive time instances. Practically speaking, this discontinuity relates to limiting the signal's first derivative in time. Therefore, we are interested not only in the current value of the input signal $u(k)$ but also in the output value at the previous time instance $y(k^-)$. Examples where this limit is relevant are the acceleration of a vehicle [91], the rate of change of altitude of an aeroplane [201].

$$y(k) = \begin{cases} y(k^-) + (k - k^-)\overline{y}^*, & \frac{u(k)-y(k^-)}{k-k^-} > \overline{y}^*, \\ y(k^-) - (k - k^-)\underline{y}^*, & \frac{u(k)-y(k^-)}{k-k^-} < \underline{y}^*, \\ u(k), & otherwise. \end{cases} \tag{3.9}$$

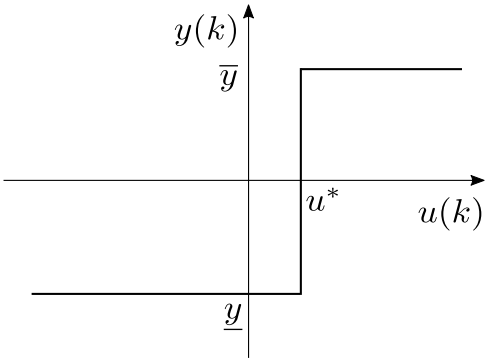
In (3.9) $\overline{y}^* \in \mathbb{R}$ and $\underline{y}^* \in \mathbb{R}$ denote the maximum and minimum rate change, i.e. the change in the signal from the previous step k^- to the current one k, moreover, $\overline{y}^* > \underline{y}^*$. This phenomenon also has three regions in which it changes its operation. Thus, we use two additional binary variables $z_1(k)$ and $z_2(k)$:

$$\begin{aligned} &z_1(k) = 1 \iff (u(k) - y(k^-))/(k - k^-) > \overline{y}^*, \\ &z_2(k) = 1 \iff (u(k) - y(k^-))/(k - k^-) < \underline{y}^*, \\ &y(k) = z_1(k)(y(k^-) + (k - k^-)\overline{y}^*) + z_2(k)(y(k^-) - (k - k^-)\underline{y}^*) \\ &\qquad\qquad + (1 - z_1(k))(1 - z_2(k))u(k). \end{aligned} \tag{3.10}$$

In some systems, depending on the modelling approach, the rate change is already

present as a state. In such systems, it is preferred to impose the constraint directly on the state and thus preserve the number of variables. In summary, the design parameters for the rate limiter in (3.10) are $\overline{y}^*$ and $\underline{y}^*$.

Ideal infinite gain relay without hysteresis is a behaviour in which a system switches between two levels: a high level $\overline{y} \in \mathbb{R}$ and a low level $\underline{y} \in \mathbb{R}$, i.e. $\overline{y} > \underline{y}$. The switching happens at a threshold level of $u^* \in \mathbb{R}$, see Figure 3.11 and (3.11).



$$y(k) = \begin{cases} \overline{y}, & u \geq u^*, \\ \underline{y}, & otherwise. \end{cases} \tag{3.11}$$

Figure 3.11: Signal transformation of an ideal relay without hysteresis.

It is modelled with one extra binary variable $z_1(k)$:

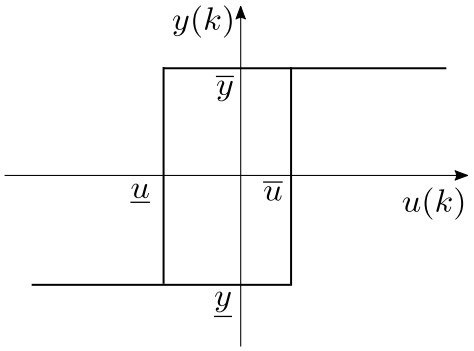
$$\begin{aligned} &z_1 = 1 \iff u \geq u^*, \\ &y(k) = z_1(k)\overline{y} + (1 - z_1(k))\underline{y}. \end{aligned} \tag{3.12}$$

Structurally close to the infinite gain relay are Coulomb and viscous friction, e.g. see [104]. Comparing the behaviour of the ideal relay and the Coulomb friction, three quantitative differences appear. First, the switching behaviour in the relay is offset by u^*. Second, after switching, the value of the friction profile increases in each direction, where the relay keeps the output value constant. Third, if the input value of the friction is equal to 0, then the output is also equal to 0, where the relay is an either-or element. Which leads to the following qualitative constraints:

$$\begin{aligned} &z_1 = 1 \iff u > 0, \\ &z_2 = 1 \iff u < 0, \\ &y(k) = z_1(k_1 u(k) + \overline{y}) + z_2(k_2 u(k) - \underline{y}). \end{aligned} \tag{3.13}$$

In summary, the design parameters for Coulomb friction in (3.13) are $\underline{y}$ and $\overline{y}$.

Relay with hysteresis is a common element in inexpensive control solutions with low precision. This discontinuity is present as a control scheme in the thermostatic control of ovens, irons, electric kettles. The width of the hysteresis zone is the tuning parameter that relates to the quality of the control process. Thus, decreasing the width leads to a smaller output variation but increases the switching frequency, which in turn means faster wear of the switching hardware.



$$y(k) = \begin{cases} \overline{y}, & (u > \overline{u}) \vee ((\underline{u} < u < \overline{u}) \wedge (y(k^-) = \overline{y})), \\ \underline{y}, & (u < \underline{u}) \vee ((\underline{u} < u < \overline{u}) \wedge (y(k^-) = \underline{y})). \end{cases} \tag{3.14}$$

Figure 3.12: Signal transformation of a relay with hysteresis.

In this case, the output $y(k)$ switches between $\underline{y} \in \mathbb{R}$ and $\overline{y} \in \mathbb{R}$, where $\overline{y} > \underline{y}$. The switching occurs at $\underline{u} \in \mathbb{R}$ and $\overline{u} \in \mathbb{R}$, where $\underline{u} < \overline{u}$, see Figure 3.12 and (3.14). To model a relay with hysteresis, we use three auxiliary variables $z_1(k)$, $z_2(k)$, and $z_3(k)$. The first two variables $z_1(k)$ and $z_2(k)$ are used to determine in which of the three regions the input signal $u(k)$ is, i.e. below $\underline{u}$, above $\overline{u}$, or between them. The third binary variable $z_3(k)$ corresponds to whether the system is at the upper or lower state inside the hysteresis region.

$$\begin{aligned} &z_1(k) = 1 \iff u(k) > \overline{u}, \\ &z_2(k) = 1 \iff u(k) < \underline{u}, \\ &z_3(k) = 1 \iff y(k-1) = \overline{y}, \\ &y(k) = z_1\overline{y} + z_2\underline{y} + (1 - z_1(k))(1 - z_2(k))(z_3(k)\overline{y} + (1 - z_3(k))\underline{y}). \end{aligned} \tag{3.15}$$

In summary, the design parameters of a relay with hysteresis in (3.15) are $\underline{y}$, $\overline{y}$, $\underline{u}$, and $\overline{u}$.

Another discontinuity that is often discussed along the aforementioned is the backlash. An application where this phenomenon occurs is mechanical gears. There are different ways to describe it mathematically, e.g. through polynomial functions or logarithmic approximations of the hysteresis zone with dynamic ranges [136] or even including rotational inertia, elasticity and damping properties with moving ranges as in [5]. Therefore, it can be reduced down to the modelling techniques here without any overhead.

In the presented cases, the phenomena are in a basic configuration. In general the values of $\overline{u}$, $\underline{u}$, $\overline{y}$, $\underline{y}$ and the other coefficients need not to be symmetric. Even more, the functions outside the discontinuity regions can also be more complex than the ones presented here. Still, these complex behaviours can often be reduced to a series of simpler ones.

3.3 Summary

In this chapter, we presented a broad class of control requirements and control scenarios and thus provided a solution to Problem 1. Based on their characteristic, we grouped them into quantitative and qualitative requirements. The quantitative requirements are characterised by a fixed range of spatial uncertainty/flexibility at specified time instances. They are used to model the transient response characteristics. We provided a formulation for each characteristic such that it fits the considered system class.

The qualitative requirements are used to formulate conditional and temporal uncertainties. We presented the formulations for Boolean algebra and temporal operators and their mixed-integer expressions. Additionally, as a particular subclass of qualitative requirements, we discussed the formulations of several discontinuity phenomena. To illustrate the applicability of the quantitative and qualitative requirements, we examined several control scenarios and presented solution approaches for each of them. Therefore, through the quantitative and qualitative requirements, we can include a wide variety of control requirements in the problem set-up and address various control scenarios. Next, we have a look at Problem 2 in Chapter 4, i.e. obtaining guaranteed controller parameterisations such that these requirements are satisfied.

4 Set-based Controller Validation and Parameterisation

> Sherlock Holmes: "When you have eliminated all which is impossible, then whatever remains, however improbable, must be the truth."
>
> Sir Arthur Conan Doyle

In this chapter, we present the theoretical foundations behind the invalidation method used for controller parameterisation. Obtaining a set of feasible controller parameters is achieved through certificates that invalidate sets of the variable space. To do so, we express the problem as a feasibility problem and outline how to use this formulation for set estimation and controller tuning. To do so, we use numerical set estimation approaches, as obtaining the sets analytically is often impossible. We work with two types of set approximations: outer and inner. The outer approximation is used to reduce the search space and aid the control engineer where to find admissible controller parameterisations. An inner approximation procedure looks for solutions that are valid for all possible combinations of uncertain variables. We illustrate the controller parameterisation validation by three examples: safe charging of a Li-ion battery, level control of a two-tank system, and reference change of a magnetic levitation plant. We conclude the chapter with a discussion of the potential challenges and give recommendations on how to overcome them. Some of the results in this chapter have been presented in [7], [9], [194].

4.1 Set-based Estimation

We use set-based estimation (SBE) to find sets of values for the variables of interest, such that the structure of the considered problem, constraints, and measurement information are satisfied. Specifically, we are interested in obtaining values that guarantee the demanded performance.

When performing parameter estimation, it is common to define a fitness criterion and use optimisation to achieve a good fit [197]. In this case, the goal is to find these parameter values that result in system output behaviour that is as close as possible to the measurement data. In contrast to parameter estimation, we are interested in a set of values that fit the model and fulfil the constraints. Thus, we consider set-based

estimation, which demands the requirements to be satisfied for the complete ranges of uncertainties.

4.2 Feasibility Problem Formulation

We are interested in parameter values such that the requirements are satisfied under all conditions. Therefore, we introduce the following definition:

Definition 3. *We denote a set as feasible, if for all elements of the set, the requirements hold for all possible initial conditions, constraints and uncertainties.*

The set-based framework is based on a feasibility formulation. The foundations of the method have been laid out in [35], [36], [186], and expanded to consider binary variables in [195].

We recall the control set-up (2.6) through the feasible set $\mathcal{F}$ that it defines:

$$\mathcal{F} = \left\{ \begin{array}{ll} \text{find} & (x, y, u, p, c, r, z, d, s) \\ \text{s.t.} & x(k^+) = f(x(k), u(k), p(k), d(k)), k \in \mathcal{T}^-, \\ & y(k) = g(x(k), p(k), s(k)), k \in \mathcal{T}, \\ & u(k^+) = h(y(k), r(k), u(k), c), k \in \mathcal{T}, \\ & l(x(k), u(k), y(k), z(k), r(k), k) \geq 0, \\ & x(k) \in \mathcal{X}^k, p(k) \in \mathcal{P}^k, r(k) \in \mathcal{R}^k, \\ & u(k) \in \mathcal{U}^k, y(k) \in \mathcal{Y}^k, s(k) \in \mathcal{S}^k, \\ & d(k) \in \mathcal{D}^k, z(k) \in \mathcal{Z}^k, c \in \mathcal{C}, \end{array} \right. \tag{4.1}$$

'Find' represents the search for feasible values, and thus $\mathcal{F}$ contains all combinations of variables that satisfy all constraints under the category 'subject to'. Therefore, the set $\mathcal{F}$ contains the controller parameterisations that satisfy the desired hybrid dynamical behaviour. As a result of that, obtaining $\mathcal{F}$ is the essential element in providing guaranteed control performance.

4.2.1 Problem Relaxation

Obtaining the feasibility set $\mathcal{F}$ in (4.1) exactly is not a trivial task. The set is, in general, non-convex and can be disjoint due to nonlinearities. To simplify the search, we relax the problem. As a first step, we derive the problem into a quadratically constrained quadratic problem (QCQP). To do so, we construct a monomial basis that we use to describe the system dynamics (4.1)

$$\phi = [1, x_\alpha, y_\beta, c_\gamma, ..., x_\alpha^2, y_\beta^2, c_\gamma^2, ..., x_\alpha y_\beta, x_\alpha c_\gamma, y_\beta c_\gamma, ...] \in \mathbb{R}^{n_\phi},$$

Here $\alpha \in \{1, .., n_x n_t\}$, $\beta \in \{1, .., n_y n_t\}$, $\gamma \in n_c$ correspond to the variables in the problem formulation. Therefore, the dimensions of the vector ϕ reflect the problem size, which grows with each time step and variable.

We note that choosing a monomial basis is not unique, e.g. a monomial ab^2 can be split either into the couple ab and b or into the couple a and b^2.

The QCQP looks like as follows:

$$\text{QCQP} = \begin{cases} \text{find} & \phi \in \mathbb{R}^{n_\phi} \\ \text{s.t.} & \phi^T A_i \phi = 0, i \in \{1, ..n_A\}, k \in \mathcal{T}, \\ & \phi_1 = 1, \\ & B\phi \geq 0, \end{cases} \tag{4.2}$$

where n_A is the number of quadratic dependencies and $B\phi > 0$ models the set constraints and other present inequalities, e.g. requirements or global bounds. Note that a QCQP includes the binary variables. For example, for $a \in \mathbb{R}$ and $a(a-1) \leq 0$ together with $a(a-1) \geq 0$ leads to $a \in \{0, 1\}$. A suitable QCQP formulation can be found for any polynomial system. Solvers that can handle QCQP directly exist, see [80], [92], [150].

In the next step, we relax to problem by introducing the matrix $\Phi = \phi\phi^T$. Additionally, we replace the constraints $\text{tr}(\Phi) \geq 1$ and $\text{rank}(\Phi) = 1$, with $\Phi \succeq 0$. This allows transforming the QCQP into a mixed-integer semi-definite program (MISDP). We can pose the relaxed MISDP as follows:

$$\text{MISDP} = \begin{cases} \text{find} & \Phi \\ \text{s.t.} & \text{tr}(A_i\Phi) > 0, i \in \{1, .., n_A\}, k \in \mathcal{T}, \\ & \text{tr}(e_1 e_1^T) = 1, \\ & B\Phi e_1 > 0, \\ & B\Phi B^T > 0, \\ & \Phi \succeq 0, \end{cases} \tag{4.3}$$

where $e = (1, 0, .., 0)^T \in \mathbb{R}$. In the case that MISDP does not contain any binary variables, the program is semi-definite and can be tackled by standard solvers, e.g. [150], [200], [218]. The presence of integer variables introduces an additional layer of complexity. For solvers capable of handling MISDP see [53], [70].

Although relaxing the problem to an SDP or MISDP renders it easier to solve, we use one further relaxation step and pose it as a mixed-integer linear program (MILP). For a technical discussion on semi-definite vs linear programming for polynomial systems, we refer to [110]. In particular, we relax the condition $\Phi \succeq 0$. For a computational analysis of the proposed relaxation, we refer the reader to [192]. We obtain the following MILP formulation:

$$\text{MILP} = \begin{cases} \text{find} & \phi \\ \text{s.t.} & \text{tr}(A_i\Phi) = 0, i \in \{1, .., n_A\}, \\ & \text{tr}(ee^T\Phi) = 1, \\ & B\Phi e \geq 0, \\ & B\Phi B^T \geq 0. \end{cases} \tag{4.4}$$

Formulating the feasibility problem (4.1) as a MILP allows the use of existing efficient solvers. For the examples presented in this work, we used CPLEX [92] and Gurobi [80] as solvers. We take a look now at the last step needed to perform a certified set-based estimation, namely, if one can still provide guarantees despite the relaxation.

4.2.2 Infeasibility Certificate

From a conceptual point of view, infeasibility means that at least one of the requirements or constraints cannot be satisfied. We note that each optimisation problem can be expressed through a dual formulation. Where instead of the original minimisation, the optimisation is expressed as maximisation and vice versa. We consider a Lagrangian dual formulation for the MILP (D-MILP) as:

$$\text{D-MILP} = \begin{cases} \max & \zeta \\ \text{s.t.} & \Sigma\nu_i A_i + \zeta ee^T + e\lambda_1^T B + B^T\lambda_1 e^T + B^T\lambda_2 B + \lambda_3 = 0 \\ & \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \succeq 0. \end{cases} \tag{4.5}$$

where ν_i and ζ are the scalar dual variables corresponding to the equality constraints of the MILP, and $\lambda_1 \in \mathbb{R}^{n_\zeta}$, $\lambda_2 \in \mathbb{R}^{n_\zeta \times n_\zeta}$, $\lambda_3 \in \mathbb{R}^{n_\zeta \times n_\zeta}$ correspond to the inequality constraints, and λ_3 is symmetric.

The set $\mathcal{Q}$ is the Cartesian product of all spaces that are present in the problem formulation. To perform the SBE, we use a probing set $\mathcal{Q}_P$, which is a subspace of $\mathcal{Q}$, i.e.

$$\mathcal{Q}_P \subseteq \mathcal{Q} \equiv \mathcal{X} \times \mathcal{U} \times \mathcal{P} \times \mathcal{R} \times \mathcal{Y} \times \mathcal{D} \times \mathcal{M} \times \mathcal{C} \times \mathcal{Z} \tag{4.6}$$

In this way, the probing space $\mathcal{Q}_P$ forms a placeholder space, which is chosen depending on the task at hand. The MILP($\mathcal{Q}_P$) and D-MILP($\mathcal{Q}_P$) are the primal and the dual problems.

We use a well-known result regarding the primal-dual relation: if the dual problem is unbounded, denoted by D-MILP($\mathcal{Q}_P$) $\to \infty$, then the primal problem does not contain any feasible solution, i.e. MILP($\mathcal{Q}_P$) $= \emptyset$, see [36], [192], [226]. The second key element to obtain a certification is the fact that the discussed relaxations are conservative, i.e. that all solutions are preserved, and only spurious ones could be added [35], [36], which leads to the following infeasibility certificate:

Definition 4. *If D-MILP($\mathcal{Q}_P$) is unbounded then MILP($\mathcal{Q}_P$) $= \emptyset$. Thus as the MILP($\mathcal{Q}_P$) is a conservative relaxation of FP($\mathcal{Q}_P$), the FP($\mathcal{Q}_P$) does not contain any feasible solutions.*

The formulation of the probing space $\mathcal{Q}_P$ allows focusing on different aspects, such as finding the control parameters. For example, for solving Problem 2 one can choose $\mathcal{Q}_P \subseteq \mathcal{C}$. In such a way, we explore the controller parameter set and invalidate the regions that do not contain any solutions and thus obtain an estimation of the feasible

controller parameters. We continue with discussing strategies on how to obtain an outer and later an inner approximation of the desired space.

4.3 Outer Approximation

We now outline how to efficiently obtain an outer approximation of the feasible set in the desired coordinates. Figure 4.1 contains a geometrical representation of the feasibility set $\mathcal{F}$ and its projection onto the space of interest.

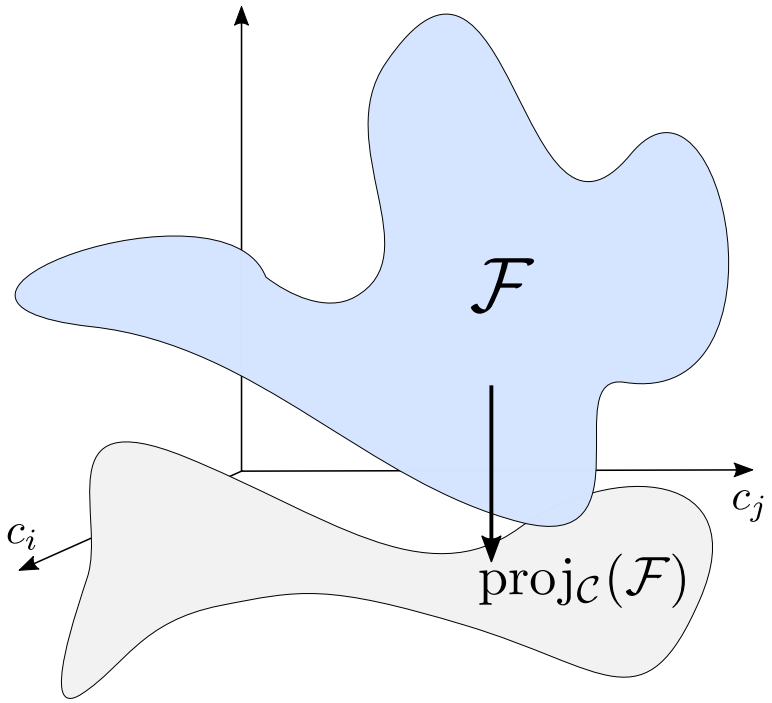


Figure 4.1: Set $\mathcal{F}$ described by the problem set-up and its projection onto two arbitrary coordinates α and β of interest.

In particular, we look for approximations of the projection of $\mathcal{F}$ onto different spaces of interest. For example, to obtain a set of certified controller parameters using the presented feasibility certificates, we project $\mathcal{F}$ onto the controller parameter space $\mathcal{C}$ and look for an outer approximation $\tilde{\mathcal{C}}$ of it, such that $\text{proj}_{\mathcal{C}}(\mathcal{F}) \subseteq \tilde{\mathcal{C}}$, cf. Figure 4.1.

We first focus on outer approximation. Remember that in the set-based estimation, the invalidation of a space $\mathcal{Q}$ is obtained through checking the unboundedness of the corresponding D-MILP($\mathcal{Q}$). This way, the set estimation guarantees that the removed regions $\mathcal{Q}^{\emptyset}$ do not have feasible solutions.

4.3.1 Approximation Techniques

We consider two SBE strategies: *outer-bounding* and *bisectioning*. The outer-bounding procedure employs a hyper-box set that contains the feasible set, and the outer approximation reduces the size of this box. The outer-bounding technique can be performed

either sequentially, for each bound of each variable, or simultaneously, by improving more than one bound at once. The problem formulation and the technical details can be found in [34], [36], [192]. To illustrate the outer-bounding approach, we sketch out consecutive estimation iterations of simultaneous outer-bounding in Figure 4.2. The

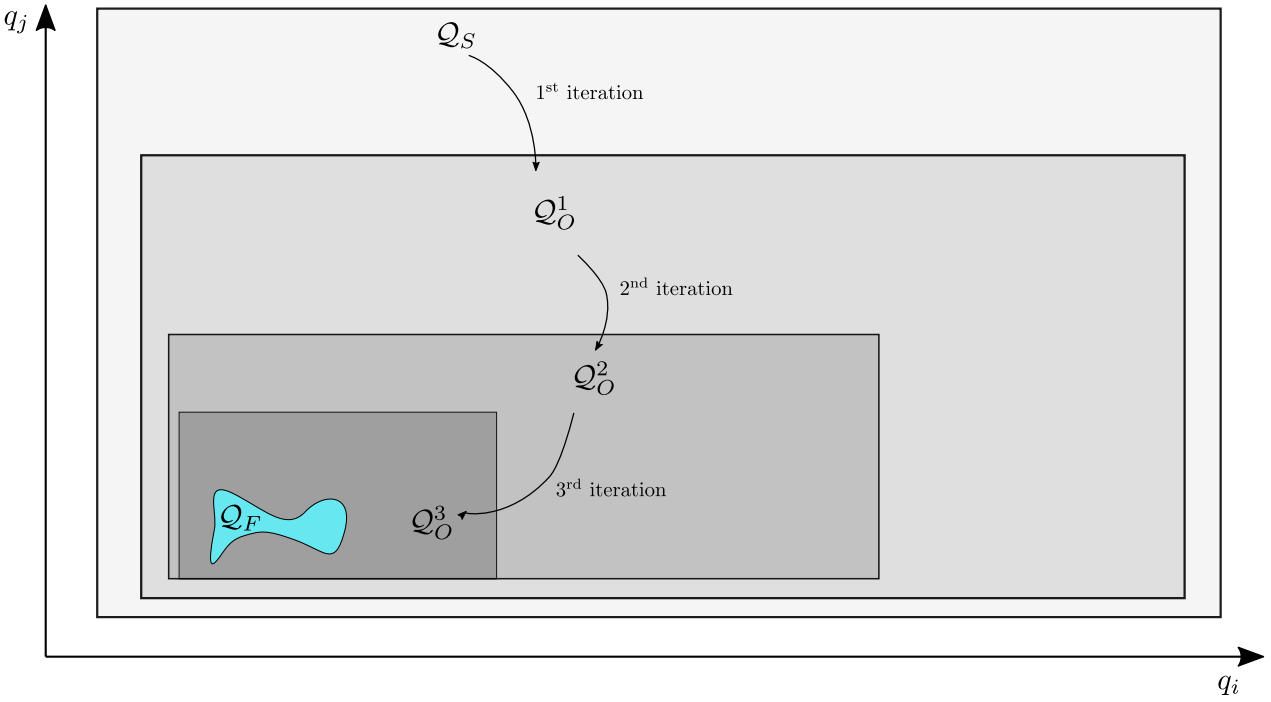


Figure 4.2: Three iteration steps of an outer-bounding approach. The goal is to obtain an outer approximation of $\mathcal{Q}_F$. The sets $\mathcal{Q}_O^1$, $\mathcal{Q}_O^2$, $\mathcal{Q}_O^3$ are respectively the first, second and third approximation iteration.

illustration in Figure 4.2 presents the general case by using the placeholder set $\mathcal{Q}$, i.e. in the general case the projection of the feasibility set onto $\mathcal{Q}$ is

$$\mathcal{Q}_F = \text{proj}_Q(\mathcal{F}).$$

The initial search space is denoted by $\mathcal{Q}_S$. The outer-bounding approximations are expressed as $\mathcal{Q}_O^i$, where $i \in \mathbb{N}^+$ is the estimation iteration. The improvement of estimated set varies with each iteration step. Moreover, it is even possible that only the upper or the lower boundary of one of the dimensions of the hyper-box is updated, which the case for the third iteration in Figure 4.2. All outer-bounding estimation results are nested, i.e.

$$\mathcal{Q}_S \supseteq \mathcal{Q}_O^1 \supseteq \mathcal{Q}_O^2 \supseteq ... \supseteq \mathcal{Q}_O^i, i \in \mathbb{N}^+. \tag{4.7}$$

To provide $\mathcal{Q}_O^i \supseteq \mathcal{Q}_F$ it is recommended to start with wide bounds for the search space. As seen from the examples in this work, the employed SBE can handle uncertainty ranges of several orders of magnitude. Moreover, in most cases, the outer-bounding estimation technique is efficient in shrinking the search space significantly

only in a few iterations. Nevertheless, it results in a hyper-box estimation set, and if the set is highly non-convex or with an odd shape, it might not provide tight estimation results.

In such cases, we can, for example, employ bisectioning. Bisectioning procedure belongs to the split-and-conquer algorithms that split the search space and certify the bisected regions one by one. Although the bisectioning procedure increases the number of invalidations that need to be performed, the computations can be parallelised.

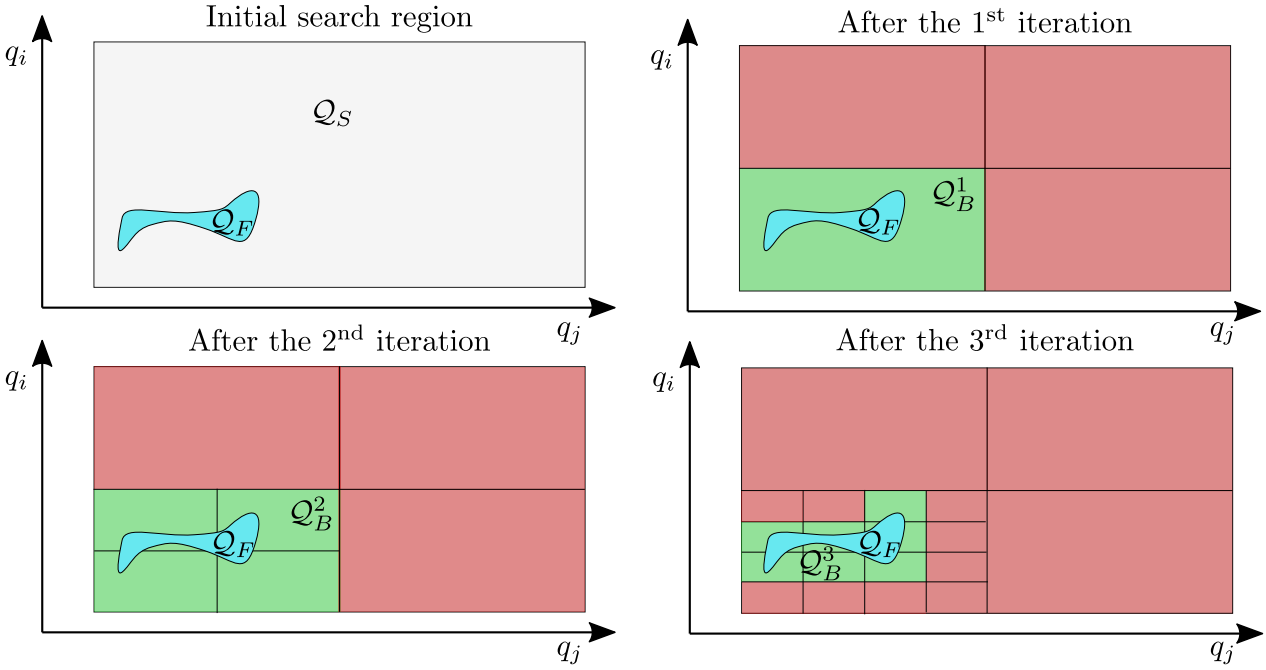


Figure 4.3: Illustrative example of applying bisectioning approach to obtain an outer approximation of the set of interest $\mathcal{Q}_F$. The union of the green sets composes the outer approximation at each iteration.

For the bisectioning, we start with the initial search space $\mathcal{Q}_S$, see Figure 4.3. At each iteration, the not-yet invalidated space is split. Then each region is checkedand if invalidated, it is removed from further consideration. The remaining sets are combined under a set union operation. The estimation result at the i^{th} bisectioning iteration is denoted as $\mathcal{Q}_B^i$. The bisectioning estimation results from the outer approximation can only shrink or remain the same size from one iteration to the next, as no new regions are added, only the old ones have been split and checked, i.e.

$$\mathcal{Q}_S \supseteq \mathcal{Q}_B^1 \supseteq \mathcal{Q}_B^2 \supseteq \ldots \supseteq \mathcal{Q}_B^i \supseteq \mathcal{Q}_F, i \in \mathbb{N}^+.$$

Remark 3. *For easing the discussion, we consider the probing space to be a multi-dimensional box. As indicated in [211], the probing space's geometry can have other shapes, e.g. spherical, ellipsoidal, simplex, etc.*

The main advantage of the bisectioning SBE is that it can provide much finer fidelity estimation results. From a practical point of view, bisectioning the dimension with

the largest absolute difference between the upper and the lower boundary could be beneficial. In contrast, splitting a dimension in which the difference between the upper and lower boundary approaches the machine precision should be avoided. We continue with raising the important question of whether any controller exists at all that can provide the desired behaviour.

4.3.2 Controller Existence

After performing an SBE for obtaining controller parameter values, although undesired, it could turn out that $\hat{\mathcal{C}} = \emptyset$. One reason for such a result could be the wrong parameter search space. Another reason could be that the controller structure, independently of the parameterisation, is incapable of providing the desired behaviour. To answer the question whether there is at all a controller that fits the problem formulation and can achieve the requirements under the system constraints, we introduce the following problem:

Problem 7. *(Set-based controller existence) Does a controller exist that could fulfil the system dynamics, control requirements and initial conditions formulated in* (2.6)*?*

Basically, Problem 7 aims at figuring out at an early stage whether a problem has challenging constraints. To do so, it removes the controller structure from the validation problem and focuses on the system dynamics, requirements, and initial conditions. Using the set-based estimation, we can provide a guaranteed invalidation statement.

In particular, to provide an answer to Problem 7, we ask if there exists an input sequence $\{u(k)\}$ that satisfies the control requirements and system dynamics, By doing so, we can invalidate all controller structures with a single check. We perform this check by treating $u(k)$ as a time-variant variable. Afterwards, we use $u(k)$ itself as the acting controller in the feasibility problem and invalidate it to produce the certificate. To illustrate the idea, we provide a block scheme in Figure 4.4. To depict the key idea in the approach, the input constraints $\mathcal{U}^k$ are shown as a separate saturation element before the system dynamics.

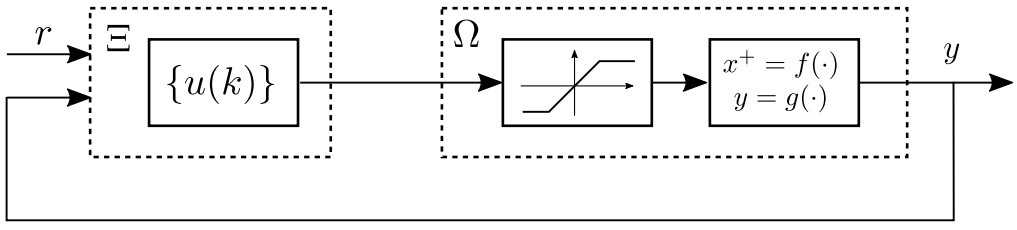


Figure 4.4: Block scheme of the proposed solution to Problem 7. The block $\{u(k)\}$ inside the controller symbolises any possible control sequence that can be created.

By reducing the controller Ξ to sequences of control signals, the controller structure and the parameterisation become irrelevant. To illustrate the idea of considering

a time-variant variable as the acting controller, we provide the following academic example:

Example 5. *Let's consider a rocket car, modelled through a discrete-time double integrator with the following dynamics:*

$$x_1(k^+) = x_1(k) + x_2(k),$$
$$x_2(k^+) = x_2(k) + u(k).$$

where x_1, x_2, and u are the car position, car speed and the control input that corresponds to the car acceleration. The system starts at $x_1(k_0) = 0$, and $x_2(k_0) = 0$. The goal is to design a controller $u = h(\cdot)$ that achieves at least 1000 distance units in two time steps, i.e. the controller requirement is $x_1(k_2) \geq 1000$, but the car has a safety constraint on the acceleration, limiting it between 0 and 10, i.e. $u(k) = [0, 10]$. As the system is linear, we can take the upper limit of u to compute the maximum position that can be achieved in two instances is 30 distance units. Therefore, no controller can fulfil this system set-up.

An obtained infeasibility certificate for Problem 7 guarantees that no controller can respect all requirements. Still, in the case when such a certificate is not obtained, it neither guarantees that a controller exists, nor shows how to obtain one. We continue by looking into a particular example that validates the system performance. To demonstrate the outer approximation through an SBE, we continue with an example with critical safety requirements and thus demanding guaranteed tuning results, namely safe battery charging.

4.3.3 Example - Safe Battery Charging

We continue with an example that looks for the admissible input values with respect to the initial conditions, such that the desired output is achieved and the constraints are obeyed. This example deals with charging a Li-ion battery with a classical constant-current constant-voltage charging policy. This policy consists of two stages. In the first stage, the maximum safe current is applied to the battery, which allows for fast charging. This is carried out until the maximum safe voltage is achieved (in our example, the voltage is 4.2V). Then, in the second charging stage, a constant voltage is maintained. This is achieved by decreasing the applied current until a minimum current is attained. At this stage, the battery is considered fully charged [190]. Exemplary profiles of constant-current constant-voltage cycles are presented in Figure 4.5, together with the corresponding the state of charge profile.

The higher the applied electrical current, the faster the battery reaches higher state of charge levels, and thus the total charging time is faster. However, higher current loads can lead to fast battery degradation. Moreover, for each battery cell, there is a defined $I_{\max}$ that determines the safe charging current. Additionally to the maximum

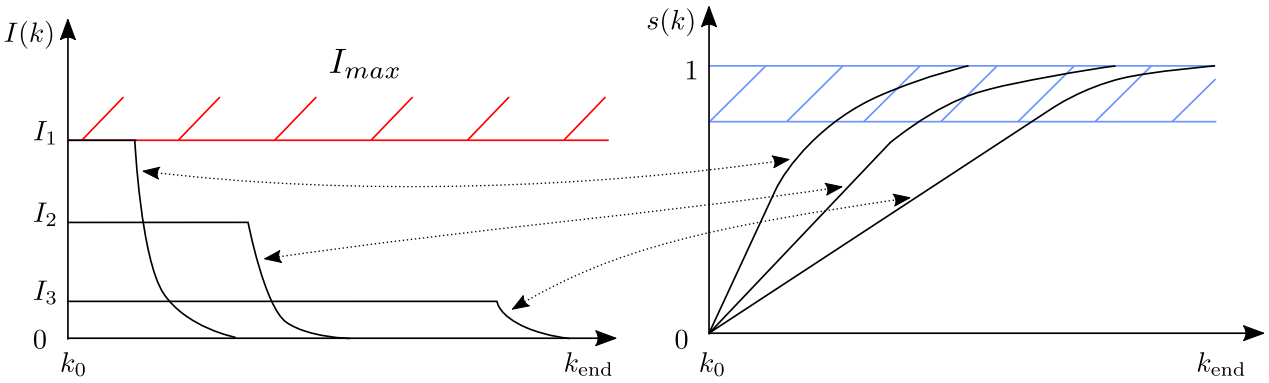


Figure 4.5: On the left are example electrical current profiles of constant-current constant-voltage charging of a Li-ion battery and on the right are the corresponding evolutions of the state of charge of the battery.

current, overcharging and under-charging a battery, as well as high or low temperatures, should be avoided for Li-ion batteries [230].

Model Description

There are different modelling approaches for describing Li-ion battery dynamics [175]. We use an equivalent circuit model that describes the overall electrical behaviour of a battery. Equivalent circuit models are based on fundamental electrical relations and are used in many practical implementations as they have proven to be reliable and sufficiently complex [57], [205].

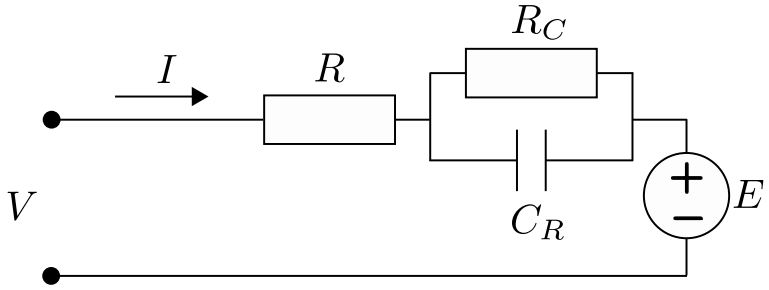


Figure 4.6: Considered simple equivalent circuit model.

The model uses basic electrical elements like resistors, capacitors, voltage sources, see Figure 4.6. Using Kirchoff's law we can derive the following dependency:

$$V(k) = E(k) - V_C(k) - I(k)R. \tag{4.8}$$

Here $V(k)$ is the applied voltage, $I(k)$ is the current that flows through the battery, the resistor R models the internal battery resistance, $V_C(k)$ is the voltage drop of the resistor R_C and a capacitor C_R. We consider a nonlinear voltage source E that corresponds to the open-circuit voltage of the cell, cf. Figure 4.7.

We use the system parameters of a Nokia BP-4L battery [1]. We approximate

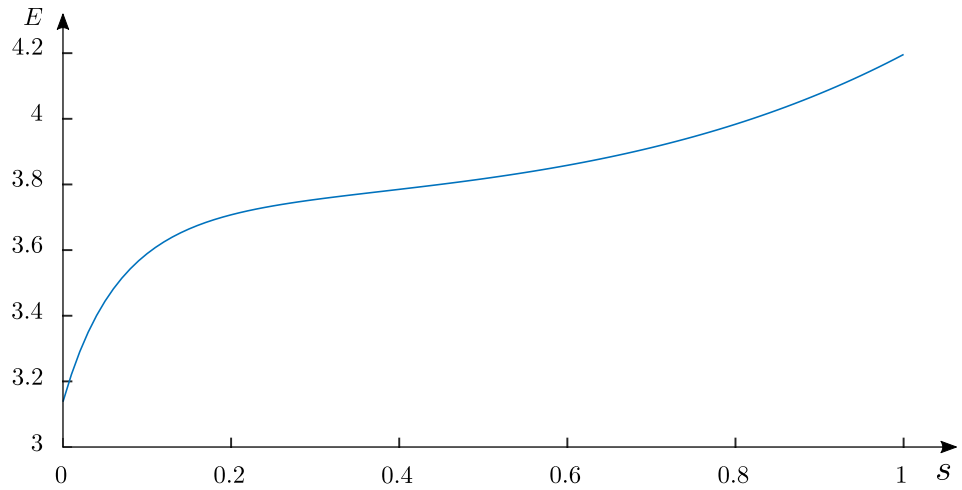


Figure 4.7: Relationship between the open-circuit voltage E and the state of charge s.

the open-circuit voltage E by a rational function $\eta(z(k), \kappa)$. The overall dynamics is described by the voltage drop $V_C(k)$ over the capacitor, and the state of charge $s(k)$.

$$\begin{aligned} &s(k^+) = s(k) + \delta(k) I(k), \\ &V_C(k^+) = V_C(k) + \delta(k)\left(\tfrac{-V_C(k)}{C_R R_C} - \tfrac{I(k)}{C_R}\right), \\ &E(k) = \eta(s(k), \kappa), \\ &\eta(z(k), \kappa) = \kappa_1 + \kappa_2 z(k) + \kappa_3 z(k)^3 + \kappa_4 z(k)^4 + \kappa_5 z(k)^{-1} + \kappa_6 z(k)^{-2}, \\ &z(s(k)) = \xi + s(k)(1 - 2\xi). \end{aligned} \tag{4.9}$$

Here $\delta(k)$ is the sampling time, ξ is a modelling coefficient, $z(k)$ is an auxiliary function that describes the open-circuit voltage. The parameters are presented in Table A.1.

Keeping the temperature of a cell inside the desired range is not only beneficial for the cell performance but is also a critical safety constraint. Heating of a battery occurs when a current passes through the cell, i.e. during charging or discharging. Commonly, the battery temperature is measured only on the cell surface. The considered temperature model for the discussed cylindrical batteries:

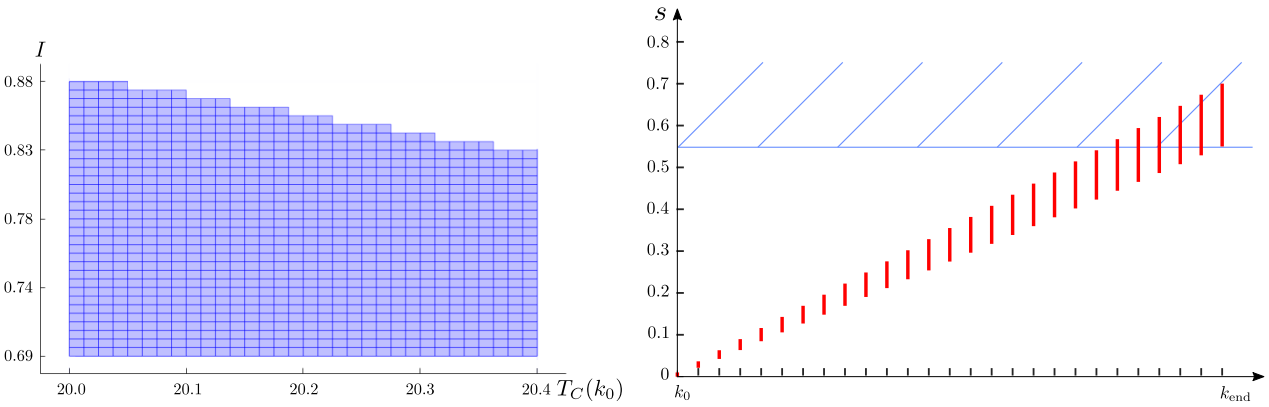
$$\begin{aligned} T_C(k^+) &= T_C(k) + \delta(k)(\gamma(T_S(k) - T_C(k)) + \beta I(k)^2), \\ T_S(k^+) &= T_S(k) + \delta(k)(-\gamma(T_S(k) - T_C(k)) + \alpha(T_{\text{amb}} - T_S(k)). \end{aligned} \tag{4.10}$$

where $T_C(k) \in \mathbb{R}^k$ is the temperature in the core, $T_S(k) \in \mathbb{R}^k$ is the surface battery temperature, T_{amb} is the ambient temperature, and α, β, γ are the thermal coefficients. All variables for the thermal model, electrical model, and initial conditions are presented in Table A.3.

Results and Discussion

The critical phase of battery overheating is typical during the first charging phase - while the maximum current is applied. Therefore, we focus our attention on satisfying

safety requirements during this first charging stage. We require that the battery is charged to at least to 55% within 78 min, i.e. $s(k_{\text{end}}) \geq 0.55, k_{\text{end}} = 78$ min. The sampling time is $\delta(k) = 3$ min. As a safety constrain, we pose $T_C(k) < 20.4$ and for the battery output voltage $V(k) < 4.2$ and $V(k) > 3.15$.



(a) Outer approximation of the maximum safe current depending on the initial core temperature.

(b) Estimated admissible values of the state of charge.

Figure 4.8: The set-based estimation results that provide certified bounds of operation during the battery charging.

The estimation results from the set-based validation are presented in Figure 4.8. The bisectioning estimation results in Figure 4.8a present the range of safe maximum charging current depending on the initial core temperature of the cell. Figure 4.8b presents the estimated ranges of the state of charge at each time steps. The blue region in Figure 4.8b depicts the desired charge level for the cell. The obtained results obey all safety requirements.

Through this safe battery charging example, we presented a scenario in which we look for the safe control inputs with respect to the initial conditions. We continue with a different control scenario, where we investigate the choice of a controller parameterisation depending on a new desired reference.

4.3.4 Experimental Validation - Magnetic Levitation

To illustrate the applicability of the presented tuning approach, we consider the control of a magnetic levitation platform, shown in Figure 4.11. One maglev application example is the maglev trains [83]. The goal is to control the position of a magnetically responsive element that hovers due to an applied magnetic field. The object's position relative to the coil can be controlled by adjusting the strength of the magnetic field, which is generated by applying voltage to an electrical coil.

We use the coil that is in the top part of the plant to control a floating disc. A glass rod passing through the disc eliminates side motions. The friction between the disc and rod is negligible. A laser measuring device provides the distance between the disc and the upper magnet.

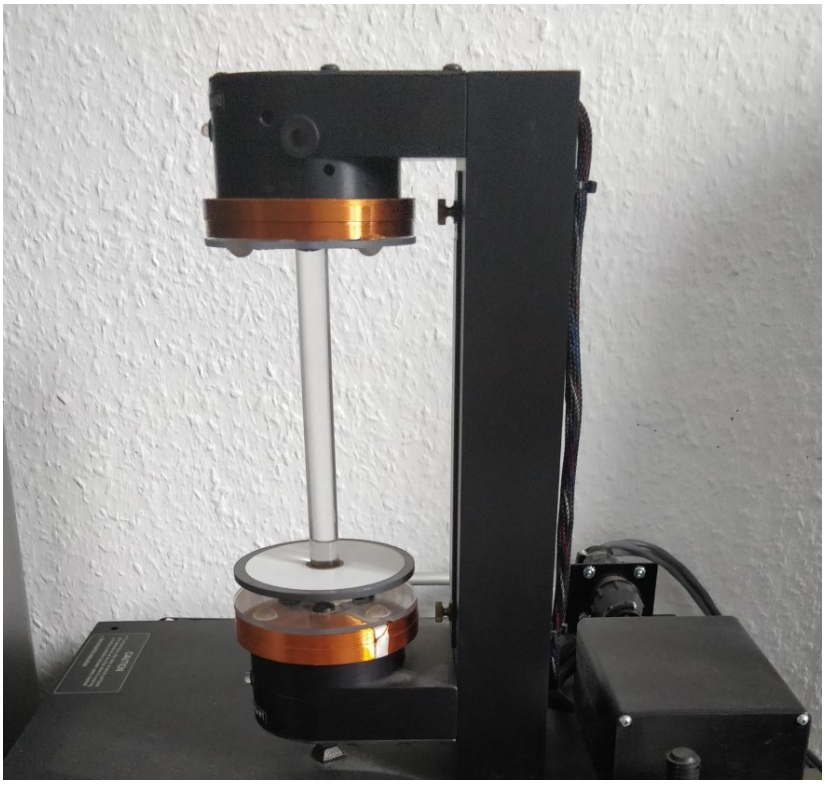


Figure 4.9: Considered magnetic levitation test plant from [130].

We aim to control the unstable configuration, using only the upper coil. The time-discretised dynamical model of the magnetic levitation is given by:

$$\begin{aligned} x_1(k^+) &= x_1(k) + \delta(k)(x_2(k) + e(k)), \\ x_2(k^+) &= x_2(k) + \delta(k)(g - \alpha x_2(k) - \tfrac{u(k)}{ma(x_1(k)+b)^2}), \\ y(k) &= x_1(k). \end{aligned} \tag{4.11}$$

Here x_1, and x_2 are the position and velocity of the levitating disc. The control input u is the applied voltage to the coil. We furthermore consider additive noise $e(k)$, where $e(k) \in [-0.5, 0.5]$. The parameters g, m, a, b and α are the gravity constant, the disc mass, an aggregated parameter, an offset parameter, and the friction coefficient. The system parameters and their corresponding uncertainty ranges are summarised in Table A.4.

As control scenario we consider is a reference change to a new set point $r_{\text{new}} \in [1.5, 2.5]$. The initial conditions are fixed within bounds $x(k_0) \in \mathcal{X}^{k_0}$ and we define control specifications for both the disc position and the disc velocity. As control law, we consider [96]:

$$u(k) = ma(y(k) + b)^2(g + c(y(k) - r_{\text{new}})), \tag{4.12}$$

where c is the controller parameter. We choose as an initial region for the controller parameter $c \in [0, 1000]$. We require the set point change to be completed in $k_{\text{end}} = 0.5$ seconds, and the sampling time is $\delta(k) = 0.02$ seconds.

In the given scenario, we specify the constraints for both states at the end of the control horizon. We limit the second state in order to reach the new position with small velocity and thus accomplish a smooth set-point regulation, leading to the following quantitative requirements: $x_1(k_0) \in [2.99, 3.01]$, $x_2(k_0) \in [-0.01, 0.01]$, $x_1(k_{\text{end}}) \in [r_{\text{new}} - \epsilon, r_{\text{new}} + \epsilon]$ and $x_2(k_{\text{end}} - 1) \in [-0.1, 0.1]$. Here $\epsilon = 0.05$ is the allowed error at the end of the control horizon. Additionally, as global state and input constraints, we set $x_1(k) \in [1, 5]$, $x_2(k) \in [-10, 0.1]$, and $u(k) \in [-0.3, 0.3]$, based on the manufacturer specifications and from experience with the plant.

We consider a projection of the feasible set onto the controller parameter space and the new disc reference. In this way, by choosing a particular reference, we know which admissible controller parameters could provide the desired behaviour. Based on the posed constraints, the estimation returns as the outer-bounding region $c \in [81.63, 99.18]$. This result shrinks the search space and aids the consecutively executed bisectioning procedure. The result of the bisectioning is depicted in Figure 4.10.

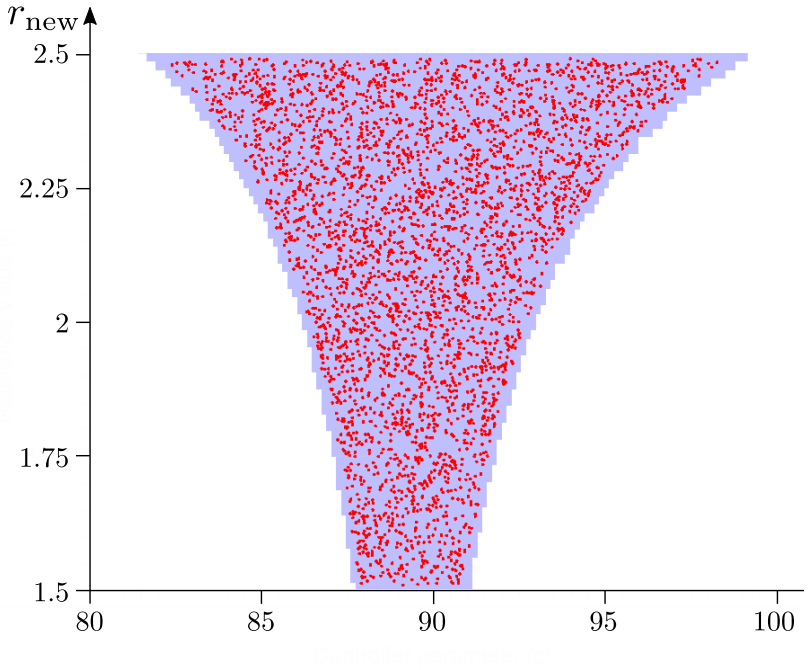
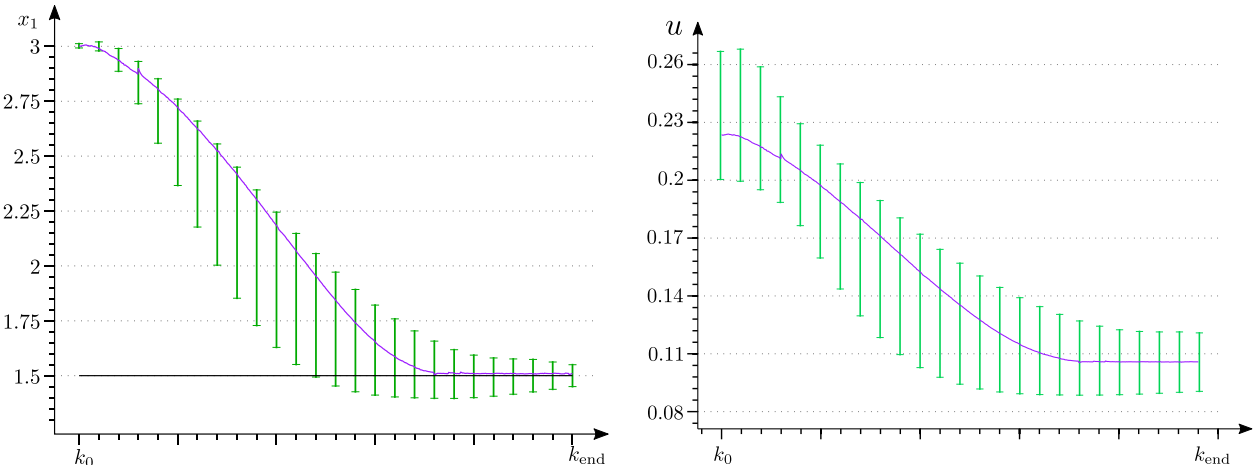


Figure 4.10: The blue set is the outer approximation of the admissible value that describe the relation between the controller parameters and the new reference. The red dots are valid Monte Carlo samples.

In Figure 4.10, we overlapped the bisectioning result, shown as the blue non-convex set, and a generic Monte Carlo sampling, depicted by red dots. All of the Monte Carlo runs are inside the outer approximation from the bisectioning procedure. Moreover, they are relatively equally spread and cover the estimated region well. All those characteristics point to the good quality of the estimation. Moreover, the distance between the Monte Carlo samples and the outer approximation bounds are relative close, which means that the estimation is tight. Still, if desired, the size of bisectioning boxes can be further decreased for higher fidelity of results.

For the implementation on the test plant, we consider a set point change to $r_{\text{new}} = 1.5\,\text{cm}$, i.e. $x_1(k_{\text{end}}) \in [1.45, 1.55]$. The green bars in Figure 4.11 show the guaranteed bounds, and in blue are the plant measurements. Despite the conservatism of the



(a) Disc position **(b)** Control action

Figure 4.11: Comparison of the set-based estimation maximum bounds for the chosen controller parameters (in green) vs. the real states of a run (in purple).

state and input estimation, the actual system delivered a smooth transition during the set-point change and settled successfully within the desired ranges.

Still, choosing and implementing the controller requires an extra step of validating the particular parameterisation. To avoid this step and to be able to obtain values that are valid for all possible combinations of admissible values, we take a look next at inner approximations of the for-all solutions.

4.4 Inner Approximation

The discussed set-based estimation approach allows to obtain outer approximations of the feasible controller parameterisations. Due to the nature of outer approximations, the set can contain parameters that are not feasible, as the outer approximation includes not only the feasible values but also spurious ones. We now outline guaranteed inner approximations of the set containing solutions such that all elements are valid for all combination of uncertainties. We use the following definition:

Definition 5. *A controller parameterisation set containing only valid values under all conditions and uncertainties is referred to as* for-all set*.*

Thus, by obtaining an inner approximation of the for-all set of controller parameter values, we can pick any value and use it directly without further validation.

4.4.1 Problem Formulation

There are three instrumental feasibility problem formulations and the three qualitatively different sets these formulations describe. These three sets are used in obtaining an inner approximation of the for-all feasibility set. We refer to them as the unconstrained, the constrained, and the inverted feasibility problem. So far, we considered the constrained feasibility problem (CFP), which describes $\mathcal{F}$ from (4.1). The unconstrained feasibility problem is a relaxed version of (4.1). We refer to it as unconstrained feasibility problem (UFP) and the set correspondingly by $\mathcal{F}_U$. The third feasibility formulation consists of the initial feasibility problem UFP together with a tailored constraint inversion. We denote this inverted feasibility problem formulation (IFP) and the for-all set it describes with $\mathcal{F}_A$. The following derivations are based on ideas outlined in [184] and [211].

To demonstrate how to use the three feasibility problems for our purposes, we start with the UFP. It contains of the system dynamics, the controller dynamics, and the uncertainty ranges for each variable:

$$\mathrm{UFP}(\mathcal{Q}_P) \quad \begin{cases} \text{find} & q \\ \text{s.t.} & q \in \mathcal{Q}_P, \\ & x(k^+) = f(x(k), u(k), p(k), d(k)), k \in \mathcal{T}^-, \\ & y(k) = g(x(k), p(k), s(k)), k \in \mathcal{T}, \\ & u(k^+) = h(y(k), r(k), u(k), c), k \in \mathcal{T}^-, \\ & x(k) \in \mathcal{X}^k, p(k) \in \mathcal{P}^k, r(k) \in \mathcal{R}^k, \\ & u(k) \in \mathcal{U}^k, y(k) \in \mathcal{Y}^k, s(k) \in \mathcal{S}^k, \\ & d(k) \in \mathcal{D}^k, c \in \mathcal{C}. \end{cases} \tag{4.13}$$

Here $\mathcal{Q}_P$ is the probing space, as defined in (4.6). If we set $\mathcal{Q}_P$ to the global safety bounds, then $\mathrm{UFP}(\mathcal{Q}_P)$ results in the set $\mathcal{F}_U$. Hence, we recommend for UFP to use $\mathcal{Q}_P$ to explore all dimensions present in the problem. Such exploration can provide a tighter estimation of $\mathcal{F}_U$ and thus a more precise description of $\mathcal{F}_U$. Next, we define the CFP, which based on the UFP adds the controller requirements $L_i, i \in \{1, ..n_l\}$

$$\mathrm{CFP}(\mathcal{Q}_P) \quad \begin{cases} \text{find} & q \\ \text{s.t.} & q \in \mathcal{Q}_P, \\ & (x, y, u, p, r, s, d, c) \in \mathcal{F}_U, \\ & l_i(x(k), u(k), y(k), z(k), r(k), k) \geq 0, i \in \{1, .., n_l\}, \\ & z(k) \in \mathcal{Z}^k. \end{cases} \tag{4.14}$$

Here n_l is the number of requirements that we are interested in guaranteeing despite all uncertainties. The CFP describes the admissible set $\mathcal{F}_C$. Obtaining an outer approximation of $\mathcal{F}_C$ was outlined in Section 4.3.

The third feasibility problem uses a tailored inversion of the controller requirements to describe the for-all set. This constraint inversion enforces that at least one of the

control requirements L_i is violated. Therefore, the IFP describes the space where some or none of the constraints might be satisfied. Thus by invalidating a probing region with the IFP, we are certain that this space contains only solutions that satisfy all constraints for all uncertainties, as long as it is a subset of $\mathcal{F}_U$. To construct this inversion, we use qualitative requirements, where a binary variable w_i is assigned to the validity of each constraint L_i and enforcing the sum of all binary variables w_i to be less than the number of control requirements. This leads to the following feasibility problem formulation:

$$\text{IFP}(\mathcal{Q}_P) \begin{cases} \text{find} & q \\ \text{s.t.} & q \in \mathcal{Q}_P, \\ & (x, y, u, p, r, s, d, c) \in \mathcal{F}_U, \\ & w_i = 1 \Leftrightarrow l_i(x(k), u(k), y(k), z(k), r(k), k) \geq 0, i \in \{1, .., n_l\}, \\ & \sum_{i=1}^{n_l} w_i \leq n_l - 1, \\ & z(k) \in \mathcal{Z}^k, w_i \in \{0, 1\}. \end{cases} \tag{4.15}$$

Through the constraint inversion, we describe the complement set $\mathcal{F}_M$ to the for-all set $\mathcal{F}_A$. The set $\mathcal{F}_M$ also contains values that do not satisfy the global uncertainty bounds. Therefore, to exclude them and obtain only the for-all values to the problem, we use the set $\mathcal{F}_U$. We perform the following step $\mathcal{F}_A = \mathcal{F}_U \backslash \mathcal{F}_M$ to end up with only the for-all solutions. Deciding the space on which to project $\mathcal{F}_U$, $\mathcal{F}_C$ and $\mathcal{F}_A$ is one of the key elements in solving the problems. Therefore, to keep the formulation correct for the various control tasks, we use the generic space of interest $\mathcal{Q}$. Hence, we define the following three projections:

$$\begin{aligned} \mathcal{Q}_U &:= \text{proj}_{\mathcal{Q}} \mathcal{F}_U, \\ \mathcal{Q}_F &:= \text{proj}_{\mathcal{Q}} \mathcal{F}_C, \\ \mathcal{Q}_A &:= \text{proj}_{\mathcal{Q}} \mathcal{F}_A, \end{aligned}$$

where $\mathcal{Q}_U$, $\mathcal{Q}_F$, and $\mathcal{Q}_A$ are the exact projections of the UFP, CFP, IFP onto $\mathcal{Q}$. For example, when we are interested in obtaining controller parameterisations, then $\mathcal{Q} \equiv \mathcal{C}$. In some of the more complex scenarios, $\mathcal{Q}$ could also include the references or the initial conditions of the dynamic system, see in Chapter 5.

We depict in Figure 4.12 the inner and outer approximations, as well as the qualitative difference between them. For illustrative purposes, we consider multi-dimensional feasibility sets in a 3-dimensional space, where $\mathcal{Q}$ is the space onto which the problem is projected. For illustrative purposes, the projection space is two-dimensional.

Figure 4.12 depicts the sets $\mathcal{F}_U$, $\mathcal{F}_C$, and $\mathcal{F}_A$ and their projections $\mathcal{Q}_U$, $\mathcal{Q}_F$, and $\mathcal{Q}_A$. As discussed, we rely on SBE methods to obtain their approximations. Analogously to the OA, we use the presented feasibility problems to check for the unboundedness of the Lagrangian dual of each of the problems. For example, if D-IFP$(\mathcal{Q}_P) \rightarrow \infty$, then it is guaranteed that the IFP$(\mathcal{Q}_P)$ contains only for-all feasible solutions and thus $\mathcal{Q}_P$

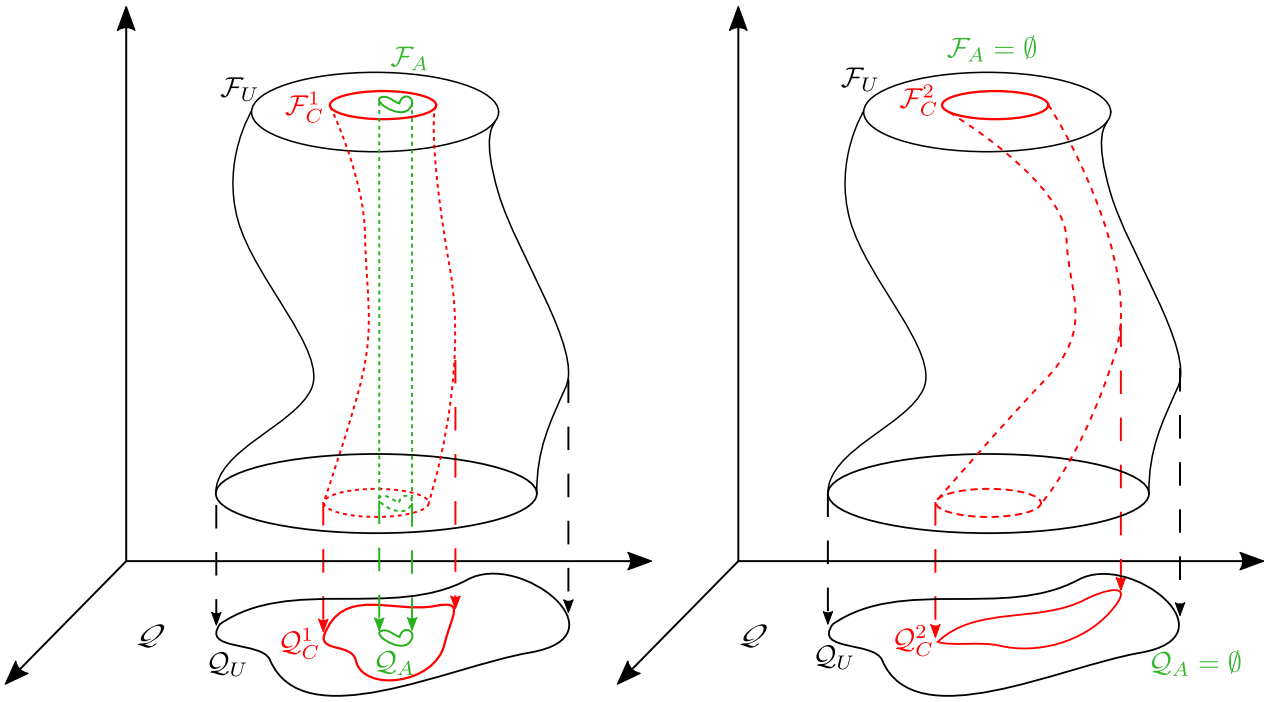


Figure 4.12: Set projections of the UFP, CFP and IFP for two different controller requirements that define two different feasibility sets $\mathcal{F}_C^1$ and $\mathcal{F}_C^2$. The figure on the left contains a set with for-all values. The one on the right, has too restrictive control requirements and thus there is no longer a pair (q_i, q_j) that is valid for all uncertainties, i.e. $\mathcal{F}_A = \emptyset$.

is part of the inner approximation of the projection of the for-all solutions.

To systematise the estimation procedure and to give insight into the technicalities of how to use the presented SBE methods, we provide Algorithm 4. Through it, we start with the search space $\mathcal{Q}_S$ and obtain an inner approximation of the projection of the for-all set $\mathcal{Q}_\mathcal{I}$, an outer approximation of the projection of the feasible values by obtaining the space $\mathcal{Q}_\emptyset$ that is guaranteed to not contain any feasible solutions, and the undetermined space $\mathcal{Q}_\mathcal{U}$ between them.

Algorithm 4 provides a systematic approach of probing, validating and classifying the space of interest. The resulting three sets are $\mathcal{Q}_U$, $\mathcal{Q}_I$, $\mathcal{Q}_\emptyset$, where

$$\mathcal{Q}_U \bigcup \mathcal{Q}_I \bigcup \mathcal{Q}_\emptyset = \mathcal{Q}_S. \tag{4.16}$$

By increasing the split depth, one can obtain a better resolution for each of the sets. Note that the gap between the outer approximation of the admissible set and the inner approximation of the for-all set does not need to close as they approximate different sets, as shown in Figure 4.12. Still, relation (4.16) holds independently of the gap size between $\mathcal{Q}_F$ and $\mathcal{Q}_A$ and the existence of $\mathcal{Q}_A$. We continue with a scenario that uses Algorithm 4 for the control tuning of a two-tank system.

Algorithm 4 Inner & Outer approximation (IOA)

Input: $\mathcal{Q}_S$
Output: $(\mathcal{Q}_U, \mathcal{Q}_I, \mathcal{Q}_\emptyset)$

set $\mathcal{Q}_U := \emptyset,\ \mathcal{Q}_I := \emptyset,\ \mathcal{Q}_\emptyset := \emptyset$
if D-CFP$(\mathcal{Q}_S) < \infty$ **then**
  if D-IFP$(\mathcal{Q}_S) < \infty$ **then**
    if split depth is **not** reached **then**
      partition $\mathcal{Q}_S \to \{\mathcal{Q}_1, ..., \mathcal{Q}_N\}$
      for $i \in \{1, \ldots, N\}$ **do**
        call recursively $(\mathcal{Q}_U^*, \mathcal{Q}_I^*, \mathcal{Q}_\emptyset^*) :=$ IOA$(\mathcal{Q}_i)$
        $\mathcal{Q}_U := \mathcal{Q}_U \cup \mathcal{Q}_U^*$
        $\mathcal{Q}_I := \mathcal{Q}_I \cup \mathcal{Q}_I^*$
        $\mathcal{Q}_\emptyset := \mathcal{Q}_\emptyset \cup \mathcal{Q}_\emptyset^*$
      end for
    else
      set $\mathcal{Q}_\mathcal{U} := \mathcal{Q}_S$
    end if
  else
    set $\mathcal{Q}_\mathcal{I} := \mathcal{Q}_S$
  end if
else
  set $\mathcal{Q}_\emptyset := \mathcal{Q}_S$
end if
return $(\mathcal{Q}_U, \mathcal{Q}_I, \mathcal{Q}_\emptyset)$

4.4.2 Example - Two-tank System

To illustrate Algorithm 4, we consider a control scenario in which we are interested in controlling the levels of two connected tanks, see Figure 4.13.

Model description

The controlled inflow $u(k)$ fills Tank 1. The two tanks are connected at the bottoms with a pipe. The second tank has the output flow of the system.

The time-discrete model governing the levels of the liquid inside the two tanks are derived from first principles using Torricelli's law. We combine the technical parameters into an aggregated parameter $p = as\sqrt{2g}$, where $a = 0.45\,\mathrm{cm}^2$ is the pipe coefficient, $s = 0.1578\,\mathrm{cm}^2$ is the cross-section of the pipes, $g = 981\,\mathrm{cm/s}^2$ is the standard gravity, leading to the following dynamics:

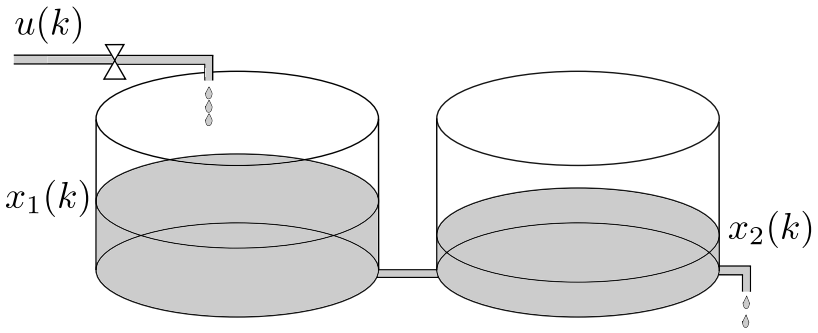


Figure 4.13: Illustration of the considered connected two-tank system.

$$\begin{aligned} x_1(k^+) &= x_1(k) + \delta(k)(u(k) - p\sqrt{x_1(k) - x_2(k)})/A, \\ x_2(k^+) &= x_2(k) + \delta(k)p(\sqrt{x_1(k) - x_2(k)} - \sqrt{x_2(k)})/A. \end{aligned} \tag{4.17}$$

Here $x_1 \in \mathbb{R}^+$, $x_2 \in \mathbb{R}^+$ are the levels in the first and the second tank. The sampling time $\delta(k)$ is 2 s and $A = 28\,\mathrm{cm}^2$ is the cross-section of the tanks. We require that the level of Tank 1, i.e. x_1, needs to be two times more than the level of Tank 2, i.e. x_2. We express the difference between the tank levels and the references as $e_1(k) = 2r_{\text{desired}} - x_1(k)$ and $e_2(k) = r_{\text{desired}} - x_2(k)$. Both errors are combined into an aggregated error $E(k) = c_1 e_1(k) + c_2 e_2(k)$ that is used in the controller given by:

$$u(k) = u(k^-) + (E(k) - E(k^-)) + c_3 \delta(k) E(k^-).$$

Using the formulation from Section 3.2, we express the saturated input by

$$\begin{aligned} z_1(k) = 1 &\Leftrightarrow u(k) \leq \overline{u}, \\ z_2(k) = 1 &\Leftrightarrow u(k) \geq \underline{u}, \\ u(k) &= z_1(k) z_2(k) u(k) + (1 - z_1(k))\overline{u} + (1 - z_2(k))\underline{u}. \end{aligned}$$

Here $\overline{u} = 3\,\mathrm{cm}^3/\mathrm{s}$, $\underline{u} = 0\,\mathrm{cm}^3/\mathrm{s}$. The initial conditions are $x_1(0) = 18 \pm 0.02\,\mathrm{cm}$ and $x_2(0) = 9 \pm 0.01\,\mathrm{cm}$. There is a safety constraint for the maximum allowed level for each tank: $x_i(k) \leq 50, i = \{1, 2\}$. We choose as starting search space for $c_1 \in [0.01, 0.51]$, $c_2 \in [0.85, 1.85]$, and the third parameter is fixed to $c_3 = 0.01$. The new reference is $r_{\text{desired}} \in [10, 10.01]$. The control horizon consists of seven steps, i.e. $\mathcal{T} = \{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2\}$. To ensure that we achieve a smooth reference change, we pose additionally $e_2(k_{end}) \leq 0.1$ and $E(k_{end}) \leq 1$. Using the SBE Algorithm 4, we obtain the results in Figure 4.14. We used ADMIT [210] to formulate the problem using CPLEX as the solver [92].

The estimation results in Figure 4.14 demonstrate successful outer and inner approximation for the two control parameters c_1 and c_2. All grey regions, i.e. the upper left and lower right blocks, are certified not to contain any feasible solutions. The union of the red blocks is the inner approximation of the for-all values for the chosen

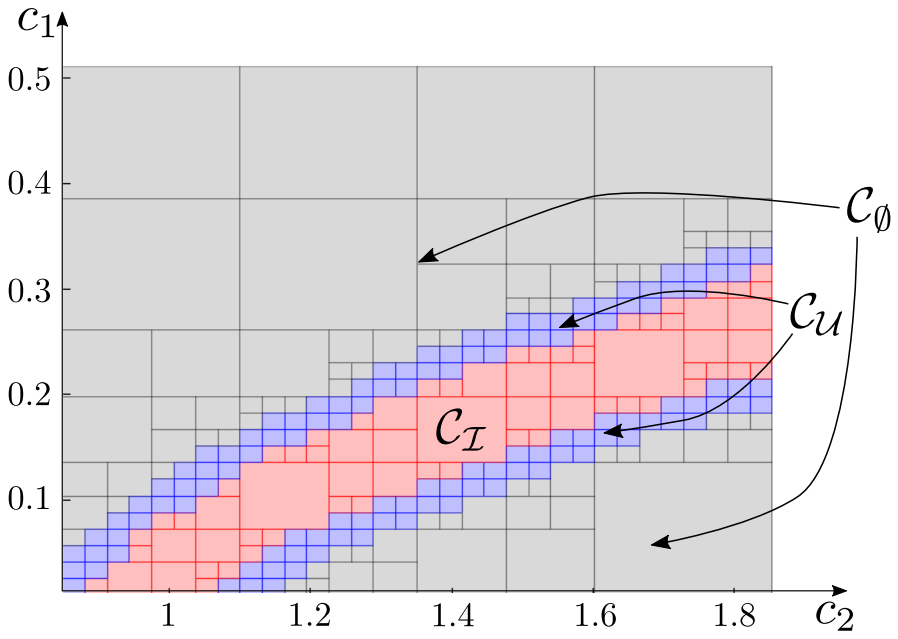


Figure 4.14: Set-based estimation results for the two-tank system. The grey areas are the outer approximation, the red are the inner approximation, and the blue are inconclusive regions.

controller under the particular control scenario. Moreover, the region between them, depicted in blue, is where the boundaries of the feasible set and the for-all set lie. If desired, a higher granularity of results is obtainable by splitting the probed spaces further. In conclusion, implementing a parameterisation $(c_1^*, c_2^*) \in \mathcal{C}_\mathcal{I}$ (i.e. from the red region) guarantees the fulfilment of the constraints despite all the uncertainties in the system.

4.5 Summary

We conclude this chapter with a proposed workflow for structured control tuning. Additionally, we discuss some of the technical challenges that could occur and explain how to overcome them.

4.5.1 Overview of the Set Relations

From a design point of view, six distinctively different sets are important for the presented set-based approach. We discuss them in the projection space $\mathcal{Q}$: the search space $\mathcal{Q}_S$ defines the region to be checked. Next, by using the presented outer-bounding method, we obtain $\mathcal{Q}_O$. Taking $\mathcal{Q}_O$ as an initial set for the bisectioning, we obtain $\mathcal{Q}_B$, which in the general case provides a better fidelity of the shape of $\mathcal{Q}_F$. Furthermore, inside the projection $\mathcal{Q}_A$ of the for-all set is its inner approximation $\mathcal{Q}_I$, see also Figure 4.15.

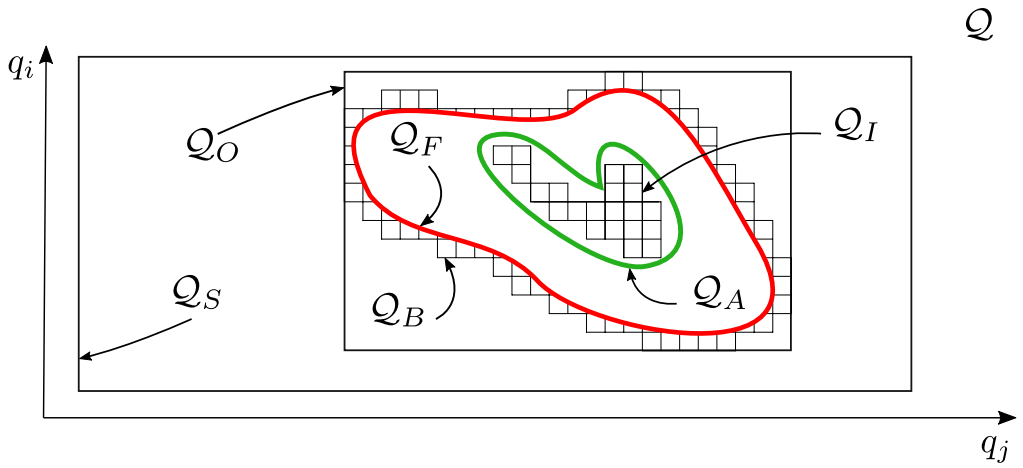


Figure 4.15: An example for the geometrical relation of the projections.

Note that some of the set boundaries can intersect, e.g. of $\mathcal{Q}_I$ and $\mathcal{Q}_S$ as in Figure 4.14. If we perform the outer-bounding and use it as the initial search region for the bisectioning, then the six sets have the following set relation:

$$\mathcal{Q}_S \supseteq \mathcal{Q}_O \supseteq \mathcal{Q}_B \supseteq \mathcal{Q}_F \supseteq \mathcal{Q}_A \supseteq \mathcal{Q}_I$$

The approximation of the projection of the UFP, although instrumental for the construction of the inner approximation, is practically not necessary. This estimation is not required because the invalidated region of the inner approximation can be intersected with the projection of the CFP. Moreover, not estimating the set $\mathcal{Q}_U$ saves computational time. To better understand the set-based controller tuning, we propose a workflow that outlines the essential steps in the design process.

4.5.2 Set-based Controller Design - Analysis & Discussion

Having discussed the set relations, we outline how bounding methods can be used for controller tuning, see Figure 4.16. First, we start with the 'Formulation of the feasibility problem'. Next, one can perform 'Outer-bounding approximation' in parallel to 'Constraint inversion'. The outer bounding results can be used as the initial search region for the 'Bisectioning approximation'. The results from the bisectioning approximation can be used as the search region for the 'Inverted feasibility problem', i.e. the inner approximation of the for-all set. In the end, we compare the obtained sets and decide whether further refinement of the results is needed, or we can proceed with picking a controller parameterisation and implementing it.

There are several general suggestions if the set-based design does not return successful estimation results. For example, one can consider a different controller structure or other ranges for the controller parameters that could lead to the correct solution. Moreover, if obtaining an outer approximation is challenging, then relaxing the requirements or the initial conditions expands the size of the feasible set and thus helps

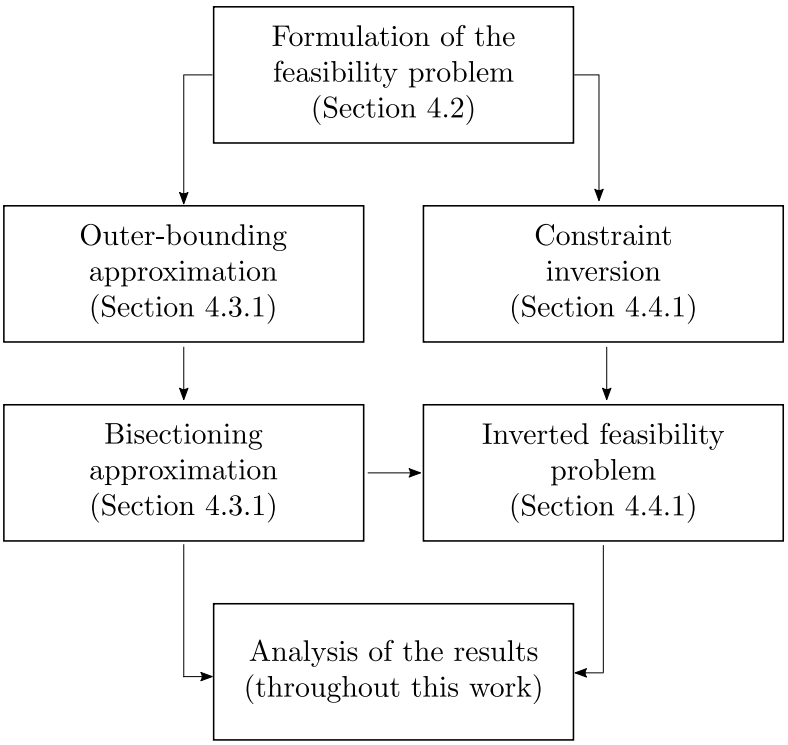


Figure 4.16: Set-based controller design workflow.

the estimation. The lack of an inner approximation could be due to too wide ranges of uncertainties in the system. In such a case, strategies like improved system identification or better sensors with higher precision could be advantageous. These suggestions outline the major strategies for tackling potentially unsatisfactory results.

When considering the effort and time invested in obtaining the estimation, we should keep in mind that we obtain guaranteed results. In Chapter 5, we use the results from Chapter 3 and 4 to focus on parameterising and validating an advanced control scenario on the process level for closed-loop control.

5 Robust Error-free Steady-state Control

> Es ist nicht genug zu wissen, man muß auch anwenden; es ist nicht genug zu wollen, man muß auch tun.
>
> Johann Wolfgang von Goethe

In this chapter, we deal with validating a control design for robust error-free steady-state control for a long time, which is a generalisation of the steady-state error control requirement from Chapter 3. We are interested in obtaining a set of controller parameterisations that fulfil the requirements for all initial conditions and all references from specified sets. We tackle this challenge by splitting it into two tasks. The first one is the set-point regulation that steers the system to a new reference. The second task maintains the system states and outputs bounded by imposing (periodic) set invariance. To address the for-all requirement for the set of initial conditions and references, we use the inverse-constrain formulation from Chapter 4. We re-formulate also the system dynamics in error coordinates. The obtained parameterisation provides a bounded stability statement for the system. The results in this chapter are based on and expand the presented results in [8].

5.1 Challenges and Problem Formulation

The error-free steady-state control is a fundamental process control requirement [59], [85], [154], [156], [191]. Section 3.2 outlined the basic formulation by specifying the size of the error margin in the transient response. We present an advanced version that is not only valid for a single reference, but it is also not limited to a finite-time consideration. We aim for a set of closed-loop controller guaranteed parameterisations $\tilde{\mathcal{C}}$ for a given controller structure Ξ.

Here, we are interested not only in achieving a particular new desired reference r_{new}, as in the basic case, but any one from a given set $\mathcal{R}_{\text{new}}$, i.e. $r_{\text{new}} \in \mathcal{R}_{\text{new}}$, which implies that the references need to be among the admissible output values, i.e. $\mathcal{R}_{\text{new}} \subseteq \mathcal{Y}^{k_{\text{new}}}$. Furthermore, any new references must be achieved starting from any initial condition $x(k_0) \in \mathcal{X}^{k_0}$, or formulated in the system output space, starting from any $y(k_0) \in \mathcal{Y}^{k_0}$. In short, the controller must bring the system to any new reference (from a specified set), starting from any initial condition (from a specified set). As achieving the exact reference for uncertain system is a very strong requirement, we relax this condition requirement only to achieve it up to a certain precision $\epsilon \in \mathbb{R}^{n_y}$ at k_{new} and afterwards

stay inside, i.e.

$$y(k_{\text{new}}) \in \mathcal{Y}_{\text{new}}(r_{\text{new}}) := [r_{\text{new}} - \epsilon, r_{\text{new}} + \epsilon], r_{\text{new}} \in \mathcal{R}_{\text{new}}. \tag{5.1}$$

Figure 5.1 illustrates the desired behaviour, where r_1 and r_2 are two examples for r_{new}. The initial conditions and the new reference are depicted in the output space, respectively $\mathcal{Y}^{k_0}$ and $\mathcal{Y}^{k_{\text{new}}}$. The horizontal stripes denote the desired steady-state precision margin.

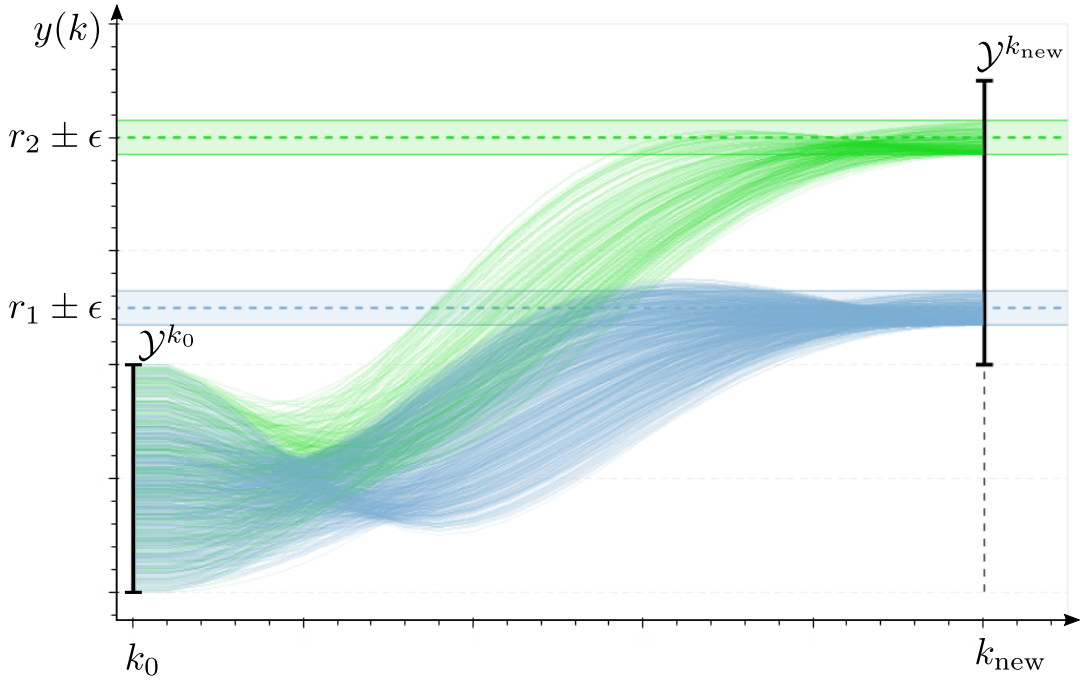


Figure 5.1: Robust steady-state error-free control for any reference value $\mathcal{Y}^{k_{\text{new}}}$, starting from any initial conditions from $\mathcal{Y}^{k_0}$. Two example values with their desired precision $r_1 \pm \epsilon$ and $r_2 \pm \epsilon$ are shown.

Error-free steady-state performance is an often demanded objective, spanning from the model predictive community [165] to PID control [13]. In practice, there is always a mismatch, which leads to an offset at the steady-state [179], [183]. To overcome this problem in model-based control, additional models can be developed to compensate the offset and thus achieve the requirement [27], [129], [166]. We encompass (i.e. over-approximate) the mismatch and work with a conservative model.

In difference to Section 3.1, we require the system outputs $y(k)$ to stay close to r_{new} endlessly long beyond k_{new}. This leads to the following problem formulation:

Problem 8. *(Set-based error-free steady-state control) Obtain a set of controller parameterisations $\hat{\mathcal{C}}$ such that a controller Ξ is able to steer the system Ω, starting from any initial condition $x(k_0) \in \mathcal{X}^{k_0}$, to any references $r_{new} \in \mathcal{R}_{new}$ with an ϵ precision within finite time k_{new}, i.e. $y(k_{new}) \in \mathcal{Y}_{new}(r_{new})$. For $k > k_{new}$, the system outputs $y(k)$ and states $x(k)$ have to stay bounded, and the outputs have to re-visit periodically the precision set $\mathcal{Y}_{new}(r_{new})$.*

Problem 8 combines the set-point regulation and stabilisation for a region of interest. Furthermore, the system maintains bounded and respects the dynamics' constraints. We note that Problem 8 is formulated such that for every output, there is a corresponding new reference. In many cases, this is not needed. The solution approach is developed to be valid in both cases, i.e. for some or all outputs.

To solve Problem 8, we use two ideas. First, we re-formulate the problem in error coordinates. Second, we split the control task into two stages. We continue by presenting the technical details of the proposed approach.

5.2 Solution Approach

Although the linear case has been extensively studied, and analytical solutions exist [59], any additional dynamical phenomena increase the complexity of the task significantly. Even the perturbed linear system case remains nowadays of interest [112].

The solution approach we propose consists of three steps. In the first one, we transform the system into error coordinates. The second step delivers the controller parameterisations that provide robust set-point regulation using the transformed system. The last step requires the system outputs to end in the precision set.

5.2.1 Error Coordinate Transformation

All new references r_{new} need to be achieved with an ϵ precision. All these references are contained in a set $\mathcal{R}_{\text{new}}$. Still, each reference is considered together with the precision defines another set $\mathcal{Y}(r_{\text{new}}) := [r_{\text{new}} - \epsilon, r_{\text{new}} + \epsilon]$. As a result, the uncertainties of these two sets $\mathcal{R}_{\text{new}}$ and $\mathcal{Y}(r_{\text{new}})$ overlap. To separate the uncertainty of the precision from the set of references, we express the system dynamics in error coordinates. Similar error-coordinate transformations are used in many control approaches, such as path-following strategies [3], funnel control [94], and sliding-mode control [221]. To re-formulate the system dynamics, we apply the following coordinate transformation:

$$e(k) = y(k) - r_{\text{new}}, \tag{5.2}$$

where $e(k) \in \mathcal{E}^{n_y} \subset \mathbb{R}^{n_y}$ is the substitution variable. For ease of discussion, we assume that there is a reference value for each output, i.e. $n_r = n_y$. Applying the transformation (5.2) to the feasibility problem (4.1), we obtain the following error-

coordinate formulation:

$$\mathrm{EFP}(\mathcal{Q}) = \begin{cases} \text{find} & q \\ \text{s.t.} & x(k^+) = f(x(k), u(k), p(k), d(k)), k \in \mathcal{T}^-, \\ & e(k) = g_e(x(k), p(k), s(k), r), k \in \mathcal{T}, \\ & u(k^+) = h_e(e(k), r(k), u(k), c), k \in \mathcal{T}, \\ & l(x(k), u(k), e(k), z(k), r(k), k) \geq 0, \\ & x(k) \in \mathcal{X}^k, p(k) \in \mathcal{P}^k, r(k) \in \mathcal{R}^k, \\ & u(k) \in \mathcal{U}^k, e(k) \in \mathcal{E}^k, s(k) \in \mathcal{S}^k, \\ & d(k) \in \mathcal{D}^k, z(k) \in \mathcal{Z}^k, c \in \mathcal{C}, q \in \mathcal{Q}. \end{cases} \tag{5.3}$$

where $g_e(\cdot)$ is the transformed output function, $c_e(\cdot)$ is the corresponding controller formulation. The variable q is a placeholder variable for the variables of interest, as in (4.6) but expanded to take $\mathcal{E}$, instead of $\mathcal{Y}$, into consideration. Applying the error coordinate transformation (5.2) preserves the qualitative behaviour of the system dynamics. We refer to the transformed system as Ω_e and the transformed controller as Ξ_e. Adding further system requirements remains possible through $l(\cdot) \geq 0$ into (5.3), as described in Chapter 3.

The purpose of the substitution (5.2) is to directly express the error between r_{new} and the system output $y(k)$. Thus, the requirement (5.1) is transformed into:

$$y(k_{\text{end}}) \in \mathcal{Y}_{\text{new}}(r_{\text{new}}) \Longleftrightarrow e(k_{\text{end}}) \in \mathcal{B}_\epsilon := \prod_{i=1}^{n_y} [-\epsilon, \epsilon],$$

where $\mathcal{B}_\epsilon$ is the allowed precision range for the outputs at steady state.

5.2.2 Robust Set-point Regulation

We solve Problem 8 by splitting it into two stages. The first stage $\mathcal{T}_{\text{SPR}}$ is the set-point regulation, during which the outputs achieve the new references. The second stage $\mathcal{T}_{\text{RPI}}$ deals with keeping the system outputs close to the desired references. Dividing the task into two tasks allows for more flexibility and design freedom by assigning different controller parameterisations for each of the tasks [8]. Following the desired behaviour from Problem 8, we pose the first step, i.e. the set-point regulation requirement, as

$$y(k_{\text{new}}) \in [r_{\text{new}} - \epsilon, r_{\text{new}} + \epsilon] \text{ s.t. } \forall x(k_0) \in \mathcal{X}^{k_0}, \forall r_{\text{new}} \in \mathcal{R}_{\text{new}}, \tag{5.4}$$

where k_{new} is the last event of the set-point control horizon $\mathcal{T}_{\text{SPR}} := \{k_0, .., , k_{\text{new}}\}$, cf. Figure 5.1. As a result of applying (5.2) to Ω, the complexity of the control task in Problem 8 is reduced to steering the transformed system Ω_e to the steady-state error margin $\mathcal{B}_\epsilon$. Therefore, we re-formulate (5.4) into

$$\forall x(k_0) \in \mathcal{X}^{k_0}, e(k_{\text{new}}) \in \mathcal{B}_\epsilon. \tag{5.5}$$

In difference to Chapter 3, we neither want to estimate those values of the initial conditions that can reach the desired references, nor those references that can be reached from the initial conditions. Thus, we look for a controller parameterisation that respects all values inside the set of initial conditions and the set of references. Therefore, we use in the feasibility problem the inverted constraint formulation from Section 4.4. Accordingly, we need to introduce the following set:

$$\mathcal{B}_{\text{inv}} := (\mathcal{Y}_{k_{\text{end}}} \ominus \mathcal{R}) \setminus \mathcal{B}_{\epsilon}.$$

The set $\mathcal{B}_{\text{inv}}$ allows to use the inverted-constraint formulation to look for an inner approximation $\hat{\mathcal{C}}_{\text{SPR}}$ of the controller parameter values $\mathcal{C}_{\text{SPR}}$, i.e. $\hat{\mathcal{C}}_{\text{SPR}} \subseteq \mathcal{C}_{\text{SPR}}$. Therefore, any controller parameterisations $c^* \in \hat{\mathcal{C}}_{\text{SPR}}$ fulfil the for-all requirement for the initial conditions and the references. To express $\mathcal{C}_{\text{SPR}}$, we need to include (5.5) into the feasibility problem leading to a new feasibility problem EFP_{SPR}. Following the set construction principles for the for-all set, we end up with

$$\mathcal{C}_{\text{SPR}} := \text{proj}_{\mathcal{C}} \, \text{EFP}_{\text{SPR}}(\mathcal{B}_{\epsilon}) \setminus \text{proj}_{\mathcal{C}} \, \text{EFP}_{\text{SPR}}(\mathcal{B}_{\text{inv}}). \tag{5.6}$$

While $\mathcal{C}_{\text{SPR}}$ provides the theoretical set, in practice, we work with set approximations. Nevertheless, the statement (5.6) holds true also for an approximation, if we use the conservative SBE methods from Chapter 4. Thus, the controller parameterisations in $\hat{\mathcal{C}}_{\text{SPR}}$ provide the behaviour desired in (5.5). For more technical details in providing robust set-point regulation, we point the reader to [8].

5.2.3 Periodic Set Invariance

We now address the second control stage, i.e. keeping the system close to the reference. Classical designs use forms of (robustly controlled) invariant sets, as e.g. often used in model predictive control [47] or for hybrid systems [77], [93]. We consider a more general case, allowing for cyclic behaviour to be achieved after the set-point change. Standard set invariance formulation requires that once all states reach the invariant set, they need to stay inside infinitely long [32], which can be formulated as follows:

$$((x(k) \in \mathcal{X}^{k_0}) \Rightarrow (x(k^+) \in \mathcal{X}^{k_0})) \wedge ((y(k) \in \mathcal{Y}^{k_0}) \Rightarrow (y(k^+) \in \mathcal{Y}^{k_0})), k^+ \in \mathbb{N}_{\geq 0}. \tag{5.7}$$

In contrast to (5.7), we consider a formulation that allows the states, outputs, inputs to leave the initial sets and return to them at the end of a specified period, see Figure 5.2.

Such a behaviour is regarded as controlled periodic set invariance [114], [199]. To do so, we consider the following definition:

Problem 9. *(Robust controlled periodic positive set invariance requirement) Obtain a set $\hat{\mathcal{C}}_{RPI}$ of controller parameterisations that bring all states, outputs and inputs back to their corresponding initial sets after a given period k_{RPI}, i.e. $x(k_0), x(k_{RPI}) \in \mathcal{X}^{k_0}$;*

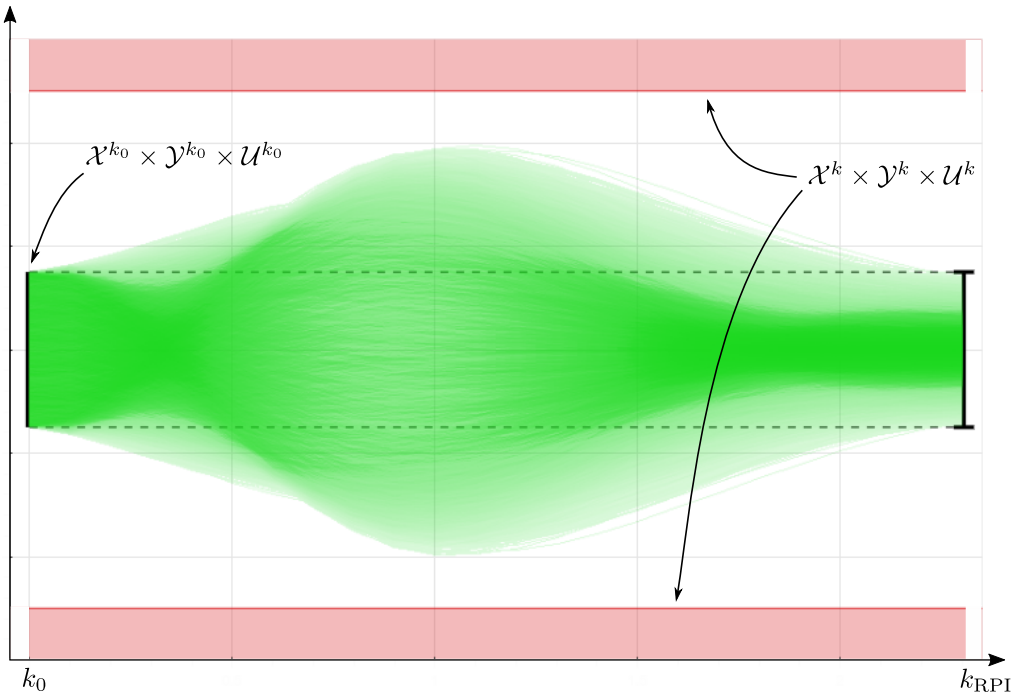


Figure 5.2: Desired periodic set-invariant behaviour. The system trajectories remain bounded at all instances and return at the end of the period to the initial set.

$y(k_0), y(k_{RPI}) \in \mathcal{Y}^{k_0}$; $u(k_0), u(k_{RPI}) \in \mathcal{U}^{k_0}$. *Additionally, the system needs to stay bounded between* k_0 *and* k_{RPI}, *i.e.* $x(k) \in \mathcal{X}^k$, $y(k) \in \mathcal{Y}^k$, $u(k) \in \mathcal{U}^k$, $k \in \mathcal{T} \backslash \{k_0, k_{RPI}\}$.

We refer to the requirement in Problem 9 as robust periodic invariance (RPI). Allowing the system to leave the desired set introduces flexibility.

In set-point regulation, we are looking for the for-all sets. To obtain the bounding sets of the system states and control inputs, in error formulation, we use the projection of the EFP_{SPR} onto the corresponding space. Moreover, to preserve the guarantees, we define $\mathcal{X}_\epsilon$ and $\mathcal{U}_\epsilon$ as conservative invariant sets of these projections:

$$\text{proj}_{x(k_{\text{RPI}})} \text{EFP}_{\text{SPR}}(\mathcal{B}_\epsilon) \subseteq \mathcal{X}_\epsilon,$$
$$\text{proj}_{u(k_{\text{RPI}})} \text{EFP}_{\text{SPR}}(\mathcal{B}_\epsilon) \subseteq \mathcal{U}_\epsilon.$$

The sets $\mathcal{X}_\epsilon$, $\mathcal{U}_\epsilon$, and $\mathcal{B}_\epsilon$ are used to obtain the outer approximation which serves as the initial region to look for the inner one of the for-all set. To describe the for-all set, we introduce the following three placeholder sets $\mathcal{E}_e$, $\mathcal{E}_u$ and $\mathcal{E}_x$ and formulate the following constraint that we use to modify the EFP:

$$e(k_0) \in \mathcal{B}_\epsilon \wedge x(k_0) \in \mathcal{X}_\epsilon \wedge u(k_0) \in \mathcal{U}_\epsilon \wedge e(k_{\text{RPI}}) \in \mathcal{E}_e \wedge x(k_{\text{RPI}}) \in \mathcal{E}_x \wedge u(k_{\text{RPI}}) \in \mathcal{E}_u,$$
$$\mathcal{T}_{\text{RPI}} = \{k_0, .., k_{\text{RPI}}\}. \tag{5.8}$$

Compared to (5.7), in (5.8) we added also the statement for the control input, as we achieve this behaviour through the influence of the controller Ξ_e. Analogously to

$\mathrm{EFP}_{\mathrm{SPR}}$, we add (5.8) to EFP and end up with the new feasibility problem $\mathrm{EFP}_{\mathrm{RPI}}$, which we use to obtain the controller parameterisations $\mathcal{C}_{\mathrm{RPI}}$ and provide the RPI requirement. To exclude the system trajectories that end outside the desired invariant sets at k_{RPI}, we define $\mathcal{X}_{\mathrm{inv}} := \mathcal{X}_{k_{\mathrm{RPI}}} \setminus \mathcal{X}_{\epsilon}$ and $\mathcal{U}_{\mathrm{inv}} := \mathcal{U}_{k_{\mathrm{RPI}}} \setminus \mathcal{U}_{\epsilon}$. In accordance to the approach for obtaining the for-all controller parameterisation, we pose the following relation:

$$\mathcal{C}_{\mathrm{RPI}} := \mathrm{proj}_{\mathcal{C}}\, \mathrm{EFP}_{\mathrm{RPI}}(\mathcal{X}_{\epsilon}, \mathcal{B}_{\epsilon}, \mathcal{U}_{\epsilon}) \setminus \mathrm{proj}_{\mathcal{C}}\, \mathrm{EFP}_{\mathrm{RPI}}(\mathcal{X}_{\mathrm{inv}}, \mathcal{B}_{\mathrm{inv}}, \mathcal{U}_{\mathrm{inv}})$$

The periodic set-invariance behaviour provides as design freedoms the variable bounds during the period and also the length of the period itself. If we set the invariance period to one step, i.e. $\mathcal{T} = \{k_0, k_1\}$, then the RPI is reduced to the classical one-step invariance, i.e. the signals are not allowed to leave the initial conditions. Therefore, the periodic invariance is a generalisation of the one-step invariance.

Analogously to the set-point regulation, we apply the same SBE methods to obtain an approximation $\hat{\mathcal{S}}_{\mathrm{RPI}}$ of $\mathcal{C}_{\mathrm{RPI}}$, i.e. $\hat{\mathcal{C}}_{\mathrm{RPI}} \subseteq \mathcal{C}_{\mathrm{RPI}}$. Thus, any value $c^* \in \hat{\mathcal{C}}_{\mathrm{RPI}}$ provides the robust periodic set invariant behaviour (5.8) and validates the controlled system behaviour for all times.

5.3 Illustrative Example

We consider a second-order system, describing e.g. the behaviour of shock absorbers in motorcycles [59] or a single joint of a robotic manipulator [204].

$$\begin{aligned} x_1(k^+) &= x_1(k) + \delta(k) x_2(k), \\ x_2(k^+) &= x_2(k) + \delta(k)(p_1 x_1(k) + p_2 x_2(k) + u(k)). \end{aligned}$$

Here $p_1, p_2 \in \mathbb{R}$ are the system parameters and $u(k)$ is the system input. The particular parameter values are $p_1 = -8$ and $p_2 = -4$. We consider that both states x_1 and x_2 are measured and are the system outputs of interest. The first state x_1 is required to achieve any reference in the range $[3, 4]$ with an error tolerance of $\epsilon = 0.15$. Where the range for x_2 is $[-0.15, 0.15]$, which practically translates that the system needs to achieve the new reference without much velocity. For this example, we consider a fixed sampling time of $\delta(k) = 0.1$, a final time of 2.5, i.e. $k_{\mathrm{SPR}} = 2.5$, and a periodic invariance of 2.3, i.e. $k_{\mathrm{PSI}} = 2.3$. To provide the desired behaviour, we used the following controller structure:

$$\begin{aligned} u_1(k^+) &= u_1(k) + \delta(k) e_1(k^+), \\ u(k) &= c_1 e_1(k) + c_2 u_1(k). \end{aligned}$$

Here u_1 is the controller state, and c_1, c_2 are the controller parameters. We apply the error-coordinate transformation (5.2) not only to the system dynamics, but also to the controller state. The offset coefficient for the controller state can be easily obtained

from simulation experiments. Therefore, the combined controlled system dynamics is expressed as

$$
\begin{aligned}
e_1(k^+) &= e_1(k) + \delta(k)e_2(k),\\
e_2(k^+) &= e_2(k) + \delta(k)(p_1e_1(k) + p_2e_2(k) - c_1e_1(k) + e_3(k)),\\
e_3(k^+) &= e_3(k) + \delta(k)c_2e_1(k^+).
\end{aligned}
$$

Here e_1, e_2, e_3 correspond to the error-coordinate formulations of x_1, x_2, u_1. The initial range of interest for the controller parameters is $c_1 = [0, 4], c_2 = [0, 16]$. We used the same ranges for the bisectioning procedure for the set-point regulation and the periodic set invariance tasks. The exact ranges for both tasks are in Table A.5 and A.6. The estimation results are presented in Figure 5.3.

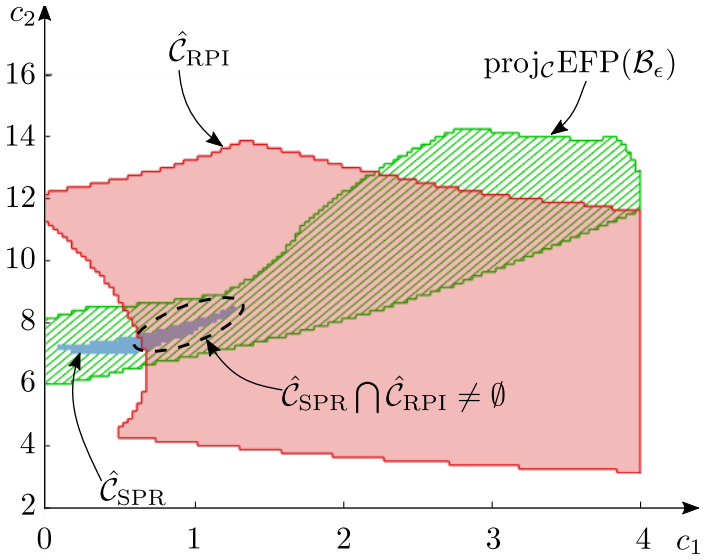


Figure 5.3: Estimated controller parameters for the set-point regulation and the periodic set-invariance tasks.

In this particular example, the estimated sets for both task overlap. As discussed before, this is not necessary. One could also tune two controllers (Ξ_{RPI} and Ξ_{SPR}) as it is often done for hybrid systems. In particular, the controller parameterisations that lie inside intersection of the blue and red sets provide both (5.5) and (5.8) and thus solve Problem 8.

To illustrate the estimation results, we performed a basic Monte-Carlo simulation by varying the initial conditions and references for controller parameterisations that were picked from inside and outside the estimated sets. The trajectories are presented in Figure 5.4. The blue trajectories are from controller parameterisations picked from inside the estimated set, where the red ones are from outside. As it can be seen from the system output and control signal trajectories, the conditions have been satisfied.

5.4 Discussion and Summary

We presented an approach to achieve robust error-free steady-state control under endlessness requirement for hybrid dynamical systems. The approach transforms the system dynamics into the error-coordinate formulation to deal with the for-all requirement of the references. We split the control task into set-point regulation and robust periodic invariance afterwards. Therefore, we intersect the two estimated sets, i.e.

$$\hat{\mathcal{C}} = \hat{\mathcal{C}}_{\mathrm{RPI}} \bigcap \hat{\mathcal{C}}_{\mathrm{SPR}}.$$

Nevertheless, if $\hat{\mathcal{C}} = \emptyset$ but still $\hat{\mathcal{C}}_{\mathrm{SPR}} \neq \emptyset \wedge \hat{\mathcal{C}}_{\mathrm{RPI}} \neq \emptyset$, we can use a switching control strategy. During the first stage $\mathcal{T}_{\mathrm{SPR}}$, we can pick one parameterisation $c_1^* \in \hat{\mathcal{C}}_{\mathrm{SPR}}$, and apply another one $c_2^* \in \hat{\mathcal{C}}_{\mathrm{RPI}}$ after $\mathcal{T}_{\mathrm{SPR}}$. Thus, by switching from c_1^* to c_2^*, we provide the desired behaviour. An additional benefit of working with the SBE from Chapter 4 is that there is always a stability statement for free. It is a form of a 'bounded-input bounded-state bounded-output' type of stability statement [207], i.e. all system elements are kept bounded. This stability statement is the results of the employed set-based approach, as successful estimation results imply that the system elements stay inside the desired bounds.

After the in-depth look at a particular advanced requirement on the process control level, we continue in Chapter 6 with the validation and monitoring of production systems with structural similarities.

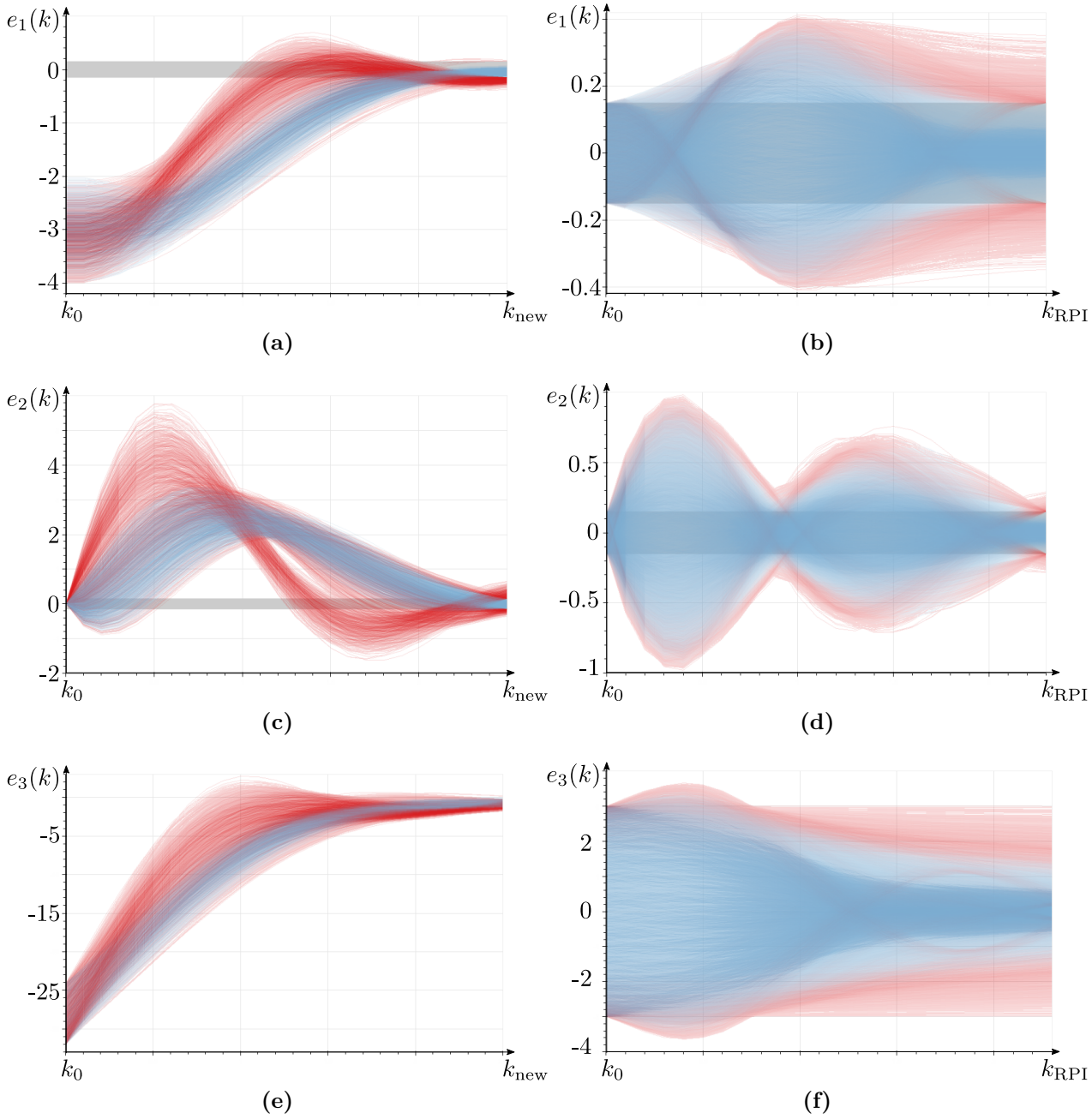


Figure 5.4: Monte Carlo runs for different initial conditions and references. The grey horizontal stripes in (a), (b), (c), (d) depict the sets of references at the end of the horizon. The blue trajectories are computed with controller parameterisation from inside the estimated sets, and the red from outside them. Figures (a), (c), (e) are the runs for the set-point regulation, and (b), (d) and (f) are for the periodic set invariance. Figures (a) and (b) are for e_1 state, (c) and (d) are for the e_2 state and (e) and (f) are the controller state e_3.

6 Validation of Production Systems with Structural Similarities

> You know, I am sorry for the poor fellows that haven't got labs to work in.
>
> Sir Ernest Rutherford

In this chapter, we present an approach for validation of production systems with numerous elements from a small set of basic elements. We abstract the basic elements and develop a framework that combines these abstractions, depending on the task at hand.

As an example of the system class, we consider transportation networks in discrete manufacturing. In such systems, the sensors produce binary signals and generate events only when a change occurs, e.g. an item is detected at defined places. We take into account the event-driven operation considering the uncertainties from measurements or system parameters. We conclude with illustrative simulation scenarios. Moreover, we provide a summary of the execution times from a manufacturing test plant. Some of the results in this chapter are based on [10].

6.1 Task Description

To understand the challenges when working with transportation systems, we outline the defining characteristics and the challenges that originate from them. We present a manufacturing test plant that serves as motivation for the developed approach and later for the test cases. We conclude the section with the problem formulation.

6.1.1 System Properties and Challenges

Many manufacturing systems are characterised by production stages with numerous process machines at different stations. We consider systems, which might change during operation by adding or removing process stations. We are interested in the case where, during operation, machines are added or removed from the system. We model these systems by small modelling units called plant elements.

In manufacturing, the processing goods pass through or are being processed by these elements. This movement across the plant describes a transportation network throughout the plant. Two examples for such a transfer are the positioning of an

item from a milling machine to a polishing one or the transport between a processing station to storage.

We consider systems that operate event-based. Furthermore, the measurement signals that we obtain are in binary format. Boolean data is standard for many manufacturing systems, as it is sufficient to signalise the completion of a task, the arrival of an item, or signalling a failure.

Moreover, the system might contain parametric and temporal uncertainties. The parametric ones are due to the initial imprecise placement of the sensors or mechanical shift or drift of their positions. The temporal ones express the uncertainty of the arrival time of the measurement in the monitoring system. One challenge is that the validation has to be performed independently of the control system. Often one does not have access to the control logic but only to the plant measurement information.

Furthermore, the validation needs to be done in real-time. Clearly, the real-time restriction is dependent on the particular system. For discrete manufacturing systems, this translates to time requirements in the tens of milliseconds. We consider constraint-based tailored models and verifying them using the presented set-based estimation methods in Chapter 4. We continue with a brief overview of the test plant that contains all the characteristics and challenges mentioned earlier.

6.1.2 Motivation

We consider a discrete-manufacturing plant consisting of different stations connected via transportation network, c.f. Figure 6.1. This plant exhibits the outlined challenges: dynamically adjustable layout, temporal and spatial uncertainties, event-driven operation.

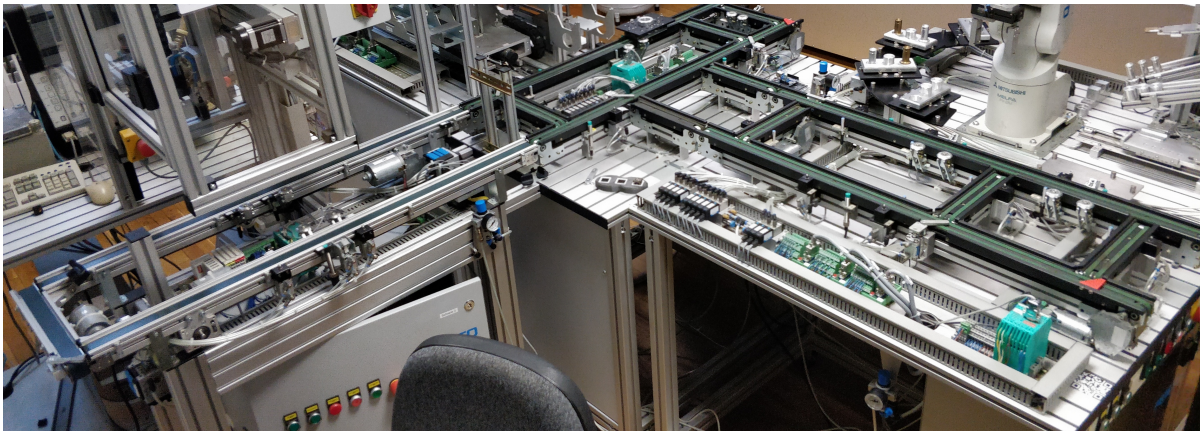


Figure 6.1: Example manufacturing test plant at the Institute for Automation in the University of Magdeburg.

We can identify the following four elementary elements: conveyor belt, proximity sensors, actuated gates, and junctions. The conveyor belts transport the items along the network. The proximity sensors detect the item's close presence, e.g. by laser

gates, hall sensors, induction elements, etc. The actuated gates obstruct the further motion of the item along the network. In the basic case, they are mechanical obstacles that stop an item. In the general case, they can be a complex process station that can be represented as an actuated gate, as it 'blocks' the movement of the processed items. Furthermore, there are junctions, which can be controlled. So-called always opened junctions mix incoming goods' flows, while controlled junctions set the consecutive motion direction. Using these four elements, we can abstract large systems, e.g. as shown in Figure 6.2.

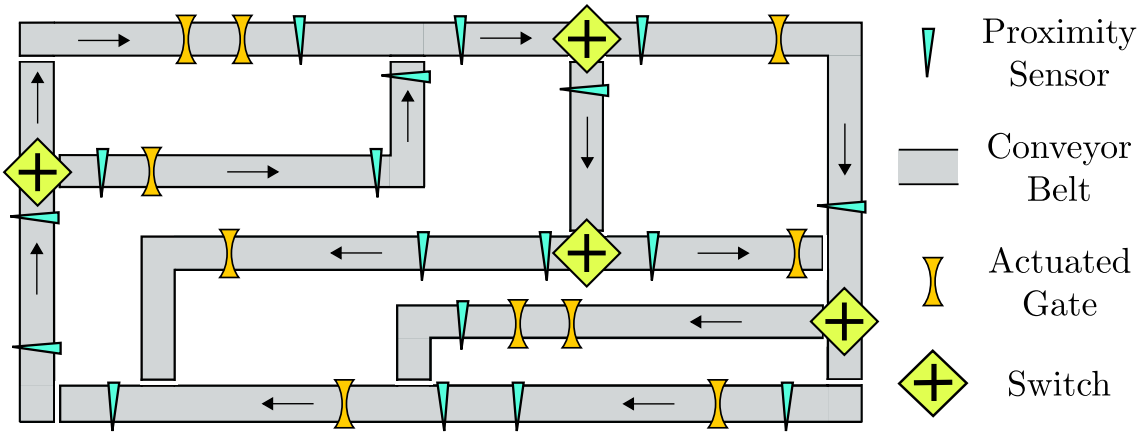


Figure 6.2: An example for the schematic layout of a manufacturing plant.

The produced items are discrete entities and are sequentially transported. The measurement information about the items we have is whether a proximity sensor detects the presence of an item or not and whether an actuated gate is deployed or not. In general, we cannot measure whether an item is actually in front of an actuated gate. Note that the movement of items is influenced by corners, friction, belt transitions, which in an abstract sense, introduces spatial uncertainties. These spatial uncertainties and the lack of information of the controlled junctions can lead to ambiguity of uncertain items' positions. Another source of uncertainty to be taken into account is the speed of the conveyor belts. Belts can operate at different speeds, e.g. due to different power units or varying load ratios, and, therefore, cannot be assumed to be constant throughout the plant and for all times.

Note that, although the plant description might appear to be very specific, several industrially relevant systems fit into this system class, see Table 6.1. The approach allows to handle systems that exhibit behaviour that can be abstracted through those few structural elements. A common characteristic between all of them is a discrete entity, e.g. goods, cars, suitcases, livestock, that passes through different processing stations that for the validation purpose of the global behaviour can be considered as obstacles at which an item is stopped.

Table 6.1: Example systems exhibiting similar behavior to the described principles of operation

System	Position sensors	Obstacle	Transported items
Discrete manufacturing	proximity sensors	actuated gates	goods
Highway traffic	traffic counters	incidents	cars
Logistic centers	proximity sensors	stoppers	stored items
Baggage handling system	photoelectric sensors	separators	suitcases
Livestock handling	photoelectric sensors	gates	livestock

6.1.3 Problem Formulation

We are interested in monitoring and validating the correct operation of manufacturing systems despite the uncertainties. Monitoring consists of validating the correct operation. We are interested not only in the events that have occurred but also in those that should have. That includes not only the current situation but also the previous execution of the processes. Detecting missing events is needed for the diagnosis of broken or missing equipment. To perform the monitoring and fault diagnosis, we rely not only on state but also on parameter estimation. Parameter estimation is used to validate a characteristic of a particular element, e.g. to check if the conveyor belt is not moving with the required speed or if a sensor is dislocated.

Another critical task of monitoring includes predictive maintenance by detecting upcoming abnormal behaviour. We summarise these goals in the following problem:

Problem 10. *Describe and validate the operation of event-driven systems with structural similarities through a flexible validation approach that is capable of*

- *state and parameter estimation,*
- *fault diagnosis,*
- *abnormal behaviour prediction.*

The estimation procedure using these models should be solved in real-time and provide guaranteed results, accounting for unknown-but-bounded uncertainties in states and parameters.

6.2 System Abstraction

We tackle Problem 10 by segmenting the plant into sub-plant parts. In addition, we combine these sub-parts into a validation entity that can be of different size and

structure depending on the task at hand.

Our abstraction and segmentation approach is motivated by the three dominant challenges: the scale of the system, the event-driven operation and the present uncertainties. To handle the challenge arising from the problem scale, we decompose it into basic elements. We exploit the event-driven operation due to the nature of the problem. To cope with the uncertainties, we use the presented SBE methods.

To do so, we first decompose the plant into plant elements, i.e. proximity sensors, actuated gates, conveyor belts. For each element, a simple first principle model is then derived. Models of different elements are combined with the plant events into so-called modules. The events are derived from the measurement data obtained from the plant by an event generator. It is important to note that this procedure remains valid when a new section is added to the plant during operation. We now present the mathematical formulation for each plant element and how to construct modules of treatable size.

Basic Elements

We decompose the system based on the transport layout and the basic elements. We focus on the following elements: conveyor belt, proximity sensor (PS), and actuated gate (AG). Afterwards, we deal with the junctions by combining the three elements. To aid the discussion of the constraint derivation for SBE, we focus on the small segment of the plant, depicted in Figure 6.3.

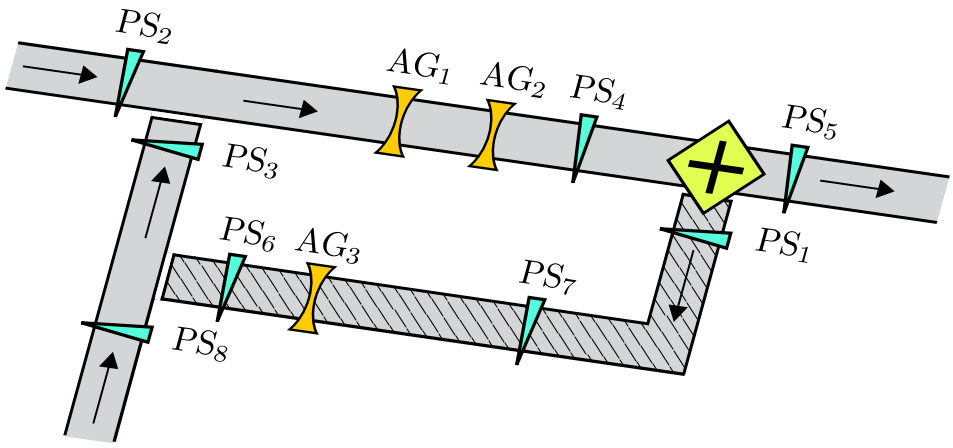


Figure 6.3: A plant segment composed of basic plant elements: proximity sensors, actuated gates, conveyor belt and junctions. The position of an item is determined relative to the elements. The hatched zone illustrates the segmentation into serial sequences of elements between two junctions.

Every basic element in Figure 6.3 interacts with the processed items in a specific way. In particular, either an element influences the item's movement, or it provides information on the item position. In the following, we present these interactions for

each plant element with the items. We refer to the resulting mathematical descriptions as constraint blocks. Due to the event-driven nature of the system, we present constraints blocks that relate to the change of the plant state between two consecutive events.

Unobstructed item movement

We start with the basic element that connects the processing stations and transports the goods - the conveyor belt. We can describe the position of an item moving along a conveyor belt as:

$$x(k^+) = x(k) + \delta(k)(p - d(k)). \tag{6.1}$$

Here $x \in \mathbb{R}_{\geq 0}$ is the item's positions, $p \in \mathbb{R}_{\geq 0}$ is the belt velocity, and $\delta(k)$ is time period between the events k^+ and k. Reverse movement of the belt is not allowed, and, if needed, it is practically realised by adding a parallel track connecting the section between two junctions. Therefore, we assume $p > \max|d(k)|$ to ensure the forward movement along the belt, where $d(k) \in \mathbb{R}^k$ are disturbances, e.g. due to friction. As the item position cannot be measured continuously, we use proximity sensors to obtain information about the items.

Proximity sensor constraints

Each proximity sensor (PS) provides binary information of whether an item is present or not. For each PS, we introduce two auxiliary event-based binary states. The first variable $z_{\text{PS}}^{\text{in}}$ corresponds to the event of a item entering the field at time $k_{\text{PS}}^{\text{in}}$ and the second one $z_{\text{PS}}^{\text{out}}$ to the event of the item exiting at time $k_{\text{PS}}^{\text{out}}$, i.e.

$$z_{\text{PS}}^{\text{in}}(k) = \begin{cases} 0, & k < k_{\text{PS}}\text{i}, \\ 1, & \text{otherwise}, \end{cases}$$
$$z_{\text{PS}}^{\text{out}}(k) = \begin{cases} 0, & k < k_{\text{PS}}^{\text{out}}, \\ 1, & \text{otherwise}. \end{cases}$$

With the help of $z_{\text{PS}}^{\text{in}}(k)$, $z_{\text{PS}}^{\text{out}}(k)$, and the proximity sensor's position $p_{\text{PS}} \in [\underline{p}_{\text{PS}}, \overline{p}_{\text{PS}}]$, we can derive the following bounds on the position of an item:

$$\begin{aligned} x(k) &\leq (1 - z_{\text{PS}}^{\text{in}}(k))p_{\text{PS}} + z_{\text{PS}}^{\text{in}}(k)\,\mathcal{M}, \\ x(k) &\geq (p_{\text{PS}} + p_{\text{size}})z_{\text{PS}}^{\text{out}}(k). \end{aligned} \tag{6.2}$$

Here $p_{\text{size}} \in \mathbb{R}^+$ is the parameter denoting the item size, and $\mathcal{M} \in \mathbb{R}^+$ is an auxiliary parameter that is greater than the length of the considered conveyor belt segment. The first constraint in (6.2) bounds the item position from above, i.e. before the item passes the sensor. Whereas the second constraint bounds it from below, i.e. after the item has passed the sensor.

Actuated gate constraints

Unlike the proximity sensors, the actuated gates (AG) can influence the item's dynamics by halting it and thus setting its position equal to the position of the deployed gate.

The AG elements do not possess the ability to detect whether an item is obstructed. They provide only information on whether and when the gates are being deployed or removed from the belt. In general, if the belt segment contains an element AG, placed at position $p_{AG} \in [\underline{p}_{\mathrm{AG}}, \overline{p}_{\mathrm{AG}}]$, the dynamics can be expressed as follows:

$$x(k^+) = \begin{cases} p_{\mathrm{AG}}, \text{ if an item is stopped in front of the AG,} \\ x(k) + \delta(k)(p - d(k)), \text{ otherwise.} \end{cases}$$

The case of being obstructed by the element AG is realised if the following two conditions are met. First, the gate must be deployed, and second, the item would have not yet passed the gate's position p_{AG} by the time k^+. To model this behavior we introduce an event-based plant state z_{AG}, connecting it to the state of the element AG as follows:

$$z_{\mathrm{AG}}(k) = \begin{cases} 0, & \text{AG is deployed and } x(k) \leq p_{\mathrm{AG}}, \\ 1, & \text{otherwise.} \end{cases}$$

Here $\mathcal{T}_{\mathrm{AG}} := \{k^j_{\mathrm{AG}}\}, j \in \{1, .., n_j\}$ denotes the time points when the state of the element AG changes between deployed and removed, and n_j are the number of changes. In other words, if the gate is deployed in front of the item at k^j_{AG}, then $z_{\mathrm{AG}}(k)$ is set to 0, otherwise to 1. The value of z_{AG} is constant between two consecutive events. Otherwise, it results in an additional event.

Since the value of z_{AG} only indicates that the item's movement might be obstructed, an additional virtual state $\mu(k)$ is introduced. We assign $\mu(k)$ a value 1 if and only if the item can reach the deployed gate by the time k^+ using the equivalence relation $\Leftrightarrow$. In this way, we can reformulate the unobstructed dynamics constraint (6.1) as follows:

$$\begin{aligned} &x(k^+) = (x(k) + \delta(k)(p - d(k)))(1 - \mu(k)) + p_{\mathrm{AG}}\,\mu(k), \\ &\mu(k) = 1 \Leftrightarrow x(k) + \delta(k)(p - d(k)) \geq p_{\mathrm{AG}}(1 - z_{\mathrm{AG}}) + \mathcal{M}\, z_{\mathrm{AG}}. \end{aligned} \tag{6.3}$$

Using the presented constraints (6.1), (6.2) and (6.3) independently for each element of the conveyor belt do not provide enough information to successfully solve all tasks in Problem 10. We demonstrate next how these constraint blocks can be suitably combined to do so.

6.3 Modules

To tackle the monitoring task described in Problem 10, we present a validation entity. Afterwards, we explain how it can be formulated in the set-based framework that we use in this work. At the end of the section, we discuss how to deal with junctions.

Definition and Formulation of A Module

To validate the correct system operation and detect anomalies, we rely upon the measured information from the plant in its basic form. In the previous section, we pointed out that only the neighbouring elements around an item influence or provide information about its position. Moreover, the propagation of uncertainties requires considering a time horizon as small as possible to reflect the desire for obtaining the results in relevant time. Still, the model should include a sufficient amount of constraints to validate it or narrow down the estimates. We propose the following definition of a local sub-plant model, satisfying these requirements.

Definition 6 (Module)**.** *A module consists of a sequence of plant elements. Each module starts and ends with a proximity sensor and can have actuated gates between them.*

Structurally, a module is a sequence of PS and AG elements, where each PS serves as the last element of the previous module and the first element in the next module. The construction of a module contains the following three steps. First, the constraint block for each element is selected. Second, each constraint block is parameterised by selecting the corresponding signals and determining the values of the introduced event-based variables. Lastly, the relevant events are organised in the time horizon for the specific module. To illustrate this concept, we present next, the simplest module - PS-PS (e. g. PS_1-PS_7 in Figure 6.3). This module consists of the dynamics constraint block (6.1) and a constraint block (6.2) for each PS. Therefore, the module requires four events: the item entering and exiting the field of each PS. The corresponding events for this module are organised in

$$\mathcal{T} := \{k^{\mathrm{in}}_{\mathrm{PS}_1}, k^{\mathrm{out}}_{\mathrm{PS}_1}, k^{\mathrm{in}}_{\mathrm{PS}_7}, k^{\mathrm{out}}_{\mathrm{PS}_7}\}.$$

Another simple module is the PS-AG-PS, which includes one actuated gate between two consecutive sensors (e. g. PS_7-AG_4-PS_6 in Figure 6.3). Such a module consists of one constraint block (6.2) for each PS and the modified dynamics (6.3). The amount of considered time steps depends on the number of events, triggered by the actuated gate between the time points $k^{in}_{\mathrm{PS}_7}$ and $k^{out}_{\mathrm{PS}_6}$, and thus equal to $|\mathcal{T}_{\mathrm{AG}_4}| + 4$. The time horizon is defined as follows

$$\mathcal{T} := \mathrm{ord}\left(\{k^{\mathrm{in}}_{\mathrm{PS}_7}, k^{\mathrm{out}}_{\mathrm{PS}_7}, k^{\mathrm{in}}_{\mathrm{PS}_6}, k^{\mathrm{out}}_{\mathrm{PS}_6}\} \cup \mathcal{T}_{\mathrm{AG}_4}\right),$$

where the function ord denotes the ordering of the time points, so that $k_i \leq k_j$ for all $k_i, k_j \in \mathcal{T}$ if $i < j$.

More complex modules consist of multiple actuated gates AG_i, situated between two consecutive proximity sensors (e. g. PS_3-AG_1-AG_2-PS_4 in Figure 6.3). This case requires further adjustment of the dynamics constraints (6.3). As multiple obstacles can be deployed at the same time, an item can only be stopped by one of them. One way to model such a scenario is to reformulate the definition of the event-based variable $z_{\mathrm{AG}}(k)$ as follows

$$\tilde{z}_{\mathrm{AG}}(k) = \begin{cases} 0, & \exists\, i \in \mathcal{I} \text{ s.t. } \mathrm{AG}_i \text{ is deployed and } x(k) \leq p_{\mathrm{AG}_i}, \\ 1, & \text{otherwise,} \end{cases}$$

with an additional variable ν, denoting the position of the first deployed AG element

$$p_{\mathrm{d}} = \min_{i \in \mathcal{I}} \left\{ p_{\mathrm{AG}_i} \mid \tilde{z}_{\mathrm{AG}}(k) = 1 \text{ and } p \leq p_{\mathrm{AG}_i} \right\}. \tag{6.4}$$

To implement (6.4) into the given set-up, we need to re-formulate it as mixed-integer linear constraints. Therefore, we introduce the following binary variables $z_{\mathrm{DAG}_1}, ...,$ z_{DAG_2}, $z_{\mathcal{M}}$ and express (6.4) as

$$\begin{aligned} & z_{\mathrm{DAG}_i} \leq z_{\mathrm{AG}_i}, i \in \{1, ..., n_{\mathrm{AG}}\}, \\ & \Sigma_{i=1}^{n} z_{\mathrm{DAG}_i} + z_{\mathcal{M}} = 1, \\ & p_{\mathrm{DAG}} \leq z_{\mathrm{AG}_i} \overline{p}_{\mathrm{AG}_i} + (1 - z_{\mathrm{AG}_i}) \mathcal{M}, i \in \{1, ..., n_{\mathrm{AG}}\}, \\ & p_{\mathrm{DAG}} \geq \Sigma_{i=1}^{n} z_{\mathrm{DAG}_i} + z_{\mathcal{M}} \mathcal{M}, i \in \{1, ..., n_{\mathrm{AG}}\}, \end{aligned} \tag{6.5}$$

where n_{AG} is the number of gates between the two sensors. Then the item dynamics for multiple gates can be formulated as

$$\begin{aligned} & x(k^+) = (x + \delta(k)(p - d(k)))(1 - \mu(k)) + \mu(k) p_{\mathrm{DAG}}, \\ & \mu(k) = 1 \Leftrightarrow x + \delta(k)(p - d(k)) \geq p_{\mathrm{DAG}} \end{aligned} \tag{6.6}$$

The time horizon in this scenario needs to include the time points corresponding to the events triggered by each AG_i, situated within the module. We denote the horizon containing the events of the actuated gates as

$$\tilde{\mathcal{T}}_{\mathrm{AG}} := \bigcup_{i \in \mathcal{I}} \mathcal{T}_{\mathrm{AG}_i}, \ \mathcal{T} := \mathrm{ord}\left(\{ k^{\mathrm{in}}_{\mathrm{PS}_3}, k^{\mathrm{out}}_{\mathrm{PS}_3}, k^{\mathrm{in}}_{\mathrm{PS}_4}, k^{\mathrm{out}}_{\mathrm{PS}_4} \} \cup \tilde{\mathcal{T}}_{\mathrm{AG}_i} \right).$$

Constructing the horizon for multiple gates concludes the discussion for selecting the correct constraint block. The next step is summarising this information in the set-based framework of this work.

Feasibility problem formulation

Once the constraints blocks for the basic elements are present, constructing a module is reduced to selecting the corresponding constraints and extracting the matching events. The constructed module does not need any additional re-formulation and can be directly posed as a module feasibility problem:

$$\text{MFP}: \begin{cases} \text{constraint blocks (6.1), (6.2), (6.3), and (6.6),} \\ \text{relevant events: } k \in \mathcal{T}, \\ \text{disturbances: } d(k) \in \mathcal{D}^k, \\ \text{dynamics' states (i.e item's position): } x(k) \in \mathcal{X}^k, \\ \text{parameters: } p \in \mathcal{P}, p_{\text{PS}} \in \mathcal{P}^{\text{PS}}, p_{\text{AG}} \in \mathcal{P}^{\text{AG}}, \nu \in \mathbb{R}_{>0}, \mathcal{M} \in \mathbb{R}_{>0}, \\ \text{auxiliary variables: } z_{\text{PS}}(k) \in \{0,1\}^k, z_{\text{AG}}(k) \in \{0,1\}^k, \mu(k) \in \{0,1\}^k. \end{cases}$$

Solving MFP allows obtaining certified results about the operation of the plant. On the one side, by invalidating the problem, the procedure can inform about the occurrence of a system failure. On the other side, by performing state and parameter estimation, the current performance of the plant can be monitored. Through the introduced modules, we can tackle the last element - junction. In short, we deal with junctions by constructing competing modules and analyse the estimation outcomes to verify which path has been taken.

Controlled Junctions

The purpose of controlled junctions is to redirect the movement of an item to one of several possible paths, as depicted in Figure 6.4. Instead of modelling them directly, we consider the possible modules that include the junction, see Figure 6.4.

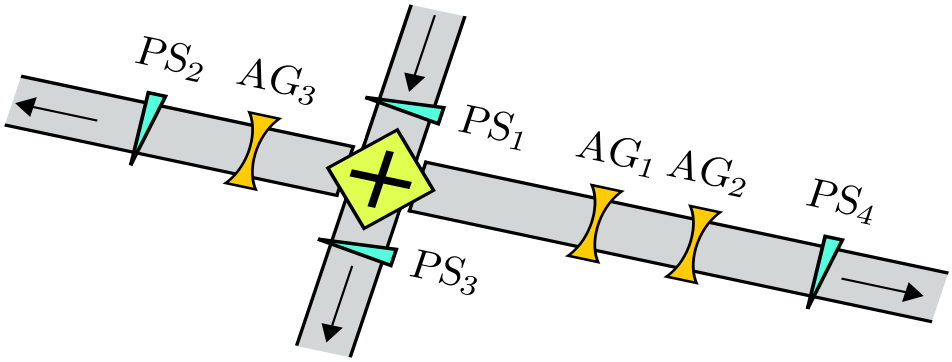


Figure 6.4: A junction, which splits the infeed into three outfeed directions.

If a controlled junction remains unchanged long enough for the complete uncertain position to traverse it, then the taken path is known. In this case, we construct the corresponding module directly and verify the operation. Nevertheless, if the state of the junction is unknown, we can construct multiple competing modules based on

every possible path taken at the junction. For instance, in Figure 6.4 to encompass all possibilities at the junction, we generate three modules. The modules are PS_1-PS_3, PS_1-OB_1-OB_2-PS_4 and PS_1-OB_3-PS_2. In addition, we need the events that belong to the elements for each module. Determining which path was taken by the item is then simply done by checking the feasibility of every module. To demonstrate the capabilities and the quality of the results of the modules, we present one example for each of the main problem scenarios.

6.4 Implementation and Results

We present several test cases that illustrate the capability of the developed approach. Moreover, we outline the implementation workflow that provides the autonomous operation of the monitoring system. In addition to the simulation results, we provide a brief overview of implementation results from a test plant.

6.4.1 Real-time Workflow

Figure 6.5 shows a possible system architecture that enables the real-time implementation of the proposed abstraction through modules on a test plant.

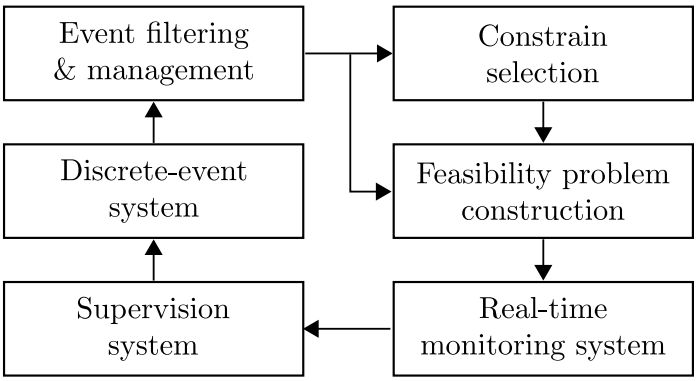


Figure 6.5: Scheme of the proposed implementation workflow.

The *discrete-event system* operates and generates measurement events. These are processed by the *Event filtering & management* that collects and organises them. Based on the measured events, the *Constraint selection* procedure is started. The module is constructed inside the *Feasibility problem construction* by combining the suitable constraint blocks and events. Then the module is passed to the *Monitoring system* that solves and analyses the validation results and informs the *Supervision system*. Which, in turn, based on the current plant-wide operation status and the monitoring information, decides how to influence the discrete-event system. Moreover, the supervision system can initiate module construction on request. This ability is useful in those cases when a sensor malfunction is suspected.

6.4.2 Experimental Results

We demonstrate the presented approach with several simulation scenarios: state and parameter estimation for predictive maintenance, prediction of abnormal behaviour, and a segment with multiple speeds. All examples are tackled using the presented abstraction approach through set-based formulation. We summarise the execution times from a test manufacturing plant. All results are achieved using MATLAB and ADMIT [210] to pose the problems, and Gurobi [80] as the solver.

Speed monitoring

In this example, the monitoring task consists of comparing the estimated speed of a conveyor belt to the nominal one. The speed of the belt is the parameter of interest. Therefore, the monitoring system compares the estimation results to the nominal operation and decides whether an abnormal behaviour has occurred.

For the estimation, we employ the simplest module, i.e. PS-PS, which in turns corresponds to combining (6.1) and (6.2) as explained in Section 6.3. In the presented example, the positions of the sensors were known with a precision of ± 2 mm. Table A.7 contains the initial values for the estimation. The plant events are $\mathcal{T} = [230.341, 230.469, 234.164, 234.299]$ s. The result of the parameter estimation procedure is

$$p = [22.83, 23.18] \text{ cm/s}$$

which lies inside the nominal range of $[22.5, 23.5]$ cm/s and thus the system is operating normally.

Alternatively to the monitoring task, the same parameter estimation procedure can be used at the initialisation phase of a plant to establish the positions of the plant elements and belt speeds. Another use case for this scenario is after concluding repair works or after a significant change in the operating conditions.

Position monitoring

One of the essential monitoring tasks is the validation of the item's position, i.e. state estimation. Based on this information, different behaviours can be identified. If the estimation procedure is invalidated, this indicates a deviation from normal behaviour, and thus a fault is detected. Here, we would like to demonstrate the quality of the estimation results, and hence we show the results from a normal run.

We consider a PS-AG-AG-PS module with parameters given in Table A.9. The initial estimation bounds on the item position x are considered unknown, i.e. they are much wider than the actual sensors' positions. The selected events are $\mathcal{T} = [0, 0.401, 0.487, 2.102, 3.803, 4.204, 4.291]$ s. We have included an additional initial event $k_0 = 0$ for illustration purposes, and it does not influence the quality of the results. The module is constructed using the approach in Section 6.3.

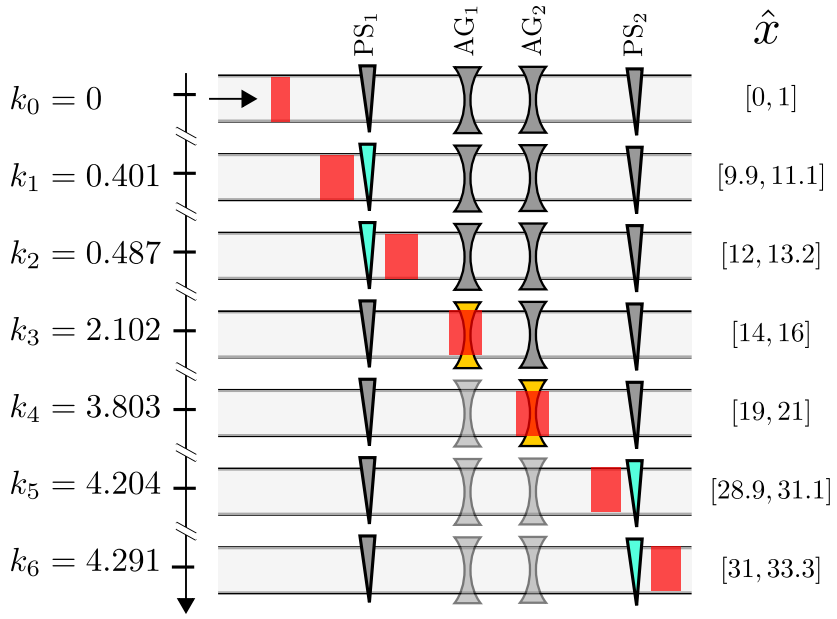


Figure 6.6: The estimated item's position $\tilde{x}$, on the right, for a PS_1-AG_1-AG_2-PS_2 module. The corresponding event instances are on the left, and the grey horizontal strips is the conveyor belt.

The broad grey horizontal stripes in Figure 6.6 depict the belt, and the arrow is showing the direction of movement. The area in red is symbolising the uncertainty range of the item's position. The exact ranges are given on the right, and the corresponding events are on the left. Additionally, only the elements that are responsible for the current estimation are in colour, and the inactive elements are in grey. Improving the estimation precision can be obtained by starting with more precise information about the four element's positions, i.e. narrower uncertainty ranges of p_{PS_1}, p_{PS_2}, p_{AG_1}, p_{AG_2}. Furthermore, measuring p more precisely or reducing the system uncertainties d can lead to further improvements.

Traffic jam detection and prediction

A well-designed industrial system should not only react to current problematic situations but be capable also of anticipating such situations. Therefore, we discuss two examples: detecting and predicting a production related traffic jam. We define a traffic jam as a zone with a defined maximum capacity $p_{\max}$ and once the threshold $p_{\max}$ is reached, a jam occurs. Furthermore, validating that it will occur ahead of time provides predictive capabilities of the monitoring system. To solve these tasks, we use the information of the last proximity sensor before the beginning of the zone. We propagate the item motion further and estimate the time at which the item enters the jam zone.

We take the time k_{PS}^{in} the item enters and time k_{PS}^{out} it leaves the proximity sensor. To the nominal speed of the belt p, we add a term $d(k)$ to encompass the disturbances

and noises. Based on this information and the relative position p_{zone} of the zone to the position of the proximity sensor, we can estimate the virtual event k_{enter} at which the item enters the zone. The parameters for the simulation are given in Table A.8 and the estimation result is

$$k_{\text{enter}} = [1.596, 1.627] \text{ s}.$$

The traffic jam detection consists of the following two conditions that are connected in logical conjunction:

$$k_{\text{enter}} < k_{\text{now}} \quad \wedge \quad p_{\max} \leq p_{\text{current}}(k) + 1, \tag{6.7}$$

where k_{now} is the time at which the check is executed, and $p_{\text{current}}(k)$ is the number of items at the time k. Thus, when both conditions in (6.7) are satisfied, then there is an active traffic jam.

Multiple belts

In this example, we consider the movement of an item along three consecutive conveyors. Each of them has a different speed. We are interested in whether only two proximity sensors are sufficient to establish the correct speeds of the belts. The first sensor is on the first belt, and the second sensor is placed on the third belt. We compare the two-sensor estimation results against a four-sensor scenario. In the latter scenario, the first sensor is on the first belt, the second and third sensors are at the beginning of the second and third belts, and the fourth sensor is further along on the third belt, cf. Figure 6.7 In both scenarios, i.e. with two and four sensors, the positions of the sensors are known. However, the item's position and the velocities of the belts are unknown and thus specified with wide uncertainty bounds. The values used in simulating both scenarios are presented in Table A.10. For the two-sensor scenario, the estimation procedure has access only to the time events of the first and fourth proximity sensor, i.e.

$$\mathcal{T}_2 = \{k^{\text{in}}_{\text{PS}_1}, k^{\text{out}}_{\text{PS}_1}, k^{\text{in}}_{\text{PS}_4}, k^{\text{out}}_{\text{PS}_4}\}$$

Accordingly, the four-sensor scenario considers additionally the events from the second and the third sensor, i.e.

$$\mathcal{T}_4 = \mathcal{T}_2 \bigcup \{k^{\text{in}}_{\text{PS}_2}, k^{\text{out}}_{\text{PS}_2}, k^{\text{in}}_{\text{PS}_3}, k^{\text{out}}_{\text{PS}_3}\}$$

Table 6.2 compares the estimation results for the belt speeds, i.e. p_1, p_2 and p_3, based either on the information from 2 or 4 proximity sensors. The exact event times are presented on the left in Figure 6.7. Furthermore, we can perform an estimation of the transition times, e.g. of the time k_{12} item leaves the first belt and enters the second and respectively the time k_{23} the item leaving the second and entering the third belt.

Table 6.2: Estimated transition times and belt velocities for the multi-belt scenario

	2 PS	4 PS	Unit
k_{12}	$[1.331, 1.626]$	$[1.593, 1.608]$	s
k_{23}	$[3.317, 3.529]$	$[3.424, 3.434]$	s
p_1	$[24.75, 30]$	$[24.7943, 25.2929]$	cm s^{-1}
p_2	$[20, 28.3648]$	$[21.7344, 22.1121]$	cm s^{-1}
p_3	$[27.5, 30]$	$[27.824, 28.1836]$	cm s^{-1}

Comparing the velocity estimates, we conclude that the precision obtained from only two sensors is not sufficient. Therefore, the results justify placing four proximity sensors for providing tighter estimates, and thus better monitoring. Moreover, the four-sensor results are contained inside the two-sensor estimates. This inclusion speaks about the correctness and conservatism of the approach. As an additional outcome, we get the item's position estimate at each instance, see Figure 6.7.

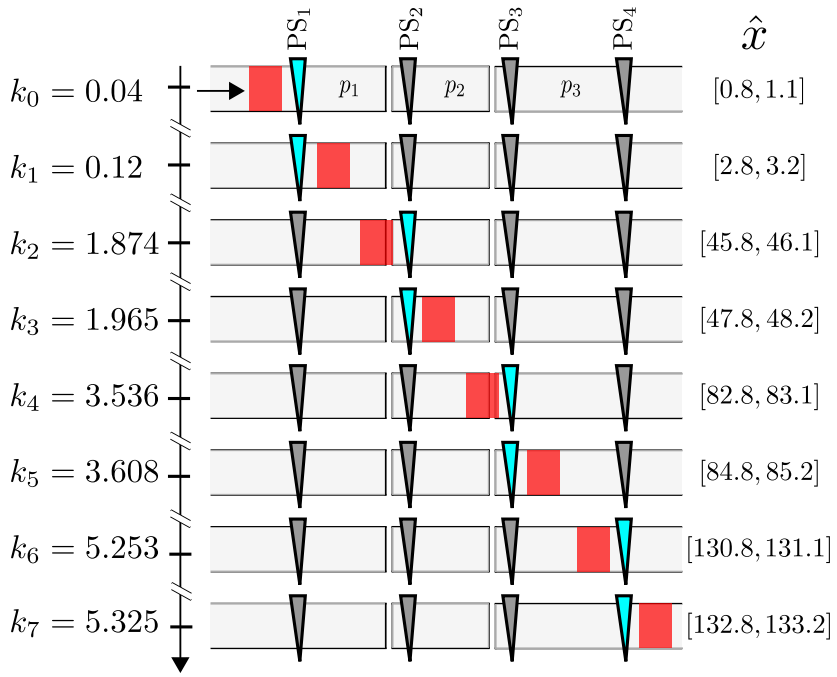


Figure 6.7: The position estimation with of the item with 4 sensors for the multi-belt case.

The grey horizontal section in Figure 6.7 are the three sequential belts and p_1, p_2, p_3 are their corresponding speeds. The figure serves illustrative purposes and thus is not in scale. The red region marks the estimated uncertainty range $\tilde{x}$ of the item, and the exact values are on the right. The proximity sensors are in grey when they are not triggered and in blue when the event corresponds to the particular one.

With the last scenario, we showed that the approach could not only be used for monitoring purposes but also for improving the system design. Therefore, we can identify potentially problematic sections during the design stage.

Test Plant Implementation

The presented approach was not only validated in simulation but also through implementation on a test plant, cf. Figure 6.8. To give a feeling of the execution times, we provide the results of six use cases of hardware-in-the-loop simulations on a test plant, cf. Table 6.3. The use cases are motivated by the Industry 4.0 guidelines [102]. The use cases vary from a validation of the healthy operation (use case 1), through parameter estimation (use case 4) and state estimation (use cases 5), up to fault diagnosis (use cases 2 & 3 & 6).

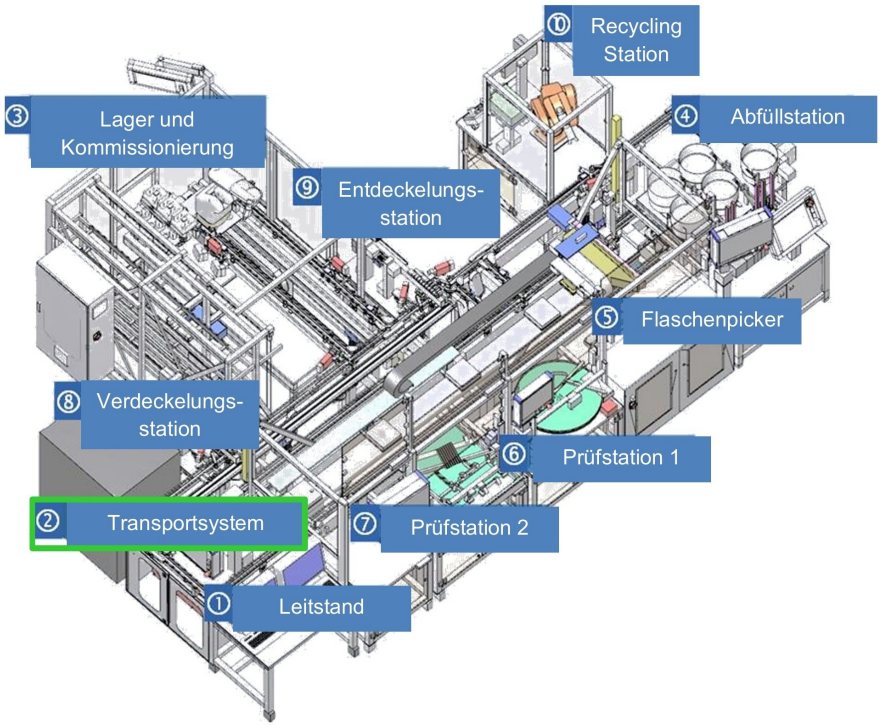


Figure 6.8: Sketch of the test plant [220].

The execution times were found to be consistent, which is a marker for reliability when implementing it on an industrial scale. Another indicator for the approach's applicability is the milliseconds' range of the times in Table 6.3. Taken together, the results in Table 6.3 prove the applicability of the approach.

6.5 Summary

In this chapter, we presented an approach for validating and monitoring the system behaviour of event-driven production systems. We reduce the complexity of such sys-

Table 6.3: Execution times of the considered use cases.

Use Case	Description	Execution Time	Unit
1	OK Check	3-4	ms
2	Detect defective sensor	4-5	ms
3	Detect defective actuator	31-35	ms
4	Parameter estimation	1-2	ms
5	Jam prediction	6-10	ms
6	Jam detection	9-14	ms

tems by abstracting the plant to a few basic types of elements. Afterwards, we derived tailored mathematical formulations for each type and demonstrated how to combine them to validate the system operation. To aid the implementation, we outlined an implementation framework that is designed to tackle plants with changing layouts and still function in real-time. Furthermore, the same approach can be used to identify missing events and even predict upcoming abnormal behaviour or faults. In summary, we demonstrated how the same SBE methods we used for controller validation can also be used for the monitoring and validation of plant-wide behaviours.

7 Conclusions

> All of physics is either impossible or trivial. It is impossible until you understand it and then it becomes trivial.
>
> Sir Ernest Rutherford

The main goal of this work is to aid the structured controller parameter tuning and controller design subject to quantitative and qualitative control requirements. We use set-based feasibility formulation to guarantee the tuning results despite the present unknown-but-bounded system uncertainties. The validation approach is illustrated through example scenarios ranging from the single control-loop level up to production-scale systems, and also varying from conceptual discussions through simulation studies to implementation results on test plants.

7.1 Summary

Tuning controllers has been and still is one of the core competences and duties of a control engineer. Although controller design and tuning are well studied and established for linear systems, there are still open questions, and there is an increasing need for suitable methods [33], [107]. This need is dictated by both the new applications and by the increasing system complexity. Each considered system properties like constraints, switching, time-variant parameters and other nonlinear dynamics complicates the design task significantly. Other challenges include uncertainties, algebraic constraints, multi-input multi-output structure and plant-model mismatch. Often any of these challenges either renders a well-established linear approach inapplicable or requires a significant adaptation [40]. It is even more challenging to rely on an approach that is capable of validating the system performance despite all the mentioned challenges.

To address the validation requirement and to provide guarantees, we work with set-based methods through a feasibility problem formulation [35], [36], [186]. Through the chosen set-based approach, we tackle one of the major challenges - uncertainties. The method allows to consider unknown-but-bounded uncertainties and provides a mathematically rigorous statement about the obtained results. We allow uncertainties to be present in all system elements, i.e. inputs, states, outputs, parameters, references, requirements. The proposed set-based approach allows to not discriminate between the different system elements but to consider them just as another variable inside the

problem. Such a formulation provides enough flexibility to formulate a variety of the different control tasks in a unified fashion.

Exploiting the problem formulation, with its polynomial form, inequalities, and discrete variables, we can express hybrid dynamical systems. Such systems have diverse characteristics and formulations. Therefore, we presented and formulated the phenomena that we handle and use throughout the provided examples, e.g. switching behaviour, cyclic dynamics, conditional dynamics, temporal and spatial uncertainties. In some cases, the hybrid system behaviour can originate from the controller or the control requirements.

Using verbal formulations for the requirements introduces challenges in posing them mathematically rigorously such that they can be taken into consideration during the controller design process. This arises from the complexity of the information, which often contains conditional requirements, temporal uncertainties, or combining different system elements. Therefore, we organised the requirements into two categories: quantitative and qualitative types of requirements.

Although set-based methods are regarded as computationally expensive, we demonstrated that computation times of milliseconds are possible. However, these times were achieved by combining the skills of a team of specialists that constructed a highly tailored implementation and custom industrial hardware. Based on these results and the experience we have gathered lead us to believe that there is significant potential to improve the computational times. Moreover, many of the estimation steps can be parallelised, e.g. the validation of the feasibility problems for each probing space. Such effort is worth pursuing as not many approaches nowadays can handle multi-input, multi-output, constrained, switching, temporally and spatially uncertain, nonlinear, event-based, hybrid systems and still provide guarantees.

7.2 Outlook

The approach presented in this work demonstrated how to validate the performance of a system and how to obtain appropriate sets of controller parameters. For all the cases, we have assumed that we have a given controller structure, and we are interested in its tuning. Thus, we avoided the question of choosing the controller structure. Considering the system class, one of the most direct ideas is to start with a low-order polynomial and, if it has been invalidated to increase the order iteratively and again look for parameterisations. This challenge is closely related to one of the fundamental problems in system identification, namely, which model structure to choose [71], [123]. Therefore, techniques from system identification could provide beneficial in figuring out the controller structure, e.g. Kolmogorov-Gabor Polynomial Models [155].

The focus of the work is, in one form or another, the controller: its tuning, formulation of requirements, tackling various control scenarios and validating its performance. Nevertheless, sometimes the inability to design a control system with the desired spec-

ifications could be due to the system dynamics itself. We can consider the problem of invalidating all controllers through one of the fundamental system constraints - the system input. Each controller can be abstracted to an input sequence, and using the sets that include these sequences leaves a single feasibility problem to be checked. Thus, by invalidating all input sequences that are admissible for the system, the effort of tuning a controller can be used to analyse the system. Another idea towards the analysis of the system itself is through adding slack variables to explore which of the requirements [74] or the system characteristics need to be relaxed such that the desired performance can be achieved [39], [66].

Another direction is putting the focus on the choice of the sampling time for nonlinear systems with switching behaviour [28], [31], which also echoes towards the field of self-triggered control [11], [135]. Through the feasibility formulation, we can investigate any of the system variables, even the time. Therefore, the feasibility problem can be re-structured such that we project the problem onto the time-space and perform SBE to obtain ranges of the sampling periods such that the performance is validated. A similar scenario is deciding the moment to switch from one controller to another such that certain requirements are guaranteed [32], [87], [120], [157].

Through the inner approximations of the for-all sets, we obtain only values that provide the desired behaviour. Nevertheless, we are still left with the choice of which particular parameterisation to pick and use for the implementation in the plant. One possibility is to divide the obtained inner approximation into subsets and to assign a performance value to each subset. The performance value could be based on a particular requirement, e.g. minimising the settling time or measuring and aiming at reducing the system uncertainties based on the controller parameterisations.

A Appendix

In this appendix, we present the particular values that we used for the various simulations and experimental implementations and were omitted in the main part of this work.

A.1 Details of the Li-ion battery example in Section 4.3.3

Table A.1 contains the estimated coefficients κ_i for the relation between the open circuit voltage and the state of the charge for the lithium-ion battery example from Section 4.3.3. Additionally, it lists the value for the offset parameter ξ. Table A.2 includes the thermal parameters values and the initial conditions. In Table A.3, we list the coefficients for the equivalent circuit model of the lithium ion battery from Section 4.3.3.

Table A.1: Fitted OCV-SOC coefficients and the SOC offset coefficient.

κ_1	κ_2	κ_3	κ_4	κ_5	κ_6	ξ
3.0545	0.8153	-0.5668	1.0457	0.2601	-0.0399	0.15

Table A.2: Parameter values for the thermal model and the initial condition.

Variable	Value	Unit
α	0.046	W J^{-1}
β	0.0196	Ω K J^{-1}
γ	0.098	W J^{-1}
$s(k_0)$	$[0, 0.01]$	-
T_{amb}	20	^{o}C
$T_S(k_0)$	20	^{o}C
$V_C(k_0)$	$[-0.01, 0]$	V
I	$[0.5, 2.0]$	A
$T_C(k_0)$	$[20, 20.4]$	^{o}C

Table A.3: Parameter values for the equivalent circuit model.

Variable	Value	Unit
R	$[0.25, 0.28]$	Ω
C_R	$[5031, 5365]$	F
R_C	$[0.093, 0.108]$	Ω

A.2 Details of the maglev example in Section 4.3.4

In Table A.4, we present the particular parameter values that we used in the maglev example in Section 4.3.4. The ranges for a, b, and α are generated by adding a 5% uncertainty to the nominal specification values.

Table A.4: Parameter values for the maglev example

Variable	Value	Unit
g	981	cm s^{-2}
m	0.125	kg
a	$[5.25 \cdot 10^{-5}, 5.75 \cdot 10^{-5}]$	$Vs^2kg^{-1}cm^{-3}$
b	$[2.33, 2.57]$	cm
α	$[14.9, 16.5]$	s^{-1}

A.3 Details of the example in Section 5.3

Here, we present the initial and desired ranges for the second-order system, used to illustrate the error-free steady-state behaviour in Section 5.3. The values in Table A.5 are for the set-point regulation task and in Table A.6 for the periodic set invariance stage.

Table A.5: Considered uncertainty ranges for the set-point regulation

State	$k_0 = 0$	$k_{\mathrm{SPR}} = 2.5$	Global constraints
e_1	[-4, -2]	[-0.15, 0.15]	[-5, 5]
e_2	[0. 0.01]	[-0.15, 0.15]	[-10, 10]
e_3	[-32, -24]	-	[-40, 40]

Table A.6: Considered uncertainty ranges for periodic set invariance

State	$k_0 = 0$	$k_{\text{RPI}} = 2.3$	Global constraints
e_1	[-0.15, 0.15]	[-0.15, 0.15]	[-0.5, 0.5]
e_2	[-0.15, 0.15]	[-0.15, 0.15]	[-1, 1]
e_3	[-3, 3]	[-3, 3]	[-7, 7]

A.4 Details of the Industry 4.0 example in Section 6.4.2

In this section, we provide the particular values for the four validation scenarios in Section 6.4.2. Table A.7 contains the initial estimation ranges for the parameter estimation scenario. In Table A.8, we detail the values for the scenario, in which we demonstrate the prediction and detection of a traffic jam capabilities of the approach.

Table A.7: The parameter values for the PS-PS module from Section 6.4.2.

Variable	Value	Unit
p_{PS_1}	$[-0.2, 0.2]$	cm
p_{PS_2}	$[87.8, 88.2]$	cm
p_{size}	$[2.8, 3.2]$	cm
v	$[0, 100]$	cm s^{-1}
Δv	$[0, 0.1]$	cm s^{-1}

Table A.8: Considered ranges for the traffic jam scenario

Variable	Value	Unit
p_{zone}	40	cm
$k_{\text{PS}}^{\text{in}}$	0.041	s
$k_{\text{PS}}^{\text{out}}$	0.121	s
p	$[24.5, 25]$	cm s^{-1}
$d(k)$[cm/s]	$[0, 0.1]$	cm s^{-1}
p_{size}	$[2, 2.07]$	cm

To illustrate the state estimation capabilities, we consider a module with two consequential actuated gates and the simulation values are in Table A.9.

Table A.9: The values for the PS-AG-AG-PS example from Section 6.4.2.

Variable	Value	Unit
x	$[0, 50]$	cm
p_{PS_1}	$[0, 15]$	cm
p_{PS_2}	$[20, 40]$	cm
$\hat{p}_{\mathrm{PS}_1}$	$[9.9, 11.2]$	cm
$\hat{p}_{\mathrm{PS}_2}$	$[28.9, 31.3]$	cm
p_{AG_1}	$[14, 16]$	cm
p_{AG_2}	$[19, 21]$	cm
p	$[24.8, 25]$	cm s^{-1}
z	$[0,\ 0.1]$	cm s^{-1}
p_{size}	$[2,\ 2.08]$	cm

The initial ranges for the multi-belt scenario are presented in Table A.10 and the estimated transition times and belt velocities are given in Table 6.2.

Table A.10: Considered initial ranges for the multi-belt scenario

Variable	Value	Unit
$x(k)$	$[0, 500]$	cm
p_{PS_1}	$[0.9, 1.1]$	cm
p_{PS_2}	$[45.9, 46.1]$	cm
p_{PS_3}	$[82.9, 83.1]$	cm
p_{PS_4}	$[130.9, 13.1]$	cm
p_1	$[20, 30]$	cm s^{-1}
p_2	$[20, 30]$	cm s^{-1}
p_3	$[20, 30]$	cm s^{-1}
$d(k)$	$[0, 0.1]$	cm s^{-1}
p_{size}	$[1.98, 2.02]$	cm

Bibliography

[1] A. Abdollahi, X. Han, G. Avvari, N. Raghunathan, B. Balasingam, K. Pattipati, and Y. Bar-Shalom. Optimal battery charging, Part I: Minimizing time-to-charge, energy loss, and temperature rise for OCV-resistance battery model. *Journal of Power Sources*, 303:388 – 398, 2016.

[2] L. Adolph, E. Ammon, J. Becker, H. Bedenbender, V. Bellinghausen, M. Börkircher, A. Braunmandl, L. Brumby, J. Cäsar, D. Czarny, J. Meer, C. Diedrich, M. Fliehe, J. Friedrich, G. Focke, J. Gayko, J. Gobert, A. Harner, W. Hartmann, and D. Wegener. *Deutsche Normungsroadmap Industrie 4.0 - Version 4*. 2020. in German.

[3] A. P. Aguiar, D. B. Dačić, J. P. Hespanha, and P. Kokotović. Path-following or reference tracking?: An answer relaxing the limits to performance. In *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, volume 37, pages 167–172, 2004.

[4] A. P. Aguiar and J. P. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, 2007.

[5] C. Akoto and H. Spangenberg. Modeling of backlash in drivetrains. In *Proceedings of the 4th CEAS Air & Space Conference*, 2013.

[6] J. Anderson and A. Papachristodoulou. On validation and invalidation of biological models. *BMC bioinformatics*, 10(1):132, 2009.

[7] P. Andonov, B. Morabito, A. Savchenko, and R. Findeisen. Admissible Control Parametrization of Uncertain Finite-time Processes With Application to Li-ion Battery Management. In *Proceedings of the European Control Conference*, pages 2338–2343, Limasol, Cyprus, 2018.

[8] P. Andonov, A. Savchenko, and R. Findeisen. Controller Parametrization for Offset-free Control Using Set-Based Feasibility Methods. In *Proceedings of the American Control Conference*, pages 2795–2800, Philadelphia, USA, 2019.

[9] P. Andonov, A. Savchenko, P. Rumschinski, S. Streif, and R. Findeisen. Controller Verification and Parametrization Subject to Quantitative and Qualitative Requirements. In *Proceedings of the International Symposium on Advanced Control of Chemical Processes*, pages 1174–1179, Whistler, Canada, 2015.

[10] P. Andonov, A. Savchenko, P. Rumschinski, T. Trenner, J. Neidig, and R. Findeisen. Monitoring and verification of event-driven discrete manufacturing systems with guarantees. In *Proceedings of the IEEE Conference on Control Tech-*

nology and Applications, pages 1029–1035, Montreal, Canada, 2020.

[11] A. Anta and P. Tabuada. To sample or not to sample: Self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030–2042, 2010.

[12] P. Antsaklis, X. Koutsoukos, and J. Zaytoon. On hybrid control of complex systems: A survey. *European Journal of Automation*, 32(9-10):1023 –1045, 1998.

[13] K. J. Åström and T. Hägglund. *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society, 2006.

[14] J.-P. Aubin and A. Cellina. *Differential inclusions: set-valued maps and viability theory*, volume 264. Springer Science & Business Media, 2012.

[15] J.-P. Aubin. A survey of viability theory. *SIAM Journal on Control and Optimization*, 28(4):749–788, 1990.

[16] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre. *Viability theory: new directions*. Springer Science & Business Media, 2011.

[17] J.-P. Aubin and H. Frankowska. *Set-valued analysis*. Springer Science & Business Media, 2009.

[18] E.-W. Bai, K. Nagpal, and R. Tempo. Bounded error parameter estimation: noise models, recursive algorithms and H_∞ optimality. In *Proceedings of the IEEE American Control Conference*, volume 5, pages 3065–3069, 1995.

[19] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer. Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data. In *Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 586–597, 2018.

[20] Batman equation. https://gist.github.com/traeblain/1119139. Accessed: 2021-02-10.

[21] C. Belta and S. Sadraddini. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:115–140, 2019.

[22] C. Belta, B. Yordanov, and E. A. Gol. *Formal methods for discrete-time dynamical systems*, volume 89. Springer, 2017.

[23] A. Bemporad, W. M. H. Heemels, and B. De Schutter. On hybrid systems and closed-loop mpc systems. *IEEE Transactions on Automatic Control*, 47(5):863–869, 2002.

[24] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[25] D. P. Bertsekas and I. B. Rhodes. On the minimax reachability of target sets and target tubes. *Automatica*, 7(2):233–247, 1971.

[26] J. Bethge, B. Morabito, H. Rewald, A. Ahsan, S. Sorgatz, and R. Findeisen. Modelling human driving behavior for constrained model predictive control in

mixed traffic at intersections. In *Proceedings of the 21st IFAC World Congress*, pages 14557 – 14563, Berlin, Germany, 2020.

[27] G. Betti, M. Farina, and R. Scattolini. A robust MPC algorithm for offset-free tracking of constant reference signals. *IEEE Transactions on Automatic Control*, 58(9):2394–2400, 2013.

[28] W. Bian and M. French. General fast sampling theorems for nonlinear systems. *Systems & control letters*, 54(11):1037–1050, 2005.

[29] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers, Carnegie Mellon University*, 58, 2003.

[30] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan. Linear encodings of bounded LTL model checking. *Logical Methods in Computer Science*, 2(5:5):1–64, 2006.

[31] S. Billings and L. A. Aguirre. Effects of the sampling time on the dynamics and identification of nonlinear models. *International Journal of Bifurcation and Chaos*, 5(06):1541–1556, 1995.

[32] F. Blanchini and S. Miani. *Set-theoretic methods in control*. Springer, 2015.

[33] M. J. Blondin, J. S. Sáez, and P. M. Pardalos. Control engineering from classical to intelligent control theory—an overview. In *Computational Intelligence and Optimization Methods for Control Engineering*, chapter 1, pages 1–30. Springer, 2019.

[34] S. Borchers, S. Bosio, P. Rumschinski, A. Savchenko, R. Weimantel, and R. Findeisen. Set-membership estimation and analysis of polynomial systems: A relaxation-based framework. Technical report, Magdeburg, Germany, 2012.

[35] S. Borchers, P. Rumschinski, S. Bosio, R. Weismantel, and R. Findeisen. Model invalidation and system identification of biochemical reaction networks. In *Proceedings of the 16th IFAC Symposium on Identification and System Parameter Estimation*, Saint Malo, France, 2009.

[36] S. Borchers, P. Rumschinski, S. Bosio, R. Weismantel, and R. Findeisen. A set-based framework for coherent model invalidation and parameter estimation of discrete time nonlinear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control held jointly with 2009 28th Chinese Control Conference*, pages 6786–6792, Shanghai, China, 2009.

[37] J. Borghoff. Mixed-integer linear programming in the analysis of Trivium and Ktantan. *IACR Cryptology ePrint Archive*, 2012:676, 2012.

[38] P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of TPTL and MTL. In *Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 432–443, Hyderabad, India, 2005.

[39] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University

Press, 2004.

[40] S. P. Boyd and C. H. Barratt. *Linear controller design: limits of performance*. Prentice Hall Englewood Cliffs, New Jersey, 1991.

[41] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[42] J. M. Bravo, T. Alamo, and E. F. Camacho. Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets. *Automatica*, 42(10):1745–1751, 2006.

[43] R. W. Brockett. Smooth multimode control systems. In *Proceedings Berkeley-Ames Conference on Nonlinear Problems in Control and Fluid Dynamics*, pages 103–110, Berkeley, USA, 1984.

[44] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

[45] S. R. Buss. An introduction to proof theory. *Handbook of proof theory*, 137:1–78, 1998.

[46] C. Byrnes and A. Isidori. Steady state response, separation principle and the output regulation of nonlinear systems. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 2247–2251, Tampa, USA, 1989.

[47] M. Cannon, V. Deshmukh, and B. Kouvaritakis. Nonlinear model predictive control with polytopic invariant sets. *Automatica*, 39(8):1487–1494, 2003.

[48] M. Cassaro, C. Roosl, and M. J-Biannicl. Robust dynamic allocation for aircraft roll-out phase directional control. In *Proceedings of the European Control Conference*, pages 2763–2768, Limassol, Cyprus, 2018.

[49] T. M. Cavalier, P. M. Pardalos, and A. L. Soyster. Modeling and integer programming techniques applied to propositional calculus. *Computers & Operations Research*, 17(6):561–570, 1990.

[50] B. Chachuat, B. Houska, R. Paulen, N. Peri'c, J. Rajyaguru, and M. E. Villanueva. Set-theoretic approaches in analysis, estimation and control of nonlinear systems. *IFAC-PapersOnLine*, 48(8):981–995, 2015.

[51] S.-Y. Cheong and M. G. Safonov. Bumpless transfer for adaptive switching controls. In *Proceedings of the 17th IFAC World Congress*, volume 41, pages 14415–14420, Seoul, Korea, 2008.

[52] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71, Yorktown Heights, USA, 1981.

[53] C. Coey, M. Lubin, and J. P. Vielma. Outer approximation with conic certificates for mixed-integer convex problems. *arXiv preprint arXiv:1808.05290*, 2018.

[54] N. Cristianini, J. Shawe-Taylor, et al. *An introduction to support vector machines*

and other kernel-based learning methods. Cambridge University Press, 2000.

[55] R. David and H. Alla. *Discrete, continuous, and hybrid Petri nets*, volume 1. Springer, 2005.

[56] J. De Schutter, R. Leuthold, and M. Diehl. Optimal control of a rigid-wing rotary kite system for airborne wind energy. In *Proceedings of the European Control Conference*, pages 1734–1739, Limasol, Cyprus, 2018.

[57] S. Dey, Z. Abdollahi Biron, S. Tatipamula, N. Das, S. Elizabeth Mohon, B. Ayalew, and P. Pisu. On-board thermal fault diagnosis of lithium-ion batteries for hybrid electric vehicle application. In *Proceedings of the 4th IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling*, volume 48, pages 389–394, Columbus, USA, 2015.

[58] B. Di Giampaolo, S. La Torre, and M. Napoli. Parametric metric interval temporal logic. In *Proceedings of the International Conference on Language and Automata Theory and Applications*, pages 249–260, Trier, Germany, 2010.

[59] R. C. Dorf and R. H. Bishop. *Modern control systems*. Pearson (Addison-Wesley), 1998.

[60] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, volume 9, pages 155–161, 1997.

[61] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.

[62] E. A. Emerson and J. Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[63] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.

[64] P. Feketa, S. Bogomolov, and T. Meurer. Safety verification for impulsive systems. In *Proceedings of the 21st IFAC World Congress*, pages 1979 – 1984, Berlin, Germany, 2020.

[65] M. Fisher. *An introduction to practical formal methods using temporal logic*, volume 82. Wiley Online Library, 2011.

[66] S. M. Focardi and F. J. Fabozzi. *The mathematics of financial modeling and investment management*, volume 138. John Wiley & Sons, 2004.

[67] E. Frazzoli, M. A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicle motion planning. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 1, pages 821–826, Sydney, Australia, 2000.

[68] R. Freeman and P. V. Kokotovic. *Robust nonlinear control design: state-space*

and Lyapunov techniques. Springer Science & Business Media, 2008.

[69] T. French. Quantified propositional temporal logic with repeating states. In *Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and 4th International Conference on Temporal Logic*, pages 155–165, Cairns, Australia, 2003.

[70] T. Gally, M. E. Pfetsch, and S. Ulbrich. A framework for solving mixed-integer semidefinite programs. *Optimization Methods and Software*, 33(3):594–632, 2018.

[71] E. Garipov. *Identification of systems*. Technical University Sofia, 1997. in Bulgarian.

[72] T. Gedeon. *Cyclic feedback systems*, volume 637. American Mathematical Society, 1998.

[73] H. J. Genrich, H.-M. Hanisch, and K. Wöllhaf. Verification of recipe-based control procedures by means of predicate/transition nets. In *Proceedings of the International Conference on Application and Theory of Petri Nets*, pages 278–297, Zaragoza, Spain, 1994.

[74] S. Ghosh, D. Sadigh, P. Nuzzo, V. Raman, A. Donzé, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia. Diagnosis and repair for synthesis from signal temporal logic specifications. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 31–40, Vienna, Austria, 2016.

[75] A. Girard. Symbolic control - from discrete synthesis to certified continuous controllers. https://drive.google.com/open?id=11-YC4rouiZMgND-1xaNOqVuh9Kcg3d6, Accessed: 2021-02-10, 2018. European Control Conference.

[76] P. F. Glenn. The politics of truth: Power in Nietzsche's epistemology. *Political Research Quarterly*, 57(4):575–583, 2004.

[77] R. Goebel, R. G. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009.

[78] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[79] D.-W. Gu, P. Petkov, and M. M. Konstantinov. *Robust control design with MATLAB ®*. Springer Science & Business Media, 2005.

[80] L. Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[81] W. M. Haddad, V. Chellaboina, and S. G. Nersesov. *Impulsive and hybrid dynamical systems: stability, dissipativity, and control*. Princeton University Press, 2014.

[82] T. Hägglund. Automatic on-line estimation of backlash in control loops. *Journal of Process Control*, 17(6):489–499, 2007.

[83] H.-S. Han and K. Dong-sung. *Magnetic Levitation: Maglev Technology and Applications*. Springer, 2016.

[84] R. Hanus, M. Kinnaert, and J.-L. Henrotte. Conditioning technique, a general anti-windup and bumpless transfer method. *Automatica*, 23(6):729–739, 1987.

[85] E. L. Harder, P. O. Langguth, and C. A. Woods. Transient and steady-state performance of potential devices. *Electrical Engineering*, 59(2):91–102, 1940.

[86] C. Heinz and B. Seeger. Statistical modeling of sensor data and its application to outlier detection. In *Proceedings of the 5. GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze "*, pages 49–53, Stuttgart, Germany, 2006.

[87] J. P. Hespanha and A. S. Morse. Switching between stabilizing controllers. *Automatica*, 38(11):1905–1917, 2002.

[88] G. Hoffmann, S. Waslander, and C. Tomlin. Quadrotor helicopter trajectory tracking control. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 7410–7423, Honolulu, USA, 2008.

[89] A. Hossain and F. Laroussinie. From Quantified CTL to QBF. In *Proceedings of the 26th International Symposium on Temporal Representation and Reasoning*, volume 147, pages 11:1–11:20, Dagstuhl, Germany, 2019.

[90] S. Hu and N. Papageorgiou. *Handbook of multivalued analysis. Volume I: Theory*, volume 419 of *Mathematics and Its Applications*. Springer US, 1997.

[91] K. Iagnemma, S. Shimoda, and Z. Shiller. Near-optimal navigation of high speed mobile robots on uneven terrain. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4098–4103, Nice, France, 2008.

[92] IBM ILOG CPLEX. V12. 1: User's manual for cplex. *Int. Business Machines Corp.*, 46(53):157, 2009.

[93] A. O. Ignatyev. On the stability of invariant sets of systems with impulse effect. *Nonlinear Analysis: Theory, Methods & Applications*, 69(1):53–72, 2008.

[94] A. Ilchmann, E. P. Ryan, and C. J. Sangwin. Tracking with prescribed transient behaviour. *ESAIM: Control, Optimisation and Calculus of Variations*, 7:471–493, 2002.

[95] A. Ilchmann, E. P. Ryan, and S. Trenn. Tracking control: Performance funnels and prescribed transient behaviour. *Systems & Control Letters*, 54(7):655–670, 2005.

[96] A. Isidori. *Nonlinear control systems II*. Springer, 1999.

[97] L. Jaulin. Interval constraint propagation with application to bounded-error estimation. *Automatica*, 36(10):1547–1552, 2000.

[98] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied interval analysis*. Springer, 2001.

[99] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993.

[100] L. Jaulin and É. Walter. Guaranteed tuning, with application to robust control

and motion planning. *Automatica*, 32(8):1217–1221, 1996.

[101] W. Jiang, C. Dong, and Q. Wang. A systematic method of smooth switching LPV controllers design for a morphing aircraft. *Chinese Journal of Aeronautics*, 28(6):1640–1649, 2015.

[102] H. Kagermann, W. Wahlster, and J. Helbig. Recommendations for implementing the strategic initiative industrie 4.0. Technical report, Federal Ministry of Education and Research of Germany, 2013.

[103] L. Keviczky, R. Bars, J. Hetthéssy, and C. Bányász. *Control engineering*. Springer, 2019.

[104] H. K. Khalil. *Nonlinear control*. Pearson Higher Ed, 2014.

[105] E. Kofman. Discrete event simulation of hybrid systems. *SIAM Journal on Scientific Computing*, 25(5):1771–1797, 2004.

[106] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.

[107] S. Kozák. State-of-the-art in control engineering. *Journal of Electrical Systems and Information Technology*, 1(1):1 – 9, 2014.

[108] O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[109] L. Lapierre and D. Soetanto. Nonlinear path-following control of an AUV. *Ocean Engineering*, 34(11-12):1734–1744, 2007.

[110] J. B. Lasserre. Semidefinite programming vs. LP relaxations for polynomial programming. *Mathematics of Operations Research*, 27(2):347–360, 2002.

[111] A. Lavaei, S. Soudjani, and M. Zamani. Compositional construction of infinite abstractions for networks of stochastic control systems. *Automatica*, 107:125 – 137, 2019.

[112] L. S. Lawrence, J. W. Simpson-Porco, and E. Mallada. Linear-convex optimal steady-state control. *arXiv preprint arXiv:1810.12892*, 2018.

[113] J. Lee, C. Yoo, Y.-S. Park, B. Park, S.-J. Lee, D.-G. Gweon, and P.-H. Chang. An experimental study on time delay control of actuation system of tilt rotor unmanned aerial vehicle. *Mechatronics*, 22(2):184–194, 2012.

[114] Y. I. Lee and B. Kouvaritakis. Constrained robust model predictive control based on periodic invariance. *Automatica*, 42(12):2175–2181, 2006.

[115] P. Lehn and M. Irvani. Discrete time modeling and control of the voltage source converter for improved disturbance rejection. *IEEE Transactions on Power Electronics*, 14(6):1028–1036, 1999.

[116] B. León de la Barra. On undershoot in SISO systems. *IEEE Transactions on Automatic Control*, 39(3):578–581, 1994.

[117] W. S. Levine. *The Control Handbook (three volume set)*. CRC press, 2018.

[118] B. Li and A. G. Alleyne. A dynamic model of a vapor compression cycle with shut-down and start-up operations. *International Journal of Refrigeration*, 33(3):538–552, 2010.

[119] R. Li and H. K. Khalil. On the steady-state error of a nonlinear regulator. *International Journal of Robust and Nonlinear Control*, 23(16):1869–1879, 2013.

[120] D. Liberzon. *Switching in systems and control*. Springer Science & Business Media, 2003.

[121] D. Limon, J. Bravo, T. Alamo, and E. Camacho. Robust MPC of constrained nonlinear systems based on interval arithmetic. In *Proceedings of the IEEE International Conference on Control Applications*, volume 152, pages 325–332, Glasgow, UK, 2005.

[122] J. Liu and N. Ozay. Abstraction, discretization, and robustness in temporal logic control of dynamical systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, pages 293–302, New York, USA, 2014.

[123] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, 2 edition, 1999.

[124] S. Lucia, M. Kögel, and R. Findeisen. Contract-based predictive control of distributed systems with plug and play capabilities. In *Proceedings of the 5th IFAC Conference on Nonlinear Model Predictive Control*, volume 48, pages 205–211, Seville, Spain, 2015.

[125] J. Lunze and F. Lamnabhi-Lagarrigue. *Handbook of hybrid systems control: theory, tools, applications*. Cambridge University Press, 2009.

[126] Y. Luo, W. Ren, Y. Huang, Q. He, Q. Wu, X. Zhou, and Y. Mao. Feedforward control based on error and disturbance observation for the ccd and fiber-optic gyroscope-based mobile optoelectronic tracking system. *Electronics*, 7(10):223, 2018.

[127] J. Lygeros, S. Sastry, and C. Tomlin. Hybrid systems: Foundations, advanced topics and applications. https://inst.eecs.berkeley.edu/~ee291e/sp21/handouts/hybridSystems_monograph.pdf, Accessed: 2021-02-11.

[128] O. J. M. Smith. Posicast control of damped oscillatory systems. *Proceedings of the IRE*, 45(9):1249–1255, 1957.

[129] U. Maeder, F. Borrelli, and M. Morari. Linear offset-free model predictive control. *Automatica*, 45(10):2214–2222, 2009.

[130] Model 730: Magnetic levitation - educational control products. http://www.ecpsystems.com/controls_maglevit.htm. Accessed: 2020-10-25.

[131] S. Malan, M. Milanese, and M. Taragna. Robust analysis and design of control systems using interval arithmetic. *Automatica*, 33(7):1363–1372, 1997.

[132] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals.

In *Formal Modeling and Analysis of Timed Systems*, pages 152–166, Grenoble, France, 2004.

[133] N. M. Mangan, T. Askham, S. L. Brunton, J. N. Kutz, and J. L. Proctor. Model selection for hybrid dynamical systems via sparse regression. *Proceedings of the Royal Society A*, 475(2223):20180534, 2019.

[134] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer Science & Business Media, 2012.

[135] N. Marchand, S. Durand, and J. F. G. Castellanos. A general formula for event-based stabilization of nonlinear systems. *IEEE Transactions on Automatic Control*, 58(5):1332–1337, 2012.

[136] J. Margielewicz, D. Gąska, and G. Litak. Modelling of the gear backlash. *Nonlinear Dynamics*, 97(1):355–368, 2019.

[137] J. Matschek, T. Bäthge, T. Faulwasser, and R. Findeisen. Nonlinear predictive control for trajectory tracking and path following: An introduction and perspective. In *Handbook of Model Predictive Control*, pages 169–198. Springer, 2019.

[138] A. S. Matveev and A. V. Savkin. *Qualitative theory of hybrid dynamical systems*. Springer Science & Business Media, 2012.

[139] D. Q. Mayne and E. C. Kerrigan. Tube-based robust nonlinear model predictive control. In *Proceedings of the 7th IFAC Symposium on Nonlinear Control Systems*, volume 40, pages 36–41, Pretoria, South Africa, 2007.

[140] M. Mazo, A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *Proceedings of the International Conference on Computer Aided Verification*, pages 566–569, Edinburgh, United Kingdom, 2010.

[141] M. Milanese, J. Norton, H. Piet-Lahanier, and É. Walter. *Bounding approaches to system identification*. Springer Science & Business Media, 2013.

[142] M. Milanese and A. Vicino. Estimation theory for nonlinear models and set membership uncertainty. *Automatica*, 27(2):403–408, 1991.

[143] R. Milner. *Communication and concurrency*, volume 84. Prentice Hall Englewood Cliffs, 1989.

[144] T. Mita and H. Yoshida. Undershooting phenomenon and its control in linear multivariable servomechanisms. *IEEE Transactions on Automatic Control*, 26(2):402–407, 1981.

[145] J. O. Moody and P. J. Antsaklis. *Supervisory control of discrete event systems using Petri nets*, volume 8. Springer Science & Business Media, 2012.

[146] R. Moore. Parameter sets for bounded-error data. *Mathematics and Computers in Simulation*, 34(2):113–119, 1992.

[147] R. E. Moore. *Methods and applications of interval analysis*. SIAM, 1979.

[148] B. Morabito, A. Kienle, R. Findeisen, and L. Carius. Multi-mode model predic-

tive control and estimation for uncertain biotechnological processes. In *Proceedings of the 12th IFAC Symposium on Dynamics and Control of Process Systems including Biosystems*, pages 709–714, Florianopolis, Brazil, 2019.

[149] M. Morari and E. Zafiriou. *Robust process control*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1989.

[150] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[151] J. Mossoba and P. T. Krein. Exploration of deadbeat control for DC-DC converters as hybrid systems. In *Proceedings of the 36th IEEE Power Electronics Specialists Conference*, pages 1004–1010, Recife, Brazil, 2005.

[152] T. Mühl. *Einführung in die elektrische Messtechnik: Grundlagen, Messverfahren, Anwendungen*. Springer-Verlag, 2014.

[153] A. R. G. Mukkula and R. Paulen. Optimal experiment design in nonlinear parameter estimation with exact confidence regions. *Journal of Process Control*, 83:187–195, 2019.

[154] G. J. Murphy. *Basic automatic control theory*. D. Van Nostrand Company, Inc., 1957.

[155] O. Nelles. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2013.

[156] C. A. Nickle. Electrical regulator, January 14 1930. US Patent 1,743,797.

[157] H. Niemann, J. Stoustrup, and R. B. Abrahamsen. Switching between multivariable controllers. *Optimal Control Applications and Methods*, 25(2):51–66, 2004.

[158] N. Nigam and I. Kroo. Control and design of multiple unmanned air vehicles for a persistent surveillance task. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, Canada, 2012.

[159] N. S. Nise. *Control systems engineering*. Wiley, 2015.

[160] C. Novara and M. Milanese. Set membership identification of nonlinear systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 3, pages 2831–2836, Sydney, Australia, 2000.

[161] A. O'Dwyer. *Handbook of PI and PID controller tuning rules*. Imperial College Press, 2009.

[162] K. Ogata. *Modern control engineering*. Prentice Hall, London, 2009.

[163] H. Osumi, M. Kubo, S. Yano, and K. Saito. Development of tele-operation system for a crane without overshoot in positioning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5799–5805, Taipei, Taiwan, 2010.

[164] W. J. Palm. *System dynamics*, volume 7. McGraw-Hill Higher Education New York, 2005.

[165] G. Pannocchia, M. Gabiccini, and A. Artoni. Offset-free MPC explained: novelties, subtleties, and applications. In *Proceedings of the 5th IFAC Conference on Nonlinear Model Predictive Control*, volume 48, pages 342–351, Seville, Spain, 2015.

[166] G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free model-predictive control. *AIChE journal*, 49(2):426–437, 2003.

[167] I. Pardoe. *Applied regression modeling*. John Wiley & Sons, 2020.

[168] C. L. Phillips and R. D. Habor. *Feedback control systems*. Simon & Schuster, Inc., 1995.

[169] G. M. Phillips. *Interpolation and approximation by polynomials*, volume 14. Springer Science & Business Media, 2003.

[170] B. Pluymers, J. Rossiter, J. Suykens, and B. De Moor. A simple algorithm for robust MPC. In *Proceedings of the 16th IFAC world congress*, Prague, Czech Republic, 2005.

[171] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[172] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Proceedings of the International Workshop on Hybrid Systems: Computation and Control*, pages 477–492, 2004.

[173] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[174] S. V. Raković, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen. Fully parameterized tube MPC. In *Proceedings of the 18th IFAC World Congress*, volume 44, pages 197–202, Milano, Italy, 2011.

[175] V. Ramadesigan, P. W. Northrop, S. De, S. Santhanagopalan, R. D. Braatz, and V. R. Subramanian. Modeling and simulation of lithium-ion batteries from a systems engineering perspective. *Journal of the Electrochemical Society*, 159(3):R31–R45, 2012.

[176] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[177] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, pages 81–87, Los Angeles, USA, 2014.

[178] F. Ranzato and F. Tapparo. Strong preservation of temporal fixpoint-based operators by abstract interpretation. In *Proceedings of the International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 332–347, Charleston, USA, 2006. Springer.

[179] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model predictive control: theory,*

computation, and design, volume 2. Nob Hill Publishing Madison, WI, 2017.

[180] G. Rebala, A. Ravi, and S. Churiwala. *An introduction to machine learning*. Springer, 2019.

[181] J. H. Richter and S. R. Friedrich. Semi-formal verification of closed-loop specifications in the concept design phase. *at-Automatisierungstechnik*, 65(2):115–123, 2017.

[182] A. Rosenblueth and N. Wiener. The role of models in science. *Philosophy of science*, 12(4):316–321, 1945.

[183] J. Rossiter. *A first course in predictive control*. CRC Press, 2018.

[184] N. Rudolph. *Set-based multi-scale modeling and analysis signal transduction pathways*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2019.

[185] N. Rudolph, P. Andonov, H. J. Huber, and R. Findeisen. Model-supported patient stratification using set-based estimation methods. In *In Proceedings of the International Symposium on Advanced Control of Chemical Processes*, pages 892–897, Shenyang, China, 2018.

[186] P. Rumschinski, S. Borchers, S. Bosio, R. Weismantel, and R. Findeisen. Set-based dynamical parameter estimation and model invalidation for biochemical reaction networks. *BMC Systems Biology*, 4(1):69, 2010.

[187] P. Rumschinski, S. Streif, and R. Findeisen. Combining qualitative information and semi-quantitative data for guaranteed invalidation of biochemical network models. *International Journal of Robust and Nonlinear Control*, 22(10):1157–1173, 2012.

[188] D. Sangiorgi. *Introduction to bisimulation and coinduction*. Cambridge University Press, 2011.

[189] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone. Taming Dr. Frankenstein: Contract-based design for cyber-physical systems. *European Journal of Control*, 18(3):217–238, 2012.

[190] D. U. Sauer, G. Bopp, A. Jossen, J. Garche, M. Rothert, and M. Wollny. State of charge–what do we really speak about. In *Proceedings of the 21st International Telecommunications Energy Conference*, pages 6–9, Copenhagen, Denmark, 1999.

[191] C. J. Savant. *Basic feedback control system design*. McGraw-Hill, 1958.

[192] A. Savchenko. *Efficient set-based process monitoring and fault diagnosis*. PhD thesis, Shaker Verlag Aachen, 2017.

[193] A. Savchenko, P. Andonov, P. Rumschinski, and R. Findeisen. Multi-objective complexity reduction for set-based fault diagnosis. In *Proceedings of the 6th International Symposium on Advanced Control of Industrial Processes*, pages 589–594, Taipei, Taiwan, 2017.

[194] A. Savchenko, P. Andonov, S. Streif, and R. Findeisen. Guaranteed set-based

controller parameter estimation for nonlinear systems–magnetic levitation platform as a case study. In *Proceedings of the 19th IFAC World Congress*, pages 4650–4655, Cape Town, South Africa, 2014.

[195] A. Savchenko, P. Rumschinski, and R. Findeisen. Fault diagnosis for polynomial hybrid systems. In *Proceedings of the 18th IFAC World Congress*, volume 44, pages 2755–2760, Milano, Italy, 2011.

[196] A. Savchenko, P. Rumschinski, S. Streif, and R. Findeisen. Complete diagnosability of abrupt faults using set-based sensitivities. In *Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, volume 45, pages 860–865, Mexico, Mexico.

[197] K. Schittkowski. Parameter estimation in dynamic systems. In *Progress in Optimization*, pages 183–204. Springer, 2000.

[198] F. Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968.

[199] S. Scialanga, S. Olaru, and K. Ampountolas. Interpolating control with periodic invariant sets. In *Proceedings of the European Control Conference*, 2020.

[200] SeDuMi, http://sedumi.ie.lehigh.edu/, 2020.

[201] N. Seube, R. Moitie, and G. Leitmann. Aircraft take-off in windshear: a viability approach. *Set-Valued Analysis*, 8(1-2):163–180, 2000.

[202] J. S. Shamma and M. Athans. Gain scheduling: Potential hazards and possible remedies. *IEEE Control Systems Magazine*, 12(3):101–107, 1992.

[203] W. Shen, T. T. Vo, and A. Kapoor. Charging algorithms of lithium-ion batteries: An overview. In *Proceedings of the 7th IEEE Conference on Industrial Electronics and Applications*, pages 1567–1572, Singapore, 2012.

[204] B. Siciliano and O. Khatib. *Springer handbook of robotics*. Springer, 2016.

[205] A. Sidhu, A. Izadian, and S. Anwar. Adaptive nonlinear model-based fault diagnosis of li-ion batteries. *IEEE Transactions Industrial Electronics*, 62(2):1002–1011, 2015.

[206] J.-J. E. Slotine, W. Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, New Jersey, 1991.

[207] E. D. Sontag. Input to state stability: Basic concepts and results. In *Nonlinear and optimal control theory*, pages 163–220. Springer, 2008.

[208] D. Spinello and D. J. Stilwell. Nonlinear estimation with state-dependent gaussian observation noise. *IEEE Transactions on Automatic Control*, 55(6):1358–1366, 2010.

[209] J. Stoustrup. Plug & play control: Control technology towards new challenges. *European Journal of Control*, 15(3-4):311–330, 2009.

[210] S. Streif, A. Savchenko, P. Rumschinski, S. Borchers, and R. Findeisen. ADMIT: a toolbox for guaranteed model invalidation, estimation and qualita-

tive–quantitative modeling. *Bioinformatics*, 28(9):1290–1291, 2012.

[211] S. Streif, N. Strobel, and R. Findeisen. Inner approximations of consistent parameter sets by constraint inversion and mixed-integer programming. In *Proceedings of the 12th IFAC Symposium on Computer Applications in Biotechnology*, volume 46, pages 321–326, Mumbai, India, 2013.

[212] S. W. Su, B. D. Anderson, and T. S. Brinsmead. Constant disturbance rejection and zero steady state tracking error for nonlinear systems design. In *Applied and Computational Control, Signals, and Circuits*, pages 1–30. Springer, 2001.

[213] D. Sun, C. Wang, W. Shang, and G. Feng. A synchronization approach to trajectory tracking of multiple mobile robots while maintaining time-varying formations. *IEEE Transactions on Robotics*, 25(5):1074–1086, 2009.

[214] P. Tabuada. Symbolic control of linear systems based on symbolic subsystems. *IEEE Transactions on Automatic Control*, 51(6):1003–1013, 2006.

[215] P. Tabuada. An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, 53(6):1406–1418, 2008.

[216] P. Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[217] N. N. Taleb. *Antifragile: Things that gain from disorder*, volume 3. Random House Incorporated, 2012.

[218] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3—a MATLAB software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

[219] R. Toscano. *Structured controllers for uncertain systems*. Springer, 2013.

[220] T. Trenner and J. Neidig. Einsatz cyber-physischer Systeme im Echtzeitkontext. In G. Reinhart, B. Scholz-Reiter, W. Wahlster, M. Wittenstein, and D. Zühlke, editors, *Intelligente Vernetzung in der Fabrik*, pages 197–212. Fraunhofer Verlag, Stuttgart, 9 2015.

[221] V. I. Utkin. *Sliding modes in control and optimization*. Springer Science & Business Media, 2013.

[222] J. Van Belzen, J. Van De Koppel, M. L. Kirwan, D. Van Der Wal, P. M. Herman, V. Dakos, S. Kéfi, M. Scheffer, G. R. Guntenspergen, and T. J. Bouma. Vegetation recovery in tidal marshes reveals critical slowing down under increased inundation. *Nature Communications*, 8:15811, 2017.

[223] A. J. Van Der Schaft and J. M. Schumacher. *An introduction to hybrid dynamical systems*. Springer London, 2000.

[224] C.-I. Vasile, D. Aksaray, and C. Belta. Time window temporal logic. *Theoretical Computer Science*, 691:27–54, 2017.

[225] J. Vehí, I. Ferrer, and M. Á. Sainz. A survey of applications of interval analysis to robust control. In *Proceedings of the 15th IFAC World Congress*, volume 35,

pages 389–400, Barcelona, Spain, 2002.

[226] S. Waldherr, R. Findeisen, and F. Allgöwer. Global sensitivity analysis of biochemical reaction networks via semidefinite programming. In *Proceedings of the 17th IFAC World Congress*, volume 41, pages 9701–9706, Seoul, Korea, 2008.

[227] E. Walter and H. Piet-Lahanier. Estimation of parameter bounds from bounded-error data: a survey. *Mathematics and Computers in simulation*, 32(5-6):449–468, 1990.

[228] J. Wan, O. Marjanovic, and B. Lennox. Disturbance rejection for the control of batch end-product quality using latent variable models. *Journal of Process Control*, 22(3):643–652, 2012.

[229] B. Wang, U. Manandhar, X. Zhang, H. B. Gooi, and A. Ukil. Deadbeat control for hybrid energy storage systems in dc microgrids. *IEEE Transactions on Sustainable Energy*, 10(4):1867–1877, 2018.

[230] Q. Wang, P. Ping, X. Zhao, G. Chu, J. Sun, and C. Chen. Thermal runaway caused fire and explosion of lithium ion battery. *Journal of Power Sources*, 208:210–224, 2012.

[231] L. C. Westphal. *Handbook of control systems engineering*, volume 635. Springer Science & Business Media, 2012.

[232] H. P. Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[233] T. Wongpiromsarn. *Formal methods for design and verification of embedded control systems: application to an autonomous vehicle*. PhD thesis, California Institute of Technology, 2010.

[234] T. Wongpiromsarn, U. Topcu, and A. Lamperski. Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems. *IEEE Transactions on Automatic Control*, 61(11):3344–3355, 2015.

[235] S. Yu, C. Maier, H. Chen, and F. Allgöwer. Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems. *Systems & Control Letters*, 62(2):194–200, 2013.

[236] G. Yurko. Automatic rise time adjustment for bi-level pressure support system, March 18 2003. US Patent 6,532,960.

[237] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2011.

[238] J. Zaytoon, V. Carré-Ménétrier, O. N. El Alaoui, and F. Gellot. A declarative framework for the characterisation of cyclic behaviour for a class of hybrid control systems. In *Proceedings of the 8th IFAC Symposium on Computer Aided Control Systems Design*, volume 33, pages 57–62, Salford, United Kingdom, 2000.

[239] K. Zhou and J. C. Doyle. *Essentials of robust control*. Prentice Hall Upper

Saddle River, New Jersey, 1998.

[240] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Transactions of the ASME*, 64:759–768, 1942.