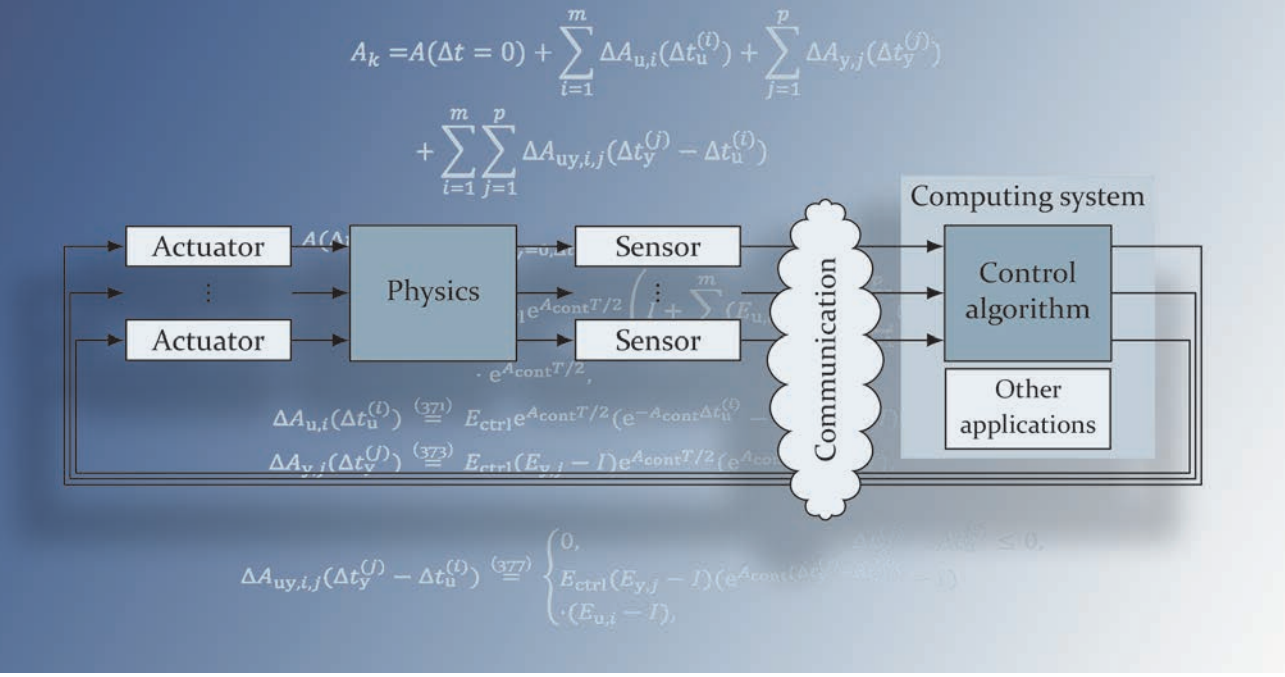


FAU Studien aus der Elektrotechnik 20

Maximilian Gaukler

Safety Verification of Real-Time Control Systems with Flexible Timing

FAU University Press

Maximilian Gaukler

Safety Verification of Real-Time Control Systems with Flexible Timing

FAU Studien aus der Elektrotechnik

Band 20

Herausgeber der Reihe:
Prof. Dr.-Ing. Bernhard Schmauß

Maximilian Gaukler

Safety Verification of Real-Time Control Systems with Flexible Timing

Erlangen
FAU University Press
2023

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Bitte zitieren als
Maximilian Gaukler. 2023. *Safety Verification of Real-Time Control Systems with Flexible Timing.* FAU Studien aus der Elektrotechnik Band 20. Erlangen. FAU University Press. DOI: 10.25593/978-3-96147-638-1.

Der vollständige Inhalt des Buchs ist als PDF über den OPUS-Server der Friedrich-Alexander-Universität Erlangen-Nürnberg abrufbar:
https://opus4.kobv.de/opus4-fau/home

Verlag und Auslieferung:
FAU University Press, Universitätsstraße 4, 91054 Erlangen

Druck: docupoint GmbH

ISBN: 978-3-96147-637-4 (Druckausgabe)
eISBN: 978-3-96147-638-1 (Online-Ausgabe)
ISSN: 2363-8699
DOI: 10.25593/978-3-96147-638-1

Safety Verification of Real-Time Control Systems with Flexible Timing

Sicherheitsverifikation von Echtzeitregelungssystemen mit flexiblem Timing

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg

zur Erlangung des Doktorgrades Dr.-Ing.
vorgelegt von

Maximilian Gaukler

Als Dissertation genehmigt von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:	17. November 2022
Gutachter:	Prof. Dr.-Ing. habil. Günter Roppenecker
	Prof. Dr.-Ing. Peter Ulbrich
	Prof. Dr.-Ing. Ulrich Konigorski

Danksagung

Meine Zeit als Doktorand am Lehrstuhl für Regelungstechnik war eine wunderbare Gelegenheit, mich über Jahre in ein Thema zu vertiefen und dabei weit mehr zu lernen als nur das fachliche Handwerkszeug. Ich blicke gerne darauf zurück, besonders auch auf viele persönliche Erinnerungen. Die wissenschaftlichen Erkenntnisse aus jener Zeit fließen nun in dieser Dissertation zusammen. Ich hoffe, dass sie neue Impulse für die weitere Forschung liefert.

An dieser Stelle möchte ich mich bei allen bedanken, die diese angenehme Zeit ermöglicht und dazu beigetragen haben. Ganz besonders bei meinem Doktorvater, Prof. Dr.-Ing. habil. Günter Roppenecker, der großes Vertrauen in meine Arbeit mitbrachte und mir weite Freiräume ließ. Für ein gutes Vorankommen sorgte er nicht nur mit dem passenden Rat, sondern auch mit der nötigen Finanzierung. Sein Nachfolger als Lehrstuhlinhaber, Prof. Dr.-Ing. Knut Graichen, ermöglichte dankenswerter Weise die Weiterführung meiner Arbeit. Herrn Prof. Dr.-Ing. Peter Ulbrich danke ich für seine Hilfe als Mentor und Forschungspartner aus der Informatik. Seine praktischen Ideen waren der Grundstein für viele der in dieser Arbeit vorgestellten theoretischen Konzepte.

Ebenso bedanke ich mich bei Prof. Dr.-Ing. Ulrich Konigorski für die Begutachtung der Arbeit. Prof. Dr.-Ing. Thomas Moor danke ich für seine Unterstützung im Bereich hybrider Systeme sowie für die Übernahme des Prüfungsvorsitzes. Des weiteren gilt mein Dank Prof. Dr.-Ing. Wolfgang Schröder-Preikschat für die Mitwirkung in der Prüfungskommission.

Bei den Kolleginnen und Kollegen am Lehrstuhl für Regelungstechnik und am Lehrstuhl für Informatik 4 bedanke ich mich für die persönlich und fachlich bereichernde Zeit. Stellvertretend etwa bei Dr.-Ing. Andreas Michalka, der stets eine offene Tür für meine Fragen hatte, sowie bei Tim Rheinfels, der mit viel mathematischem Scharfsinn neue Lösungen fand und zusammen mit Paulina Spenger beim Korrekturlesen des Manuskripts unterstützte. Viele weitere könnte ich noch nennen, zum Beispiel die Studentinnen und Studenten, die zu meiner Forschung beigetragen haben.

Abschließend gilt mein größter Dank meiner Familie und meinen Freunden. Sie haben mir so viel unbezahlbar Gutes gegeben und ich freue mich, jetzt wieder mehr Freizeit mit ihnen verbringen zu können.

Erlangen, im November 2022 *Maximilian Gaukler*

Abstract

Control loops are an essential part of many safety-critical systems, such as autonomous vehicles. Often, such controllers are sensitive to execution timing and are therefore considered real-time applications. In particular, the input and output timing is critical: If sensing or actuation do not occur exactly at the intended time, control performance can deteriorate up to instability.

Classically, controller design and analysis assume that the input and output timing is exactly periodic. The underlying computer system must ensure this exact timing. This requirement is not desirable because computing systems exhibit a general design conflict between predictable timing and cost-efficiency. Furthermore, the growing complexity of modern control systems leads to increasing timing variation, e.g., due to multi-core processors and networked sensors. This makes exact timing increasingly difficult to implement.

To overcome the burdensome requirement of exact timing in a control loop, research has proposed to accept a certain amount of timing variation and verify that this does not lead to unsafe behavior. The present thesis investigates the necessary *safety verification of real-time control systems with flexible timing*. The focus is on control-theoretical methods that lead to mathematically proven upper bounds on the worst-case behavior. The key contribution of this thesis is to consider the multiple-input-multiple-output (MIMO) case, where multiple sensors and actuators each have individual timing deviations. While this case necessarily occurs in modern, complex control systems, it has yet received little attention in the literature. Existing methods are generally restricted to two timing variables, e.g., simultaneous sampling of all sensors at one time and all actuators at a second time. A direct extension of existing work to the MIMO case is not possible because the increased number of timing variables leads to infeasible computational complexity.

The present thesis considers two problem settings: *Design-time analysis* determines the worst-case performance for a given, fixed time window for sensing and actuation. In contrast, *dynamic resource management* adjusts the permissible time window at run-time, which allows for larger short-term deviations as long as the "average" timing is good enough.

The linear case of design-time analysis is considered based on the discretization of a linear-impulsive system model. To reduce computational complexity, a novel splitting theorem breaks down the dynamics into summands that depend on at most two timing variables. Then, stability is shown by a norm bound and a common quadratic Lyapunov function. A numerical experiment validates that the method is feasible for the MIMO case as long as the system dimensions and timing deviations are not too large.

For the nonlinear case, the present thesis investigates *continuization*, a method that converts the partly discrete-time control loop into a purely continuous-time system. Existing work by Bak and Johnson, 2015 is put onto a new formal basis using abstractions of hybrid automata. This investigation uncovers that existing work is limited to static, such as proportional, controllers. A complementary variant, *first-order continuization*, is introduced that supports certain dynamic controllers, such as an integral controller. The method is illustrated by an experiment for the case of ideal timing, and an extension to MIMO timing deviations is outlined.

Finally, dynamic resource management is investigated. Here, the computation of timing decisions occurs at run-time, which makes the complexity arising from the MIMO case particularly challenging. The novel framework of *convergence rate abstractions* overcomes this difficulty by reducing the complex effect of timing on control performance to a simple, one-dimensional dynamic system, the *abstraction*. This system computes a guaranteed upper bound on the state of the control loop and can be used to make timing decisions at run-time by means of simple decision rules. An exemplary implementation shows that the method can make safe decisions for the MIMO case of flexible timing and has low computational overhead despite a large number of timing variables. This goes beyond the capabilities of existing methods. Simulations indicate that the approach is advantageous if the spread between average- and worst-case timing is sufficiently high.

Kurzzusammenfassung

Sicherheitsverifikation von Echtzeitregelungssystemen mit flexiblem Timing

Regelkreise sind ein Kernbestandteil vieler sicherheitskritischer Systeme, wie etwa autonomer Fahrzeuge. Häufig reagieren diese Regelungen empfindlich auf zeitliche Abweichungen in der Ausführung und werden dann als Echtzeitanwendung bezeichnet. Kritisch ist unter anderem das Ein-/Ausgabe-Timing: Wenn die Abtastung der Messwerte durch die Sensoren oder die Ausgabe der Stellsignale durch die Aktoren nicht exakt zum geplanten Zeitpunkt stattfinden, kann dies die Regelgüte bis hin zur Instabiliät verschlechtern.

Für den Entwurf und die Analyse des Reglers wird klassischerweise ein exakt periodisches Ein-/Ausgabe-Timing angenommen. Das zugrundeliegende Rechnersystem muss dieses Timing sicherstellen. Diese Anforderung erweist sich als problematisch, denn bei Rechnersystemen besteht ein Zielkonflikt zwischen vorhersagbarem Zeitverhalten und Kosteneffizienz. Zusätzlich führt die wachsende Komplexität moderner Regelungssysteme zu wachsender zeitlicher Unsicherheit, etwa durch Mehrkernprozessoren oder vernetzte Sensoren.

Als Abhilfe für diese Nachteile des exakten Zeitverhaltens wird in der Forschung vorgeschlagen, ein gewisses Maß an zeitlicher Unsicherheit zuzulassen und nachzuweisen, dass trotzdem kein unsicheres Verhalten auftritt. Die vorliegende Dissertation betrachtet die hierzu notwendige *Sicherheitsverifikation von Echtzeitregelungssystemen mit flexiblem Timing*. Der Schwerpunkt liegt auf regelungstechnischen Methoden, die obere Schranken für das schlimmst mögliche Verhalten nachweisen. Der wesentliche Beitrag der Dissertation ist die Erweiterung auf den Mehrgrößenfall, d. h. auf ein System mit mehreren Sensoren und Aktoren, welche jeweils eigene zeitliche Abweichungen aufweisen. Während dieser Fall in vielen modernen Regelungssystemen auftritt, hat er bisher in der Literatur nur wenig Beachtung gefunden. Bestehende Methoden sind im Allgemeinen auf zwei Timing-Variablen beschränkt; so wird etwa angenommen, dass alle Sensoren gleichzeitig an einem Zeitpunkt und alle Aktoren gleichzeitig an einem zweiten Zeitpunkt abgetastet werden. Eine direkte Erweiterung bestehender Methoden auf den Mehrgrößenfall ist nicht möglich, weil die größere Anzahl an Timing-Variablen zu einer unpraktikabel langen Berechnungsdauer führen würde.

Die vorliegende Dissertation betrachtet zwei Konzepte zum Umgang mit flexiblem Timing: Die *Analyse zum Entwurfszeitpunkt* ermittelt die schlimmste mögliche Regelgüte für ein gegebenes, festes Ein-/Ausgabe-Zeitfenster. Im Gegensatz dazu wird beim *dynamischen Ressourcenmanagement* das Zeitfenster zur Laufzeit angepasst, sodass kurzzeitig größere Timing-Abweichungen möglich sind, solange im „langfristigen Mittel" das Timing gut genug bleibt.

Für die Analyse zum Entwurfszeitpunkt wird zunächst ein Ansatz für den linearen Fall vorgestellt. Dabei wird ein lineares impulsives System diskretisiert und die Dynamik in Summanden aufgeteilt, die von höchstens zwei Timing-Variablen abhängen. Diese neuartige Zerlegung reduziert die Komplexität des Mehrgrößenfalls auf ein praktikables Maß. Abschließend wird die Stabilität mit einer gemeinsamen quadratischen Lyapunovfunktion nachgewiesen. Die Erprobung zeigt, dass der Ansatz im Mehrgrößenfall anwendbar ist, solange die Systemdimension und Timing-Unsicherheit nicht zu groß werden.

Für den nichtlinearen Fall wird die *Continuization*-Methode betrachtet, welche den teils zeitdiskreten Regelkreis in ein rein zeitkontinuierliches System umwandelt. Die bestehende Methode von Bak und Johnson, 2015 ist hierbei auf statische, z. B. proportionale, Regler beschränkt. Zur Weiterentwicklung wird eine neue formale Basis aufgestellt, basierend auf der Abstraktion hybrider Automaten. Auf dieser Grundlage wird eine neue Variante, die *Continuization erster Ordnung*, entwickelt, welche bestimmte dynamische, z. B. integrierende, Regler unterstützt. Dies wird für den Fall exakten Ein-/Ausgabe-Timings erprobt und eine Erweiterung auf den Mehrgrößenfall skizziert.

Den Abschluss der Arbeit bildet das dynamische Ressourcenmanagement. Hier ist die Komplexität des Mehrgrößenfalls besonders herausfordernd, weil die Berechnung der Timing-Entscheidung zur Laufzeit des Regelungssystems geschehen muss. Das vorgestellte Konzept der *Konvergenzraten-Abstraktion* überwindet diese Schwierigkeit, indem es den komplexen Zusammenhang zwischen Timing und Regelgüte auf ein einfaches eindimensionales dynamisches System, die *Abstraktion*, reduziert. Diese berechnet eine garantierte obere Schranke für den Zustand des Regelkreises. So können zur Laufzeit Timing-Entscheidungen mittels einfacher Regeln getroffen werden. Eine beispielhafte Umsetzung zeigt, dass der Ansatz vorteilhaft ist, solange der Abstand zwischen durchschnittlichem und schlimmstem Timing hinreichend groß ist. Die Methode kann im Mehrgrößenfall sichere Entscheidungen zum flexiblen Timing treffen und erfordert dabei nur geringen Rechenaufwand zur Laufzeit. Dies übersteigt die Fähigkeiten bestehender Methoden.

Contents

Symbols and Abbreviations

Abbreviations

Abbreviation	Description
ACET	average-case execution time
CGES	continuous-time globally uniform exponential stability (Definition 3.2.1 on page 53)
CPU	central processing unit
CQLF	common quadratic Lyapunov function
DGES	discrete-time globally uniform exponential stability (Definition 4.1.12 on page 67)
EISS	exponential input-to-state stability (Definition 6.4.1 on page 153)
HA	hybrid automaton (Definition 4.2.1 on page 73)
IO	input/output
LET	logical execution time
LIS	linear impulsive system (Definition 4.1.8 on page 60)
LMI	linear matrix inequality (Definition 5.1.20 on page 115)
LPV	linear parameter-varying
MIMO	multiple-input-multiple-output
ODE	ordinary differential equation
OS	operating system
PDE	partial differential equation
SH	sample-and-hold
SISO	single-input-single-output
SMT	satisfiability modulo theories
SOS	sum-of-squares
UAV	unmanned aerial vehicle
WCET	worst-case execution time
WOET	worst observed execution time

Important Symbols and Notation

General Notation

■	placeholder for explaining the notation
$f(\cdot)$	function f with omitted arguments
...	omission
Pr(■)	probability
const	constant
$\sum$	sum: $\sum_{i=a}^{b} X_i = X_a + X_{a+1} + \cdots + X_b$ if $b \geq a$, else 0
$\prod$	product: $\prod_{i=a}^{b} X_i = X_a X_{a+1} \ldots X_b$ if $b \geq a$, else I
$\widetilde{\prod}$	reversed product: $\widetilde{\prod}_{i=a}^{b} X_i = X_b X_{b-1} \ldots X_{a+1} X_a$ if $b \geq a$, else I (Definition 4.1.9 on page 61)
■$^{+}$	right-side limit: $f(a^{+}) \coloneqq \lim_{x \to a,\ x>a} f(x)$
■$^{-}$	left-side limit: $f(a^{-}) \coloneqq \lim_{x \to a,\ x<a} f(x)$
■$\vert_{a=b}$	substitution
$\exists x :$ ■	"There exists at least one x such that ..."
$\forall x$ ■	"For all x it holds that ..."

Fonts

aA	*upright:* operators, basic functions
a	*italic lowercase:* scalar and vector variables and functions
A	*italic uppercase:* variables and functions, mostly matrices
$\mathcal{A}$	*calligraphic:* sets and set-valued functions
$\mathbb{A}$	*blackboard bold:* basic sets
■$_{a}$	*italic index:* refers to variable, e.g., $c_i = c_2$ if $i = 2$
■$_{\mathrm{a}}$	*upright index:* refers to name, e.g., A_{p} is the *plant* dynamics matrix and does not depend on the variable p

Scalar Operations

$\blacksquare!$	faculty: $3! = 3 \cdot 2 \cdot 1$
$\lfloor\blacksquare\rfloor$	rounding towards negative infinity: $\lfloor x \rfloor \coloneqq \max\{z \in \mathbb{Z} \mid z \leq x\}$
mod	remainder of division: $x \bmod y \coloneqq x - \left\lfloor \frac{x}{y} \right\rfloor y$ for $x, y > 0$
sat	saturation function: (299) on page 188

Vector Operations

$(\blacksquare, \blacksquare)$	tuple, considered equivalent to a column vector
$\blacksquare^{(i)}$	i-th component
$\lvert\blacksquare\rvert$	Euclidean norm: $\lvert x \rvert = \sqrt{x^\top x}$
$\partial\blacksquare/\partial x$	Jacobian matrix: (165) on page 123

Matrix Operations

$\blacksquare_{a \times b}$	annotated dimension: matrix is in $\mathbb{R}^{a \times b}$
$\blacksquare^\top$	transposition
$\blacksquare_{(i,j)}$	entry at i-th row and j-th column
$\lVert\blacksquare\rVert$	norm, mainly: matrix norm (Definition 4.1.1 on page 58)
$\lVert\blacksquare\rVert_2$	spectral norm (Definition 4.1.6 on page 59)

Operations with Square Matrices

$\lVert\blacksquare\rVert_P$	P-ellipsoidal norm (Definition 5.1.5 on page 104)
$\lVert\blacksquare\rVert_P^{\mathrm{ub}}$	upper bound on P-ellipsoidal norm (page 114)
$\blacksquare^{-\top}$	transposed inverse
$\blacksquare^{1/2}$	Cholesky Decomposition: $P = P^{1/2}(P^{1/2})^\top$ (Theorem 5.1.3 on page 103)
diag	diagonal matrix with given entries
det	determinant
e$^{\blacksquare}$	matrix (or scalar) exponential
$\lambda_i\{\blacksquare\}$	i-th eigenvalue, $i = 1, \ldots, n$, in arbitrary order
$\rho\{\blacksquare\}$	spectral radius: $\max_i \lvert\lambda_i\{\blacksquare\}\rvert$

Basic Sets

Let $a, b, r \in \mathbb{R}$ with $a \leq b$ and $r \geq 0$.

$f : \mathcal{X} \mapsto \mathcal{Y}$	function f from $\mathcal{X}$ to $\mathcal{Y}$
$\mathbb{B}_r$	closed ball of radius r: $\{x \in \mathbb{R}^n \mid \lvert x\rvert \leq r\}$
$\mathbb{R}$	real numbers
$\mathbb{R}_{\geq 0}$	nonnegative numbers
$\mathbb{N}_0$	$\{0, 1, 2, \dots\}$
$\mathbb{N}_1$	$\{1, 2, \dots\}$
$\mathbb{Z}$	$\{\dots, -2, -1, 0, 1, 2, \dots\}$
$\{x \mid f(x)\}$	set of x for which the condition $f(x)$ is true
$\{a, b\}$	set containing two elements a and b
$(a; b)$	open interval; not to be confused with the tuple (a, b)
$[a; b)$	half-open interval, including a but not b
$[a; b]$	closed interval

Operations on Sets

Let $\mathcal{X}, \mathcal{Y}$ be appropriate sets, $A \in \mathbb{R}^{n \times n}$ and $c \in \mathbb{R}$.

$\times$	Cartesian product: $\mathcal{X} \times \mathcal{Y} \coloneqq \{(x, y) \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$
$\mathcal{X}^i$	Cartesian power: $\mathcal{X}^3 \coloneqq \mathcal{X} \times \mathcal{X} \times \mathcal{X}$
$\setminus$	set difference: $\mathcal{X} \setminus \mathcal{Y} \coloneqq \{x \in \mathcal{X} \mid x \notin \mathcal{Y}\}$
$\oplus$	Minkowski sum: $\mathcal{X} \oplus \mathcal{Y} \coloneqq \{x + y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$
$\otimes$	set-valued product: $\mathcal{X} \otimes \mathcal{Y} \coloneqq \{xy \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$
$A\mathcal{X}$	set-valued matrix multiplication: $A\mathcal{X} \coloneqq \{Ax \mid x \in \mathcal{X}\}$
$c\mathcal{X}$	scaling of set: $c\mathcal{X} \coloneqq \{cx \mid x \in \mathcal{X}\}$
$\lvert\mathcal{X}\rvert$	number of elements in set
$\square\mathcal{X}$	enclosing closed multidimensional interval
conv	convex hull: $\operatorname{conv}\mathcal{X} \coloneqq \left\{ \sum_{i=1}^{n} \alpha_i x_i \mid x_i \in \mathcal{X}, \quad \alpha_i \in [0; 1], \quad \sum_{i=1}^{n} \alpha_i = 1, \quad n \in \mathbb{N}_1 \right\}$
Approx	outer approximation: $\operatorname{Approx}\mathcal{X} \supseteq \mathcal{X}$, typically $\operatorname{Approx}\mathcal{X} \approx \mathcal{X}$.

Operations on Hybrid Automata

Let H, H_1, H_2 be hybrid automata and $t_1, t_2 \in \mathbb{R}$ with $t_2 > t_1$.

$H_1 \parallel H_2$	Composition (Definition 4.2.3 on page 78)
$\mathrm{Reach}(H)$	Set of reachable (location, state)-tuples (Definition 4.2.4 on page 81)
$\mathrm{Reach}_x(H, t_1)$	Set of reachable x at time t_1 (Definition 4.2.6 on page 81)
$\mathrm{Reach}_x(H, [t_1; t_2])$	Set of reachable x in time range $[t_1; t_2]$ (Definition 4.2.7 on page 81)

Relations

$\overset{(123)}{\square}$	relation holds due to equation (123)
$:\square$	relation holds by definition; "is defined as"
$:=$	*in equations:* equal by definition. The left-hand-side is a new variable; the right-hand-side is an already defined expression. *in algorithms:* set a variable, possibly overwriting the old value.
$=:$	in equations: $b =: a$ means $a := b$. Herein, a is the new variable, and b an already defined expression.
$\overset{\text{lin}}{=}$	equal in the case of linear system dynamics
$\equiv$	equal for all time
$\prec$	negative definite (Definition 5.1.2 on page 102)
$\trianglelefteq$	tight upper bound (Definition 6.5.2 on page 157)
$\subset$	proper subset
$\subseteq$	proper subset or equal
$\overset{\alpha}{\longrightarrow}$	transition with label α (Definition 4.2.1 on page 73)
$\to$	goes to in the limit ($x_k \to x_\infty$ for $k \to \infty$)
$=_x$	x-equivalence of hybrid automata (Definition 4.2.8 on page 82)
$\subseteq_x$	x-abstraction of hybrid automata (Definition 4.2.9 on page 87)

The mirror image of any relation denotes swapped arguments, e.g., $A \supseteq_x B :\Leftrightarrow B \subseteq_x A$. Mirror-symmetric shapes denote symmetric relations, e.g., $A =_x B \Leftrightarrow B =_x A$.

Time Arguments

- $\square_k$ discrete time argument (may be omitted)
- $\square(t)$ continuous time argument (may be omitted)
- $\square(t^-)$ left limit, "just before"
- $\square(t^+)$ right limit, "just after"
- $\dot{\square}$ time derivative
- $\square'$ successor state of transition (Definition 4.2.1 on page 73)

Common Subscripts

- $\square_0$ initial value
- $\square_{\mathrm{ctrl}}$ controller computation event
- $\square_{\mathrm{d}}$ discrete-time dynamics, including controller
- $\square_{\mathrm{dist}}$ disturbance
- $\square_{\mathrm{p}}$ continuous-time plant dynamics
- $\square_{\mathrm{u}}$ actuation event
- $\square_{\mathrm{y}}$ measurement (sampling) event

Annotations for Set Variables $\mathcal{X}_{\mathrm{p}}, \mathcal{X}_{\mathrm{pd}}, \Delta, \Omega$ in Continuization

- $\underline{\square}$ initial guess for bound
- $\square$ enlarged version of $\underline{\square}$
- $\square'$ bound resulting from analysis, should be within $\underline{\square}$

Important Variables and Functions

Name	Type	Description	Page
A_d	$\in \mathbb{R}^{n_\mathrm{d} \times n_\mathrm{d}}$	controller dynamics matrix (6)	42
A_p	$\in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}$	plant dynamics matrix (3)	41
A_cont	$\in \mathbb{R}^{n \times n}$	continuous-time dynamics matrix (Definition 4.1.8, Section 4.1.3)	60, 62
$A(\sigma_k)$	$\in \mathbb{R}^{n \times n}$	timing-dependent discrete transition matrix	64
A_k	$= A(\sigma_k)$		64
$A(\Delta t)$	$= A(\sigma_k)$	same as $A(\sigma_k)$, assuming no skips	64
ΔA	$\in \mathbb{R}^{n \times n}$	summand of timing-dependent transition matrix (Theorem 5.1.4, Equations (115) to (118))	104, 108
$\mathcal{A}$	$\subset \mathbb{R}^{n \times n}$	set of transition matrices $A(\Delta t)$ within timing bounds, without skips	64
b_i	$\in \{0, 1\}$	binary digit (Lemma B.5)	213
B_d	$\in \mathbb{R}^{n_\mathrm{d} \times p}$	controller input matrix (6)	42
B_p	$\in \mathbb{R}^{n_\mathrm{p} \times m}$	plant input matrix (3)	41
C_d	$\in \mathbb{R}^{m \times n_\mathrm{d}}$	controller output matrix (6)	42
C_p	$\in \mathbb{R}^{p \times n_\mathrm{p}}$	plant output matrix (3)	41
C_0	$\in \mathbb{R}^{n \times n_0}$	shape matrix of initial set (213c)	150
C_s	$\in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{p}}$	safety output matrix (25)	53
$\bar{C}_\mathrm{s}$	$\in \mathbb{R}^{n_\mathrm{s} \times n}$	safety output matrix for extended state (75)	71
$d(t)$	$\in \mathbb{R}^{n_\mathrm{dist}}$	generalized disturbance (3)	41
d_k	$\in \mathbb{R}^{n_\mathrm{dist}}$	discretized generalized disturbance	69
$\lvert d \rvert_\mathrm{max}$	$\in \mathbb{R}_{\geq 0}$	maximum disturbance amplitude (276)	172
$\lvert d_k \rvert_\mathrm{max}$	$\in \mathbb{R}_{\geq 0}$	time-varying maximum disturbance amplitude (213)	150

Name	Type	Description	Page
$\mathcal{D}$	$\subset \mathbb{R}^{n_{\text{dist}}}$	bounded set of disturbance values (5)	42
e	≈ 2.718	Euler's number	
e_i	$\in \mathbb{R}^{\blacksquare}$	i-th base vector $\begin{bmatrix}0 & \dots & 0 & 1 & 0 & \dots & 0\end{bmatrix}^\top$	
$E_{\blacksquare}$	$\in \mathbb{R}^{n \times n}$	event matrix (Definition 4.1.8, Section 4.1.3)	60, 62
f_ω	function	dynamics of continuization error (177)	129
f_{p}	function	plant dynamics. Mainly: (3),	41
		Section 5.2.4: (174),	126
		Section 5.2.5: (193)	133
f_{d}	function	controller dynamics. Mainly: (6),	42
		Section 5.2.4: (174),	126
		Section 5.2.5: (193)	133
f_{pd}	function	dynamics of joint state x_{pd} (Section 5.2.5, Figure 34)	135
$flow_i$	function	continuous dynamics (flow) (Definition 4.2.1)	73
$\mathcal{F}_{\blacksquare}$	function	set-valued bound for $f_{\blacksquare}$ (Section 5.2)	127, 135
$guard_{i,j}$	function	guard condition of HA (Definition 4.2.1)	73
g_{d}	function	controller output function (6)	42
g_{p}	function	plant output function (3)	41
G_{p}	$\in \mathbb{R}^{n_{\text{p}} \times n_{\text{dist}}}$	plant disturbance input matrix (3)	41
$G(\sigma_k)$	$\in \mathbb{R}^{n \times n_{\text{dist}}}$	discretized disturbance input matrix	69, 71
H_{p}	$\in \mathbb{R}^{p \times n_{\text{dist}}}$	plant measurement noise matrix (3)	41
$H_{\blacksquare}$	HA	hybrid automaton	
I	$\in \mathbb{R}^{n \times n}$	identity matrix	

Name	Type	Description	Page
inv_i	function	invariant of HA (Definition 4.2.1)	73
i	$\in \mathbb{N}_0$	event counter in LIS	60
i, j	$\in \mathbb{N}_0$	generic counter	
IA(k)		induction assumption in proof	
$\mathcal{J}nit$	$\subseteq \mathcal{L}oc \times \mathbb{R}^n$	initial location-state-pairs of HA (Definition 4.2.1)	73
k	$\in \mathbb{N}_0$	discrete time; generic counter;	
		k-th control period (Definition 3.1.2)	46
K	$\in \mathbb{N}_1$	window length for (M, K) skip ratio (Definition 3.1.1)	46
l	$\in \mathbb{N}_0$	generic counter	
$\mathcal{L}oc$	set	locations of hybrid automaton (Definition 4.2.1)	73
m	$\in \mathbb{N}_1$	number of actuators = dimension of u	41
$\bar{m}$	$= K - M$	maximum skips for (M, K) skip ratio (267b)	163
M	$\in \mathbb{N}_1$	minimum executions for (M, K) skip ratio (Definition 3.1.1)	46
$M_{1,2}$	$\in \mathbb{R}^{n \times n}$	factors in general form (378) of ΔA	219
n_0	$\in \mathbb{N}_1$	dimension of normalized initial state $\tilde{x}_0$	150
n_d	$\in \mathbb{N}_1$	dimension of controller state x_d	42
n_s	$\in \mathbb{N}_1$	dimension of safety output s	53
n	$\in \mathbb{N}_1$	dimension of combined state x	47
		Sections 4.2 and 5.2 and Appendix C: dimension of continuous state vector x of HA	73
N	$\in \mathbb{N}_0$	general number of elements or events	

Name	Type	Description	Page
$N_{\mathrm{ev},k}$	$\in \mathbb{N}_0$	number of events in period k (Section 4.1.4)	66
p	$\in \mathbb{N}_1$	number of sensors = dimension of y	41
P	$\in \mathbb{R}^{n\times n}$	positive definite matrix for candidate Lyapunov function	103
q	function	state transformation for continuous equivalence	
		zero-order continuization: (178)	129
		first-order continuization: Figure 34	135
R	$\in \mathbb{R}^{n\times n}$	preconditioning matrix (Appendix B.3.2)	221
s_k	$\in \mathbb{R}^{n_\mathrm{s}}$	safety-relevant output (26)	53
$\lvert s\rvert_{\max}$	$\in \mathbb{R}_{\geq 0}$	permissible bound of safety output (26)	53
$\lvert s_k\rvert_{\mathrm{worst}}$	$\in \mathbb{R}_{\geq 0}$	worst-case result for safety output (296)	180
S_i	$\in \mathbb{R}^{\square\times\square}$	selector matrix $e_i e_i^\top$ (42)	63
t	$\in \mathbb{R}$	absolute time,	
		exception: witness $\xi(t)$ in HA uses relative time (Definition 4.2.1)	73
T	$\in \mathbb{R}_{>0}$	sampling period	42
$t_{\mathrm{b},k}$	$\in \mathbb{R}$	timing barrier (11)	44
$t_{\mathrm{ctrl},k}$	$\in \mathbb{R}$	event time for controller update (44)	64
$t_{\mathrm{u},k}$	$\in \mathbb{R}^m$	event times for actuation (19)	48
$t_{\mathrm{y},k}$	$\in \mathbb{R}^p$	event times for measurement (18)	48
$trans_{i,j}$	function	transition function (Definition 4.2.1)	73
u_k	$\in \mathbb{R}^m$	discrete-time actuator signal (6)	42

Name	Type	Description	Page
$u(t)$	$\in \mathbb{R}^m$	continuous-time actuator signal (8)	43
v_k	$\in \mathbb{R}_{\geq 0}$	abstraction state (Definition 6.2.2)	151
$V_P(x)$	$\in \mathbb{R}_{\geq 0}$	Lyapunov candidate function $x^\top P x$ (105)	103
$x(t)$	$\in \mathbb{R}^n$	combined state (17)	47
$x_\mathrm{d}(t), x_{\mathrm{d},k}$	$\in \mathbb{R}^{n_\mathrm{d}}$	controller state	42
$\tilde{x}_\mathrm{d}(t)$	$\in \mathbb{R}^{n_\mathrm{d}}$	interpolated controller state (Section 5.2.5, (196))	133
$x_{\mathrm{d,cont}}(t)$	$\in \mathbb{R}^{n_\mathrm{d}}$	continuous approximation of controller state (Section 5.2.5, (197))	133
$x_\mathrm{p}(t)$	$\in \mathbb{R}^{n_\mathrm{p}}$	physical plant state (3)	41
$x_\mathrm{pd}(t)$	$\in \mathbb{R}^{n_\mathrm{p}+n_\mathrm{d}}$	joint state of plant and continuized controller (Theorem 5.2.10)	136
$x_{\blacksquare,0}$	$\in \mathbb{R}^{\blacksquare}$	initial value of $x_{\blacksquare}$	
$\tilde{x}_0$	$\in \mathbb{R}^{n_0}$	normalized initial state (213c)	150
$\tilde{x}$	$\in \mathbb{R}^{\blacksquare}$	transformed continuous state vector of hybrid automaton	
$\lvert\tilde{x}_0\rvert_\mathrm{max}$	$\in \mathbb{R}_{\geq 0}$	maximum amplitude of normalized initial state (213c)	150
$\mathcal{X}_{\blacksquare}$	$\subset \mathbb{R}^{\blacksquare}$	set-valued bound for $x_{\blacksquare}$ (Section 5.2)	
$\mathcal{X}_{\mathrm{p},0}$	$\subset \mathbb{R}^{n_\mathrm{p}}$	bounded initial set of x_p (4)	42
$y(t)$	$\in \mathbb{R}^p$	continuous-time measurement (3)	41
y_k	$\in \mathbb{R}^p$	sampled measurement (9)	43
$y_\mathrm{d}(t)$	$\in \mathbb{R}^p$	sample-and-hold measurement (18)	48

Name	Type	Description	Page
α	$\in [1;\infty)$	overshoot factor	
		continuous time: Definition 3.2.1	53
		discrete time: Definition 4.1.12	67
		abstraction: Equation (216a)	151
$\alpha_{i,j}$	$\notin \mathbb{R}$	discrete transition label in HA (Definition 4.2.1)	73
β_i		transition label in trajectory of HA	73
β_σ	$\in \mathbb{R}_{>0}$	disturbance gain of abstraction (Definition 6.2.2)	151
δ	$\in \mathbb{R}_{\geq 0}$	duration; relative time	
$\delta_{\mathrm{p}}(t)$	$\in \mathbb{R}^{n_{\mathrm{p}}}$	sampling error of x_{p} (Section 5.2.5)	135
$\delta_{\mathrm{d}}(t)$	$\in \mathbb{R}^{n_{\mathrm{p}}}$	approximation error of $\tilde{x}_{\mathrm{d}}$ (Section 5.2.5)	135
$\delta_{\mathrm{pd}}(t)$	$\in \mathbb{R}^{n_{\mathrm{p}}+n_{\mathrm{d}}}$	deviation of x_{pd} from last sample (Theorem 5.2.10)	136
Δ	$\subset \mathbb{R}^{n_{\mathrm{p}}+n_{\mathrm{d}}}$	set-valued bound for δ_{pd} (Theorem 5.2.10)	136
$\Delta_{\mathcal{A}}$	$\subset \mathbb{R}^{n\times n}$	Set of $\Delta A_{\blacksquare}$ in Theorem 5.1.4	116
Δt_k	$\in \left(-\frac{T}{2};\frac{T}{2}\right)^{m+p}$	timing deviation (14)	47
$\Delta t_{\mathrm{u},k}$	$\in \left(-\frac{T}{2};\frac{T}{2}\right)^{m}$	actuator timing deviation (8)	43
$\Delta t_{\mathrm{y},k}$	$\in \left(-\frac{T}{2};\frac{T}{2}\right)^{p}$	sensor timing deviation (9)	43
$\overline{\Delta t}_{\blacksquare}$	$\in \left(-\frac{T}{2};\frac{T}{2}\right)$	upper timing bound (10)	43
$\underline{\Delta t}_{\blacksquare}$	$\in \left(-\frac{T}{2};\frac{T}{2}\right)$	lower timing bound (10)	43
λ	$\in \mathbb{R}$	stability exponent (Definition 3.2.1), eigenvalue	53
ρ	$\in \mathbb{R}_{\geq 0}$	discrete-time stability factor (Definition 4.1.12), spectral radius	67
$\tilde{\rho}$	$\in \mathbb{R}_{\geq 0}$	Section 5.1: bound on ρ for uncertain timing (140)	114
$\tilde{\rho}_{\mathrm{approx}}$	$\in \mathbb{R}_{\geq 0}$	Section 5.1.8: approximate bound on ρ for uncertain timing	118

Name	Type	Description	Page
ρ_0	$\in \mathbb{R}_{\geq 0}$	Section 5.1: bound on ρ for nominal timing (Theorem 5.1.6)	105
ρ_σ	$\in \mathbb{R}_{>0}$	Chapters 6 and 7: discrete-time stability factor in abstraction (Definition 6.2.2)	151
σ_k	$\in \Sigma$	execution of period k:	
		Section 2.5: generalized timing	29
		Chapters 3 to 5: timing and skips per (16),	47
		Chapters 6 and 7: generalized execution per Section 6.2	149
$\sigma_{\mathrm{ctrl},k}$	$\in \{0,1\}$	skip controller in period k	45
$\sigma_{\mathrm{skip},k}$	$\in \{0,1\}^{m+p+1}$	skipped events in period k (15)	47
$\sigma_{\mathrm{u},k}$	$\in \{0,1\}^{m}$	skipped actuators in period k	45
$\sigma_{\mathrm{y},k}$	$\in \{0,1\}^{p}$	skipped sensors in period k	45
Σ	set	range for generalized execution σ_k; see there	
$\hat{\Sigma}_k$	$\subseteq \Sigma$	permitted execution range for upcoming period k	173
τ_i	$\in \mathbb{R}$	event times of LIS (Definition 4.1.8)	60
$\tau(t)$	$\in \mathbb{R}$	timer state in hybrid automata	
ξ	function	witness of continuous transition in HA (Definition 4.2.1)	73
$\omega(t)$	$\in \mathbb{R}^{n_\mathrm{d}}$	approximation error of zero-order continuization (Section 5.2.4)	127
Ω	$\subset \mathbb{R}^{n_\mathrm{d}}$	set-valued bound for ω (Section 5.2.4)	126

1 Introduction

From coffee machines to cars, the standard of living we enjoy today would not be possible without feedback control. To improve safety, performance, efficiency and comfort, increasingly autonomous and complex systems are being developed, notably self-driving cars and autonomous aircraft.

Among other factors, the societal acceptance of these developments hinges on trustworthiness and cost (Maurer et al., 2016, Chapter 29). For example, customers will only buy a self-driving car if they can afford it and believe it drives safely. Engineering such safety-critical autonomous systems is challenging due to the unprecedented combination of complexity, safety requirements and cost pressure (Maurer et al., 2016, Chapter 23). Existing systems do not achieve this combination to the extent needed for a self-driving car: Today's driver assistance systems are less critical because they largely rely on human supervision. As a further example, fly-by-wire control systems for aircraft are highly safety-critical but are generally less complex and more expensive.

To achieve the seemingly impossible combination of low cost, high complexity and strict safety requirements, research has considered changes to many aspects of the established design process. This doctoral thesis addresses one such aspect: the input/output timing of the control loop due to the underlying real-time computing system.

Real-Time Control Figure 1 shows the hardware architecture of a modern computer-based feedback control system: The control algorithm is executed by a computing system that is shared with other applications, e.g., other controllers. One or multiple communication networks connect the computing system with the sensors and actuators, which interface to the physical world.

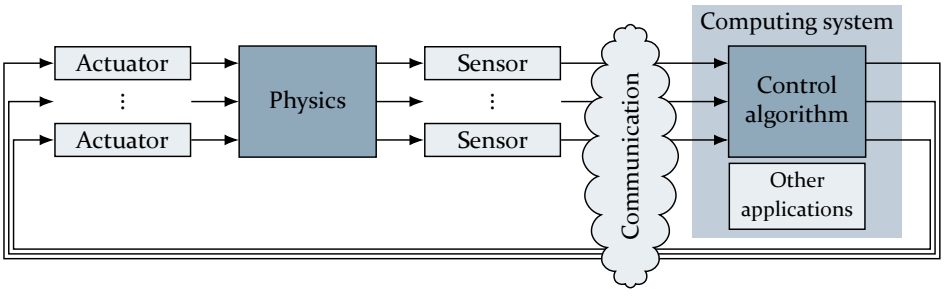


Figure 1: Hardware architecture of a modern computer-based feedback control system

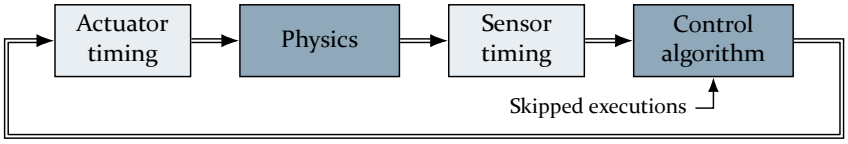


Figure 2: Timing effects in a computer-based feedback control system

Classical control theory assumes that the control algorithm instantaneously acts on the physical world. In practice, however, the temporal behavior of the control loop depends on details of the sensors, actuators, communication network and computing system. In particular, shared resources can lead to interference, e.g., other applications running on the same computing system can influence the execution timing of the controller. Figure 2 summarizes the effects on the control loop: The actuation and sensing times may deviate from the intended point. Also, overload of the computing system or communication network can lead to skipped executions of the control algorithm.

Such timing deviations can deteriorate the performance of control systems and even lead to instability (Marti, Villa, et al., 2001). Control loops in which the timing requires special consideration are commonly termed *real-time control*. One example thereof is the lane-keeping control of a vehicle. Here, a delayed response can lead to nervous driving or even a crash, analogous to a sleepy human driver with increased reaction time.

Traditionally, the timing of sensors, controller and actuators in a real-time control system is required to be precisely periodic. This ideal timing simplifies control-theoretical verification but shifts the burden towards the design and verification of the computing platform.

However, such precise timing is challenging to ensure, particularly in modern computing platforms: To increase performance and cost efficiency, computing systems are becoming more complex but also less predictable with regard to timing. This is exemplified by the trend towards multi-core or distributed real-time systems, e.g., with networked smart sensors. The next chapter will give further background and show that, ultimately, there is a trade-off between strict timing and a cost-efficient computing platform. At the same time, experiments and industrial practice point out that some imprecision in timing is acceptable (Akesson et al., 2020; Cervin, 2003).

Towards Flexible Timing In summary, the requirement of ideal timing is difficult to ensure, leads to inefficient use of hardware and is more stringent than necessary. Consequently, there is a large body of research on more flexible timing. These works have suggested modifications to the computing system, the controller or both. Approaches range from analysis and optimization at design time to dynamic adaptation at run-time (*dynamic resource management*). A detailed discussion will follow in the next chapter.

From a control-theoretic point of view, the fundamental necessity for any relaxation of the timing constraints is *safety verification*, i.e., to prove that the modified timing can not lead to unacceptable behavior.

Despite the large variety of existing work, one aspect has yet received little attention in the literature: Modern control systems are often multiple-input-multiple-output (MIMO) systems consisting of multiple sensors and actuators. Each sensor and actuator generally has an *individual* timing deviation, e.g., a vehicle controller may receive acceleration and velocity from separate sensors over different communication paths. However, as discussed later, there is no known method to analyze this MIMO case of uncertain timing.

Scope of this Work This doctoral thesis aims to close that gap, i.e., to answer the following research question: *How can the safety of real-time control systems with multiple inputs and outputs be verified while avoiding unnecessarily strict timing requirements on the computing platform?*

Real-time control systems are a wide-ranging topic. The focus of this work is mainly on a detailed control-theoretic investigation, whereas the computing system is considered at a more abstract level. Implementation details of the latter, such as scheduling algorithms, therefore remain out of scope.

This work follows the view that "because of the impact that they can have on the real world, [real-time control systems] deserve proofs as safety evidence" (Platzer, 2018). Testing and simulation can not provide this proof due to the infeasibly large, often infinite, number of possible trajectories. Therefore, core functions, such as control loops, should be mathematically proven (Clarke et al., 2018; Platzer, 2018). Consequently, this thesis employs proofs and upper bounds. This approach leads to well-founded results, though at the inevitable cost of pessimism and certain simplifying assumptions.

Previous Publications For many parts of this thesis, previous versions have been published in conference proceedings and technical reports. Details will follow later in Section 2.7.

Contents This doctoral thesis is divided into eight chapters: First, Chapter 2 details the background, related work and contribution. Then, Chapter 3 formalizes the system model and problem setting. In Chapter 4, this system model is rewritten in two mathematical frameworks: linear impulsive systems (LIS) and hybrid automata (HA).

Next, Chapter 5 presents two techniques for the design-time analysis of stability and safety of MIMO real-time control systems with bounded timing deviations. A first technique is based on the discretization of a LIS. A novel decomposition makes the *stability analysis* for the linear MIMO case computationally feasible, overcoming the limitations of previous work. A second technique is based on *continuization*, a transformation method for HA that is a counterpart to discretization. Here, theoretical progress is made towards determining *state bounds* in the nonlinear MIMO case.

The subsequent Chapters 6 and 7 consider *dynamic resource management*. The idea is that timing bounds are dynamically widened at run-time to the extent permitted by given state bounds and the influence of previous timing. Implementing the idea proves difficult, as the technique must provide provable safety but still be fast enough to make decisions at run-time. The complexity of the MIMO scenario makes the problem particularly challenging.

As a solution, Chapter 6 introduces the novel concept of *convergence rate abstractions*. The key idea is to reduce the control loop to a one-dimensional dynamical system, the *abstraction*, which is used to compute the permissible timing bounds at run-time. Chapter 7 implements this idea for the MIMO case of timing deviations, again overcoming the limitations of previous work.

Finally, Chapter 8 summarizes the results and looks out on future research.

2 State of the Art and Contribution

To give background on the rationale and to point out the focus of this work, this chapter discusses the aspects of safety, control and computation in more detail (Sections 2.1 to 2.3). Then, the interaction of control and computation is considered, including existing ideas for a joint design of both aspects (Section 2.4) and related work on stability analysis (Section 2.5). Finally, Section 2.6 presents the scope and contribution of this doctoral thesis, and Section 2.7 references previous publications of the results.

2.1 Safety Verification

Safety is the "absence of unreasonable risk". This definition is given in the automotive standard ISO 26262, whose terminology will be used in the following to discuss the scope and context of this doctoral thesis.

The overall system safety can be divided into multiple aspects (Miller, 2020): *Functional safety* is concerned with malfunctions of electrical and electronic systems, such as sensor failures due to aging. The *safety of the intended function* covers the behavior without malfunctions, e.g., performance limitations of camera-based sensors at night. Among further aspects of safety is the *physical safety*, e.g., flammability or sharp edges.

Within the context of functional safety, ISO 26262 distinguishes two classes of failures: *Systematic failures* have a clear cause in the design or production process, e.g., a software bug or the wrong choice of material. In contrast, *random failures* are best described by failure rates, e.g., sporadic wrong computations due to cosmic radiation.

A goal of this doctoral thesis is to prove the absence of systematic failures resulting from the timing variation of the computing platform. Practically, this means that the state must not exceed certain bounds for specified worst-case disturbance and timing.

Verification and Validation A typical workflow of safety engineering starts with the assessment of hazards to generate safety requirements. These requirements are addressed in the design and implementation of the system components and then checked by verification and validation (Walden et al., 2015).

Safety *verification* determines "whether or not an examined object meets its specified requirements" (ISO 26262). Typical activities range from human review of the design to testing, simulation and analysis. Such analysis for safety verification is the topic of this doctoral thesis. Safety *validation* ensures, "based on examination and tests, that the safety goals are adequate and have been achieved with a sufficient level of integrity" (ISO 26262).

As an informal summary, *verification* ensures that the product meets the defined requirements, and *validation* ensures that these requirements and the development process are strict enough to achieve safety.

Computer-Aided Safety verification In general, many verification tasks are suitable for computer-based formal analysis, while most validation tasks require human intelligence. As an example, consider a lane-keeping assistance system for a car with the requirement that up to 50 km/h of side wind shall not cause more than 30 cm of deviation from the lane. A typical verification question is whether the requirement is satisfied. Given an appropriate system model and specification, this question is mathematically well-defined and can be solved by computer algorithms. In contrast, the validation question "Is this requirement sufficient to avoid accidents?" is too vague to be analyzed by a computer. As this doctoral thesis aims at mathematically proven results, only verification is considered further.

Formal Verification To allow for automatic verification, the model and specification must be unambiguous and machine-readable. This is termed *formal notation*, defined as a description that "has both its syntax [(notation)] and semantics [(meaning)] completely defined" (ISO 26262). For example, a natural-language specification is informal, while mathematical equations and their computer representation are formal notation. Techniques for computing whether a formal model satisfies a formal specification are subsumed under the term *formal verification* (ISO 26262). Within this doctoral thesis, the term is defined narrowly as follows:

Definition 2.1.1 (Formal Verification)**.** Given a system model and a safety specification, a *formal verification* algorithm returns *safe*, *unsafe* or *unknown*. The result must be provably correct, i.e., *safe* is only returned if there is a mathematical safety proof and *unsafe* only if there is a counterexample in which the model violates the specification. The algorithm must return *unknown* if no conclusion can be reached. Furthermore, it must be able to show safety for some systems, i.e., not only return *unknown* or *unsafe*. ◁

This excludes *falsification* techniques that can only prove the unsafe case, as well as *unsound* approximative techniques without mathematical guarantees. It is, however, permissible to use *pessimistic* approximations. Then, the *safe* case is still valid but *unknown* can no longer be distinguished from *unsafe*: The algorithm will return *unknown* if the system is unsafe but also if the approximations were too coarse or, e.g., the memory limit was exhausted.

The previous definition of formal verification is mostly synonymous to the term *model checking* used in a more theoretical context (cf. Clarke et al., 2018). Note that despite its name, model checking does *not* check whether the model itself is realistic and correctly implemented. Instead, the model is assumed to be true and checked against the specification.

Formal verification comes at a late stage of the development process, when the design is already given. The idea can be extended to formal *synthesis* techniques to derive a design that is correct by construction. This work, however, is restricted to verification as a necessary first step of research.

Nondeterminism and Uncertainty For physical objects, a model can not exactly predict the behavior. To achieve valid safety conclusions, such uncertainty must be considered explicitly. Within this work, a model is denoted *nondeterministic* if multiple output behaviors are possible for the same input. Nondeterminism naturally arises from "random" external influence, e.g., thermal noise in an electrical circuit. Aside from physical reasons, uncertainty also results from a lack of knowledge: Unmodeled or unknown aspects of the system, such as the road friction coefficient for a car, are omitted and instead represented by a range of possible values.

Hard vs. Stochastic Guarantees For dealing with safety under uncertainty, there are two major approaches: Hard and stochastic guarantees.

Hard safety guarantees use Boolean logic. Then, the task of safety verification is to prove that a safety property is always true as long as certain assumptions on the uncertain variables hold. A typical example is to prove bounds on the output for given bounds on the disturbance.

Stochastic guarantees are a prominent alternative. Here, probability distributions are assumed for all uncertain variables. Then, the goal is to compute probabilities of staying within safety bounds.

Both variants have their specific advantages. Stochastic guarantees are a good match for phenomena that are modeled probabilistically, such as random hardware failures. On the other hand, if an adverse condition is common, such as medium side wind, then the system should always be safe under this condition, which leads to a hard Boolean requirement.

Due to the vastly different theoretical frameworks, the research in this thesis only considers hard safety guarantees. Related results for stochastic analysis are given in Gaukler, Michalka, et al., 2018.

Formal vs. Simulative Approach For safety-critical systems with hard requirements, there are two ways to achieve verification: "by a provably correct, efficient procedure or by exhaustive simulation and testing" (Liu, 2000).

While simulation and testing may be preferable due to their straightforward implementation, it is often impossible to exhaustively cover all states of a system, particularly with increasing complexity. Whenever simulation or testing are incomplete, they are not usable for formal verification of hard safety requirements: If a counterexample was found, the system is proven unsafe. Else, the result is unknown in a Boolean sense, and at best a stochastic guarantee. For the example of autonomous cars, even a statistical verification is considered impossible by testing alone (Kalra and Paddock, 2016).

To close this gap and verify hard safety requirements, specialized techniques for formal verification have been developed, at first for discrete-event systems including software programs but more recently also for hybrid systems, which combine continuous-time and discrete-event dynamics (Clarke et al., 2018). This topic will be detailed later (Sections 2.2 and 2.5).

Research Question 2.1.2. *How can formal verification be used to assess the impact of timing on the safety of control systems?*

In the following, the aspects of control and computing are discussed in detail.

2.2 Feedback Control

This section will explain the context of control engineering, illustrating the typical design process as put forth by standard textbooks (Åström and Wittenmark, 1996; Dorf, 2005), and its relation to safety verification.

Terminology The purpose of a feedback control system is to ensure the desired behavior of a physical process despite uncertainties, e.g., that a car keeps its lane despite side wind. The process to be controlled is generally termed *plant*. The situation of the plant is measured by a sensor (plant output), an appropriate corrective action is computed according to a control law, and then this command is sent to an actuator (plant input). This process is repeated continuously.

Note that in control design, the terms input and output generally refer to the plant, while in computer science, these terms refer to the computer. Hence, the meaning is reversed depending on the context, as the computer output is the plant input and vice versa. Within this work, input and output refer to the plant, if not specified otherwise.

Requirements The basic requirement on a control algorithm is the stability of the control loop. Stability means that after a short disturbance the system converges back to its steady state. Instability can render the control loop useless by uncontrollable oscillations or a diverging output.

A further major requirement is that up to the expected amount of disturbance, the system stays within a defined safe set of states. For example, a lane assist system must keep the position within the lane to avoid crashes.

These two requirements can be considered as the bare minimum to achieve safe operation. They are not yet sufficient for comfort, which gives rise to a multitude of further requirements, such as low actuation effort and a subjectively good transient response to disturbance and reference commands.

For the sake of brevity, this thesis focuses on the fundamental stability and safety requirements, providing the foundation for a future extension to comfort and performance.

Design Methodology (Dorf, 2005) Control engineering mainly employs an abstract view that is agnostic of the underlying implementation. For the example of a lane keeping system, the controller is designed mostly independent of the type of sensor used to measure the position and the choice of computer system to calculate the actuator commands. The execution of the controller is typically assumed as infinitely fast in continuous time, or, somewhat more realistic, discrete-time with a precise period.

The typical design approach is iterative: After specifying the architecture and requirements, a control algorithm is designed with respect to a part of the requirements, such as stability and the approximate settling time. Then, the remaining requirements are verified, often approximately by simulations, and, to a lesser extent, exactly by analytical calculations. If the result is not adequate, the design process is repeated with adjusted controller parameters, structure or even architectural changes such as further sensors.

For safety-critical requirements, the uncertainty incurred by approximate simulative verification is problematic. This can be avoided by formal verification. The contribution of this doctoral thesis is to use a more realistic model of the computer system in the formal verification procedure. The discussion of formal verification for control systems will be continued later.

Architecture and Complexity The starting point of control theory are single-input-single-output (SISO) systems with one input and output, i.e., one sensor and actuator with only one scalar variable each. However, many physical processes are MIMO systems that have multiple relevant variables and can not be split into separate SISO systems due to cross-couplings. This is particularly true for modern control systems. For example, the rotational and translational movement of a quadrotor helicopter is controlled by varying the rotational speed of four propellers and sensed by a combination of sensors for acceleration, angular velocity and absolute position.

An advanced control of such MIMO systems uses all inputs and outputs within one algorithm. To cope with system complexity, the controller is often structured into sub-controllers and signal processing blocks, leading to cascaded and hierarchical architectures. Beyond the traditional linear controllers with low computational cost, computation-heavy algorithms such as image processing, model predictive control or neural networks are used to increase comfort and performance.

The increasing complexity of modern control systems leads to a growing number of sensors and actuators as well as a greater demand for computing power. As will be detailed later, this makes it difficult to ensure the traditional design assumption of strictly periodic input and output timing.

Hence, this work aims at verifying MIMO systems with more relaxed timing. Within the frame of this work, only a basic monolithic controller is considered, which resembles an isolated low-level block of a larger control architecture. Owing both to the limitations of existing verification techniques and the

goal of understandability, only standard linear and, to some extent, certain nonlinear controllers and plants are considered. Two issues are left for future work: Transferring the results to the verification of other controller types and extending the results to the component-wise verification of a larger system.

Research Question 2.2.1. *How can the stability of a MIMO control loop with relaxed input/output (IO) timing be verified, mainly for the linear case?*

Nondeterminism In control systems, nondeterminism can result from sensor and actuator noise, external disturbances and uncertain dynamics, such as parameter variations or unmodeled effects (Åström and Murray, 2021, p. 5). Often, such nondeterminism only results in slightly reduced control performance, e.g., a larger deviation from the target state. However, it can also lead to catastrophically large oscillations due to instability.

From a theoretical point of view, each nondeterministic influence can be considered as an extra input to the control loop. A key theoretical property is whether for all bounded signals at this input the resulting state deviation of the control loop is also bounded. For the case of linear control loops, this is generally true for *additive* disturbance, e.g., sensor measurement noise, but not for changes to the plant or controller dynamics, e.g., parameter variations.[1] In the nonlinear case, there are no such simple results, but the linear case serves as a first approximation.

In basic control design methods, the controller is initially designed for appropriate disturbance rejection assuming an exact model of the plant. Therefore, bounded disturbance is typically tolerated, while parameter variations may be not. In general, stability under parameter variations or similar unmodeled effects is only guaranteed up to a certain robustness of the controller.

Timing variation If the input or output is not updated in the strict periodical schedule assumed in controller design, this unmodeled effect may lead to instability, even in the linear case (Marti, Villa, et al., 2001). Furthermore, variable timing can make the problem worse than merely a constant timing offset (Hetel, Fiter, et al., 2017, Sec. 2.3). Verifying stability despite timing variation is the key topic of this thesis.

[1] The background from linear control theory is that control loops must be internally stable for practical use. Internal stability implies the aforementioned boundedness property for *additive* inputs. In contrast, uncertain dynamics can lead to instability because they change the loop transfer function (Åström and Murray, 2021, Chap. 12.1 and 13.2).

There is an important structural difference between timing effects and noise. Consider the case of a linear control loop. Here, noise is harmless in the sense that bounded noise only has a bounded effect on the state. In contrast, even bounded timing deviations can cause instability, i.e., an unbounded effect. This may seem paradoxical because, for the example of measurement, both timing variation and noise lead to measurement error. For timing, however, this error depends on the system state, so it is not necessarily bounded.

In summary, timing deviation is less comparable to measurement noise but rather a case of uncertain dynamics, where a too large variation can lead to instability. Even for the linear case, *a system that is robust to bounded input and output noise is not necessarily robust to bounded timing variation.*

Verification As noted before, verification follows the design process. Standard textbooks on controller design consider two kinds of verification:

A first category are mathematical proofs using standard theorems such as the Nyquist or Routh-Hurwitz stability criteria. While, in principle, these can be considered formal verification methods, such tests are often limited to the stability of simple (e.g., linear) systems without timing deviations.

For further requirements, such as state bounds, the traditional alternative is simulation. It is at first impossible to cover all cases because the state space is uncountable. Also, the result of simulation is only approximate due to numerical errors. Hence, as argued earlier, simulation itself is not a formal verification technique. Therefore, the verification of nontrivial requirements traditionally remained incomplete, which motivated further research.

Special-Purpose Formal Verification One direction of research are verification methods specialized to a single problem setting. There are many works on stability verification for specific forms of timing uncertainties, which will be detailed later in Section 2.5. While there are approaches for many different timing scenarios, none of the reviewed literature applies to MIMO systems with individual timing uncertainties for each sensor and actuator. However, as noted before, this case is important for modern control systems.

General-Purpose Formal Verification To support the verification of a broader class of dynamical systems, general-purpose techniques have been developed. Often, *hybrid systems* are considered that combine continuous-time behavior and discrete events. This covers most problem settings arising in modern control systems. Hybrid systems are often modeled by hybrid automata, which will be discussed later.

The most prominent verification techniques for continuous-time and hybrid dynamical systems can be summarized as follows (Clarke et al., 2018, Chap. 30; Guéguen et al., 2009; Tomlin et al., 2003):

A first approach is systematic simulation (Figure 3a): Using sensitivity analysis, one can conclude from a bundle of simulations to the overall "tube" of trajectories (Donzé, 2010; Fan et al., 2016; Kapinski et al., 2003). The idea of *reachability analysis* is related: Instead of simulating a single trajectory, the evolution of the whole set of states is computed. For *set-valued reachability analysis*, this is implemented by a set-valued variant of numerical simulation (Figure 3b), which will be detailed in Section 4.2.6.

Level-set methods (Tomlin et al., 2003) are another variant of reachability analysis. They represent the reachable set as the sub-level set $\{x \mid g(x,t) < 0\}$ of a scalar function $g(x,t)$, as illustrated in Figure 3c. The evolution of this function is modeled by a partial differential equation (PDE). The difficulty is that a state-space discretization is required to solve the PDE, which causes an exponential growth of computation time with increasing state dimension.

Another alternative is to reduce the continuous-valued system to a discrete-event system by quantizing the state space (Figure 3d). Then, one can use the mature and efficient verification tools available for discrete-event systems. However, the large number of discrete states can quickly outweigh the benefits.

Further methods are more removed from a graphical interpretation, such as *numerical certificates* (Figure 3e): Roughly, a certificate is a mathematical object represented by a list of numbers. A certificate can be easily tested and, if it is correct, shows a property of the system. Most of these certificates are related to invariant sets, i.e., a set of states that the system can never leave once it is inside. Lyapunov functions are the most prominent example and will be explored further in this thesis. Related ideas are barrier certificates (Guéguen et al., 2009; Ravanbakhsh, 2018) and invariant equations or inequalities.

Finally, it is possible to apply methods for automatic theorem proving (Immler, 2015; Platzer, 2018). These tools build upon the framework of formal logic, which has both advantages and disadvantages: The majority of the code of the tool is proven correct by the tool itself, while only a small core of the tool must be trusted. This significantly limits the possibility of bugs. A further advantage is that the same framework is already used for the verification of computer programs and digital hardware, opening the future possibility of unified verification from low-level hardware up to the high-level control loop. The disadvantage of formal logic is, however, that it has little connection to standard control theory as taught in typical textbooks or university courses,

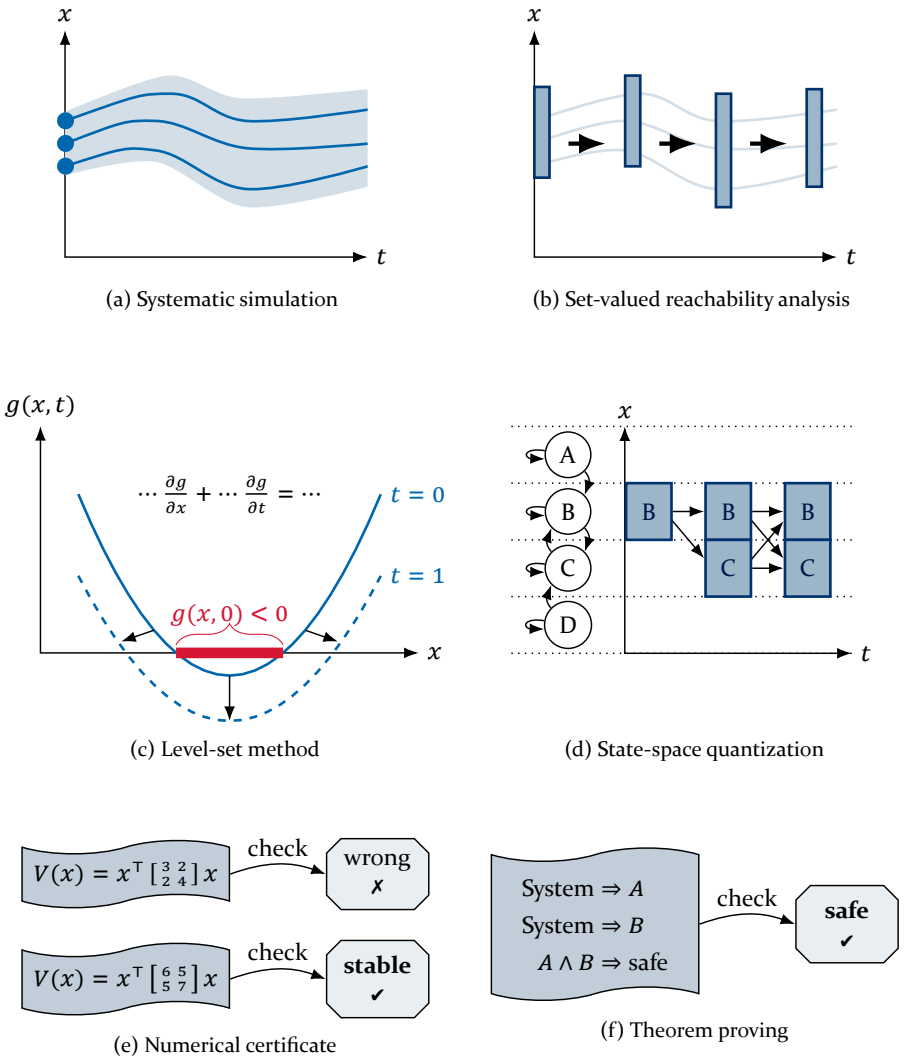


Figure 3: Overview of important general-purpose formal verification techniques

making it difficult to use for many control engineers as long as these tools are not a fully automatic black box. "It is generally believed that formal methods will not have a significant impact unless they can be used by people who do not understand them." (Bahig and El-Kadi, 2017)

Common to all techniques for formal verification is a trade-off between accuracy and speed. To determine a safety proof, any inexact computation must be replaced by worst-case bounds, such as an interval around the exact solution. As will be detailed later, if this approximation is too coarse, safety can no longer be shown, while the computation time will increase drastically for finer approximations. Hence, many practical examples will be in a gray area where the system can neither be proven safe nor unsafe, except possibly with more computing power or with "more intelligent" methods. Many typical problem classes of formal verification are proven to be undecidable (Henzinger, Kopke, et al., 1998), i.e., there is no algorithm that verifies every system in finite time.

As the further investigation will show (Section 4.2.10), considering IO timing variations in general-purpose formal verification tools increases the complexity and therefore the computation time. Hence, existing work on verifying control systems with such general-purpose tools mainly uses the simplifying assumptions already employed in control design, i.e., continuous or strictly periodic execution. As an alternative to general-purpose tools, there are specialized techniques for very narrow classes of systems, which will be discussed later in Section 2.5.

Research Question 2.2.2. *How can general-purpose formal verification tools be used to efficiently verify MIMO control with relaxed timing requirements?*

Conclusion In consequence, there are two possibilities to verify the safety of MIMO control systems: The traditional choice would be to ensure ideal timing and applying existing knowledge for verification. The alternative pursued in this work is to extend verification techniques to support IO timing uncertainty for the MIMO case.

2.3 Real-Time Computing

Providing the required timing precision for sensing and actuation is a substantial constraint on the computing system. As the following discussion will show, there is a general trade-off between predictable timing and a cost-efficient computing platform. Consequently, unpredictable timing variations occur in many levels of a typical computing system, from processor subsystems to the overall application. This problem is enlarged for modern, complex control systems, so that the classical requirement of ideal timing is no longer feasible.

A key example is the principle of shared hardware resources, e.g., when multiple applications share a single communication network. This can also occur hidden at lower hardware layers such as processor subsystems. In general, such resource sharing saves cost but leads to temporal interference, e.g., waiting times when a resource is already in use. Nevertheless, it is widely employed in modern control systems to cope with increasing system complexity.

To further explain the background of unpredictable timing variations, this section introduces key aspects of real-time systems based on the textbooks of Lee and Seshia, 2016; Liu, 2000 and Tanenbaum, 2009. Readers with a good knowledge of real-time systems may skip to the conclusion on page 22.

Processor For a closer look on the details, the starting point is the heart of the computing platform, the central processing unit (CPU). It executes a program given as a list of low-level instructions, such as: loading from memory, performing arithmetic, storing the result and jumping to another point of the program if a certain condition is true. In a simple implementation, a processor would execute the instructions at a fixed rate. However, most modern processors deviate from this principle to improve the average performance even though the timing becomes less predictable and more varying.

One example is the use of an *instruction pipeline*, which is analogous to a factory with a multi-step assembly line: Instead of computing one instruction after another, every instruction is subdivided into small steps (*pipeline stages*) that are processed by separate units. As a simplified example, the addition of two 16-bit integers can be split into five steps: reading the instruction, loading the inputs, adding the lower 8 bits, adding the next 8 bits and storing the result. These individual steps can be computed faster than a whole instruction because they can be implemented by shorter chains of logic gates. As soon

as a step has finished, the corresponding unit is available again and used for the next instruction, so that multiple instructions can be processed at the same time. As a rough approximation, the computing throughput increases proportionally to the number of pipeline stages.

In contrast to the assembly-line analogy, a processor instruction may depend on the result of a previous instruction. If this previous instruction has not yet left the pipeline, the result is not yet available, so extra waiting time is required. In summary, pipelining increases the average performance but causes timing variations that are difficult to predict. Similar effects occur for further optimization measures, such as speculative execution, out-of-order execution and memory caching (Lee and Seshia, 2016, Chapters 8 and 9).

The computing throughput can also be increased by using multiple processors that execute one program each. This is typically realized in the form of multiple *cores* within one physical CPU chip that share certain expensive resources such as the memory subsystem. However, this resource sharing and the exchange of data between multiple cores can also lead to timing variations.

Lee and Seshia, 2016 conclude that "an embedded system designer needs to [...] avoid processors where the requisite level of timing precision is unachievable". This exemplifies that timing requirements lead to reduced performance efficiency and therefore increased hardware cost. In particular, overly strict timing requirements preclude the use of cheap "off-the-shelf hardware", such as the CPUs used for desktop computers, because such hardware is highly optimized for average performance at the cost of predictable timing.

Input and Output IO interfaces are essential for control loops, as they provide the connection to the physical world. For the control algorithm, it is no problem if the computation finishes earlier than required. In contrast, the sensors and actuators must be updated neither too early nor too late.

Historically, most IO signals were analog voltages, for which the computing system used analog-to-digital and digital-to-analog converters. In the simplest case, these converters have no own timing circuitry but are triggered by special CPU instructions. Then, the sampling times are coupled to the CPU timing and subject to similar uncertainties. Effectively, the implementation results in an *IO time window* around the ideal periodic sampling point.

To save cabling cost and avoid the inaccuracies of analog signaling, modern control systems use digital communication networks (buses) shared by multiple sensors, actuators and controllers. This shared medium can lead to varying transmission delays (Wittenmark, 2001). Furthermore, signal preprocessing,

e.g., downsampling, is often moved from the main CPU into the sensors to reduce the amount of transmitted data. These intelligent sensors have their own internal clock and sampling rate that is generally decoupled from the main system, which may contribute to further delay variations.

Without further design measures, the sampling time of data in such network-based control loops does not strictly correspond to the time at which it was requested by the computing system. The resulting sample time variations can be reduced by synchronizing the sampling rates of all components, which however requires increased development effort and is only possible within finite precision. Again, the result is a time window.

Research Question 2.3.1. *How can the safety of control systems with IO time windows be verified?*

Operating system Modern control systems often integrate multiple functions on one computing system. For this, they use an operating system (OS), which is an intermediary software layer between application and hardware that allows multiple application programs to share a single computing system. The OS manages the hardware resources, mainly CPU, memory and IO and distributes them among the programs so that each can run almost as if it had its own computing system. For example, a single-core CPU can only run one program at the same time, so a simple OS would run each program for a fixed period of time before pausing it and switching to the next program.

In principle, a simple controller can be implemented without an OS as a single endless loop: Receive from the sensors, compute the next actuator value, wait until the end of the period, send to the actuators and repeat. However, this OS-less approach is infeasible for complex systems, which typically consist of a multitude of interdependent activities with widely varying timing requirements: For example, autonomous driving requires multiple control layers on different time scales, ranging from high-level route planning to the low-level control of a single wheel. Additionally, each control layer consists of multiple activities with different timing requirements, such as reading sensor values and performing the computation.

By using an OS, system developers no longer need to specify the execution flow explicitly. Instead, they only define the timing parameters and dependencies of each activity, while the execution is orchestrated by the OS. This comes with the drawback of greater timing uncertainty, in the sense that the timing of one activity has a complex dependency on the timing of other activities and on internal OS strategies. These strategies are discussed in the following.

OS Terminology The temporal allocation of shared resources, here CPU and IO, is termed *scheduling*. As the strategies and effects are similar for any resource, the discussion will focus on the CPU without loss of generality.

Following the terminology of Liu, 2000, an application consists of multiple *tasks*, each of which repeatedly creates a *job*. A *job* is a unit of work, e.g., reading the sensors in the k-th period, that has concrete *timing constraints* and generally runs for a finite time: It must not be started before its *release time* and it must finish before its *deadline*. The CPU duration that a job needs to complete is termed the *execution time*. As will be discussed soon, it is not exactly known in advance but a bound is necessary to verify the timing.

A *task* is a set of related jobs. Typically, it is an ongoing stream of activity that exists throughout the whole life of the system and corresponds to a specific function, e.g., reading the sensors every period. For a *periodic task* with period T and relative deadline D, a job with release time kT and deadline $kT + D$ is created in every period $k = 1, 2, \ldots$. Most control activities are such periodic tasks. Besides, there are *sporadic tasks* whose jobs are triggered by asynchronous events such as the arrival of a network packet or of user input.

In the analysis of time durations, e.g., from releasing a job to the time when it finishes, the temporal variation between worst and best case is termed *jitter*, while the average time offset is termed *delay*. For control systems, the impact of delay can be compensated to some extent, e.g., by accounting for it in the control algorithm or by releasing the task earlier. Jitter is more problematic, as the resulting timing varies and is not known in advance.

For scheduling, there are a multitude of strategies. Two important distinctions can be made: static versus dynamic and preemptive versus nonpreemptive.

Static scheduling means that a fixed time-table is determined at design time according to the worst-case execution times (WCETs) and timing constraints of all tasks. At run-time, this time-table is repeated periodically. Static scheduling is predictable but pessimistic: If a task executes faster than the WCET, the resulting gap in the schedule is unused. In pure form, the scheme is restricted to periodic tasks. This motivates the use of *dynamic scheduling*, where decisions are made at run-time. Thereby, idle times are avoided and the average delay of jobs is improved, though with the drawback of more jitter. In a common variant, the CPU always executes the job with highest priority among those that are released but not yet finished. Herein, the priority can be a task attribute, e.g., that all controller-related jobs are more important, or

be determined from a criterion such as earliest-deadline-first. The principles of dynamic and static scheduling can be combined by the use of *server jobs* where, in the basic case, a periodic pseudo-task in a static schedule serves as a container within which a group of sporadic tasks is scheduled dynamically.

The second important distinction is whether the scheduling is *preemptive*: To allow for long-running, computationally heavy background tasks, jobs may be paused (*preempted*). This is possible both for static and dynamic scheduling. It is necessary in complex systems but causes difficulty because many resources are not preemptable. To give a simplified example, sending a network packet can not be paused and continued later, so if one job has started writing to the network interface, all other jobs will have to wait until the packet is finished. Such a non-interruptable program part is called a *critical section*. Note that the interference occurs independent of the priority of the jobs, so a job of little importance can temporarily block the execution of an urgent job. This phenomenon, called *priority inversion*, occurs in many aspects of multi-task systems and is hard to avoid.

In summary, growing system complexity leads to interference between different tasks, which can not be avoided without a significant reduction of average-case performance. Hence, weakening timing requirements is beneficial for the design of complex real-time control systems.

Real-Time Systems A computing system is denoted as *real-time* if the timing is important for the correct function of the system. By Liu, 2000, the jobs in a real-time system can be divided into *hard* and *soft*. Hard real-time jobs have timing requirements that must be deterministically guaranteed and rigorously verified. For example, jobs in an airplane control system are generally considered hard real-time due to the potential loss of lives. In contrast, the timing constraints of soft real-time jobs are not guaranteed but only met on a best-effort or probabilistic basis. To give an example, a consumer multimedia system is focused on cost and typical performance, so a small probability of deadline misses is acceptable. As a third category, non-real-time jobs do not have explicit timing constraints. For example, a word processing application on a personal computer does not specify any response time to keyboard input. In practice, the borders between hard, soft and non-real-time are blurred: Taken strictly, all jobs have some real-time aspect, as every job should finish in reasonable time, and no job is ideally hard real-time because timing verification is only possible up to some level of rigor.

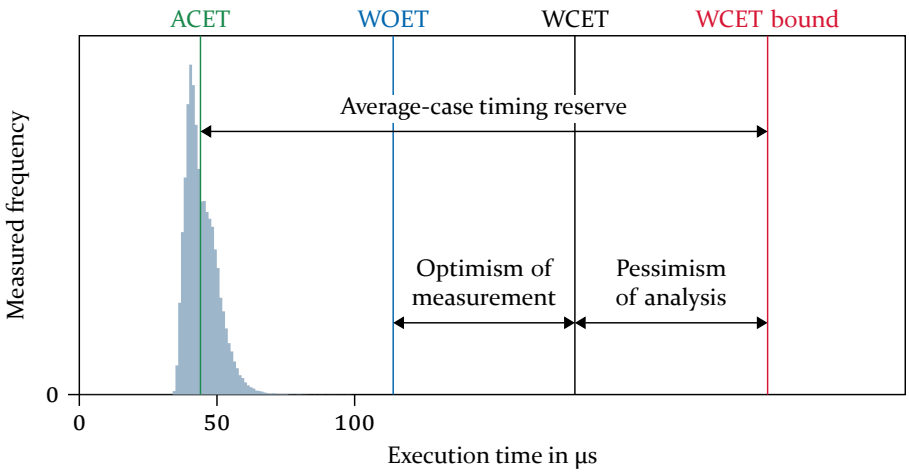


Figure 4: Key figures of an execution time distribution. Measurement data is taken from Hansen et al., 2009, Fig. 2. The WCET value and WCET bound are assumed for illustration, as they were not given in the original source.

With regard to real-time systems, the goal of this doctoral thesis is to widen the hard IO timing constraints of real-time control without sacrificing hard safety guarantees. A first example is the concept of time windows, which directly matches the established concept of deadlines and release times. The focus of this thesis is on the control-theoretical side, considering methods for safety analysis under such flexible timing concepts. The implementation on a real-time operating system is left for future research.

Formal Timing Verification A crucial aspect of real-time system design is to verify that the timing requirements of all hard real-time jobs are satisfied. The starting point of this analysis is to determine an upper bound on the WCET of each job (Lee and Seshia, 2016, Chap. 16). To illustrate the difficulty, Figure 4 shows the execution time distribution of an exemplary task measured by Hansen et al., 2009. The task is part of an unspecified commercial control system with 74 tasks. The diagram shows an average-case execution time (ACET) of 44 microseconds, while the worst observed execution time (WOET) was 114 microseconds. As it is practically impossible to cover all potential cases in a finite measurement, the WOET is generally below the WCET.

For soft real-time jobs, one can accept the WOET as optimistic approximation of the WCET. In contrast, hard real-time jobs require rigorous verification, so one must resort to provably correct upper bounds. In general, such a bound is computed by summing up the CPU instruction times along the “longest”

path in the execution flow. This is nontrivial due to the internal states of the processor, e.g., the pipeline, and the states of the program, which affect loops and branches. Interference by other tasks must also be considered, e.g., blocked IO or the effect of preemption on the pipeline state.

Overall, the analysis must resort to overapproximations, so that the resulting WCET bound is typically much larger than the actual WCET. This leads to a gray area between WOET and WCET-bound in which the timing will probably, but not provably, never be. As the ACET is even below, this means that hard real-time jobs generally have a *timing reserve*, i.e., a task will typically finish significantly before its WCET-bound.

After the bound on the WCET of each job has been determined, the next step is to verify that the scheduling meets the timing constraints of all jobs.

Static scheduling is predictable but inefficient: The schedule is constructed at design time, so it can be easily checked. However, if a task finishes early, then the processor is idle until the end of the allocated time slot. Only with dynamic scheduling, this timing reserve can be used to improve the average performance, e.g., because the leftover time is used for soft- or non-real-time tasks. However, it is nontrivial to verify that dynamic scheduling meets the timing constraints (Liu, 2000, Chap. 4–6). This analysis generally requires pessimistic overestimations, particularly for multi-core systems. Again, there is a trade-off between average performance and worst-case timing precision.

Conclusion In summary, computing systems for real-time control are facing a conflict between predictable timing and an efficient use of computational resources. As long as the safety verification of control systems relies on the classical assumption of precise input and output timing, this results in a dilemma between safety and hardware cost. This dilemma is growing for modern control systems because they use many techniques that impede predictable timing, e.g., networked sensors and multi-core processors.

As a solution, this doctoral thesis investigates how to verify the safety of the control algorithm in the presence of bounded IO timing deviations. Then, timing requirements can be loosened accordingly, which enables the use of modern, efficient hardware and reduces the effort spent on ideal timing.

2.4 Co-Design of Computing and Control

The previous sections discussed the two main aspects of real-time control systems: computing and control. This section considers how these aspects are merged in an overall design process. Traditionally, computing and control are considered separately, which leads to opposing optimization goals, particularly to the conflict between computational efficiency and precise IO timing. Hence, a separate design tends to be simple but inefficient. As an alternative, researchers have suggested a joint design of computing and control, often radically deviating from classical concepts such as periodic timing. This co-design tends to be complex but more efficient.

A shorter version of this section was previously published in Gaukler, Roppenecker, et al., 2019; Gaukler, Roppenecker, et al., 2020. Here, the discussion is expanded to incorporate the literature reviews of Årzén, Bernhardsson, et al., 1999; Simon et al., 2012; Simon et al., 2017; Xia and Sun, 2008.

Classical Design Workflow As noted before, the classical approach to the design of real-time control systems uses a *separation of concerns*: Controller design assumes exactly periodic IO, regardless of the actual timing. Then, a hard real-time system is designed to implement this ideal timing as good as possible, regardless of the timing tolerance of the controller.

One such solution focused on low jitter is clock-driven scheduling, which starts jobs according to a predetermined schedule (Baker and Shaw, 1989; Sha and Goodenough, 1989). However, this comes with the inefficiencies of static scheduling discussed earlier.

A related idea, though not restricted to static scheduling, is the logical execution time (LET) paradigm (Henzinger, Kirsch, et al., 2003): To avoid the impact of varying computation time, outputs are delayed to a fixed time, e.g., shortly before the end of the period, even if the computational result is already available earlier. This reduces jitter but increases the delay. From a control-theoretic perspective, constant delays can be accounted for more easily than time-varying jitter but must nevertheless be considered in the stability analysis. To some extent, LET can be implemented by appropriately splitting the controller into multiple jobs (Årzén, Bernhardsson, et al., 1999, Section 4). Still, various sources of timing uncertainty remain, such as preemption, critical sections or network usage by other devices. As discussed earlier, the resulting IO timing is generally not exact but lies in a certain time window.

Towards Co-Design The aspired ideal timing is a doubtful goal: It can neither be realized on typical hardware, nor is it required for stability: Virtually every control loop tolerates some amount of timing deviation, as confirmed by industrial practice (Akesson et al., 2020) and theoretical results given later in this doctoral thesis (Theorem 5.1.9). At the same time, flexibilizing the timing could reduce the performance impact and implementation effort. This has motivated research and applications to move beyond the classical restriction of ideal timing: For a *co-design of control and scheduling*, the real-time system is considered in the design of the controller and vice versa to allow for relaxed timing while guaranteeing controller stability.

Design-Time Measures A first class of co-design approaches is applied at design time, so that the timing constraints are fixed at run-time. The constraints are widened only as far as stability and performance of the controller can still be shown for any timing within the constraints and any disturbance within the assumed bounds. Applicable methods for stability analysis will be discussed later in Section 2.5. Analysis methods can also be extended to design-time optimization, adjusting scheduling parameters (Fontanelli et al., 2013; Seto, Lehoczky, et al., 1996), controller dynamics (Ben Gaid et al., 2008; Hetel, Daafouz, et al., 2006; Schinkel et al., 2002; Simon et al., 2017) or both to achieve an optimal compromise between CPU load and quality of control.

A further flexibilization leads to *weakly-hard* real-time systems that shift away from strict deadlines. A popular example is (M, K)-firm scheduling (Hamdaoui and Ramanathan, 1995; Ramanathan, 1997): Out of K consecutive job releases, only at least M jobs must meet their deadlines, while up to $M - K$ jobs may be aborted after the deadline or entirely skipped.

This skipping of controller executions is possible due to the inherent inertia (e.g., mass inertia) of typical plants. Therefore, a single skipped execution has little impact, so that the control performance mainly depends on the average "execution quality" over multiple control periods. The same principle also holds for timing variation.

Research Question 2.4.1. *How can stability be shown for MIMO control systems with IO time windows and/or skipped executions?*

The advantage of design-time approaches is that they have little computational overhead at run-time. However, they are pessimistic because the stability analysis assumes that timing and disturbance meet the worst case all of the time. Hence, the timing flexibility will be limited to what is possible as a permanent worst case.

Damage Mitigation A first extension of these design-time measures is to measure and compensate for the timing deviations from the nominal case. Controller parameters can be varied to weaken the effects of jitter (Marti, Fuertes, et al., 2001; Nilsson et al., 1998), sampling rate variations (Schinkel et al., 2002; Simon et al., 2017) or skipped executions (Sinopoli et al., 2005).

Mainly for soft real-time applications, overload management strategies are used to reduce the harm caused by deadline overruns. One aspect is how the control algorithm should react (Simon et al., 2017, Section 2.2.2, Maggio et al., 2020). *Anytime control* algorithms can be aborted prematurely and still return an approximate result (Ben Gaid et al., 2008; Greco et al., 2011). This idea can also be applied to model predictive control (Henriksson et al., 2002; Scokaert et al., 1999). A second aspect is the scheduling algorithm (Årzén, Bernhardsson, et al., 1999, Section 5), which can be modified to avoid that one missed deadline leads to a never-ending cascade of further deadline misses.

The *mixed criticality* paradigm extends such ideas on overload handling towards hard real-time (cf. Burns and Davis, 2017 and the references therein): A mixed-criticality system combines software parts with multiple levels of safety requirements (*criticality*). For example, in an aircraft, flight stability is more critical than autopilot navigation, which is more critical than passenger entertainment. To ensure safety, a mixed-criticality system is designed such that a time overrun of a less critical part does not affect the more critical parts. In the typical case, less critical parts can use resources, e.g., CPU time, that were reserved for the critical parts but not used.

A related idea from control engineering is the Simplex architecture (Seto, Krogh, et al., 1998) that switches between two controller variants: Normally, a performance-oriented controller without safety guarantees is active. If a supervision logic detects that the control error becomes too large, a safety-proven baseline controller is activated instead. Similar to the mixed-criticality paradigm, typical performance can be combined with worst-case safety.

The reviewed literature on mixed criticality and the Simplex architecture is restricted with regard to IO timing: The Simplex architecture considers control performance under the assumption of ideal IO timing, which is impractical. Mixed-criticality scheduling considers hard deadlines, i.e., not a single deadline may be missed, and fixed criticality levels. However, as discussed before, control tasks tolerate occasional deadline violations, so their criticality varies depending on the recent execution timing. Hence, the open question addressed by this doctoral thesis is:

Research Question 2.4.2. *Simplex and mixed-criticality approaches combine efficient normal-mode operation with a safety fallback. How can this principle be applied to real-time control systems with flexible IO timing?*

Adaptation at Run-Time In benign scenarios, the timing and disturbance will be far below their assumed worst-case bounds for most of the time. Then, fixed timing constraints may be overly restrictive. To overcome this limitation, co-design was extended to run-time mechanisms. Such *dynamic resource management* optimizes the resource (CPU and network) allocation as far as permitted by the current state of the control system (Xia and Sun, 2008).

It should be noted that in a theoretical worst-case scenario, there is nothing to gain by dynamic resource management: If all "randomness" is controlled by a fictitious evil opponent, the largest permissible time window or skip pattern is exactly the same as obtained from design-time analysis. In practice, however, the actual execution will typically be better than the permitted bound, particularly due to the long tail of the typical probability distribution of Figure 4. This difference between reality and worst-case can be exploited for more flexibility: Due to inertia in the system dynamics, good timing in one period can compensate for some amount of bad timing in the next period. Hence, it is possible to temporarily allow large timing deviations that would not be possible over a longer time horizon.

Figure 5 shows that resource management can be interpreted as an unusual control loop, namely the *control of the computing system* (Xia and Sun, 2008). Consequently, *feedback scheduling* approaches (Årzén, Bernhardsson, et al., 1999) use control algorithms such as PID to adapt the timing parameters. Depending on the choice of actuation, measurement and target variables, different goals can be achieved: sharing resources among multiple controllers (Castane et al., 2006), overload management, or executing the controller as often possible (Lu et al., 1999) or as seldom as necessary. Examples for actuation variables (Årzén, Bernhardsson, et al., 1999, pp. 19–23) are period adjustment, skipped executions, multiple controller modes or processor speed (Xia, Tian, et al., 2008). Available measurements are the CPU load, deadline miss ratio (Lu et al., 1999) or controller state (Castane et al., 2006). The main "disturbance" inputs of this feedback loop are the CPU load by non-control applications, the nondeterminism in timing and the physical disturbance. A major difficulty is that modeling the "plant", i.e., the scheduler and its influence on the quality of control, is largely impossible within standard

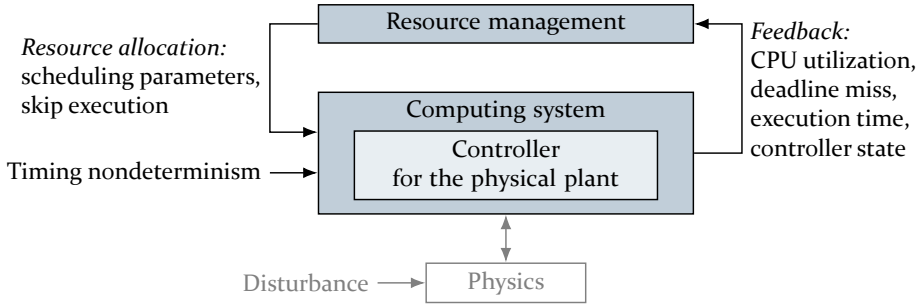


Figure 5: Dynamic resource management

control theory (Leva et al., 2020). A proper model of scheduling algorithms instead requires extensive use of discrete-event systems theory (Altisen et al., 2002), in particular to capture anomalous behavior (Liu, 2000, Sec. 4.8.1). For example, shorter execution times can lead to more deadline misses.

Of the reviewed publications on feedback scheduling, many use control-theoretical methods but only few provide theoretical guarantees on stability or control performance. Without such guarantees, they are not suitable for hard safety requirements. The few that do provide safety guarantees rely on stability for *arbitrary* timing variations within the possible bounds (Castane et al., 2006). Hence, for systems with hard safety requirements there is no known variant of feedback scheduling that allows for an improvement beyond what is possible as a permanent worst case.

Research Question 2.4.3. *How can timing adaptation achieve hard safety guarantees but also high flexibility beyond the discussed worst-case limit?*

In particular, consider a given controller with a variable IO time window and, optionally, skipped executions. How can these timing parameters be adjusted dynamically based on a measurement of the previous execution timing?

Event-Based Control Timing adaptation in a broader sense is enabled by methods for *Event-Based Control* (Heemels, K. Johansson, et al., 2012; Miskowicz, 2015; and the references therein). The general idea is to execute the controller only after a sufficient change of the plant state. In the basic variant, event-triggered control, a sample–compute–actuate cycle is executed immediately as soon as the system state has changed sufficiently, e.g., whenever the sensor signal has changed by a certain amount. This asynchronous behavior is problematic with regard to the implementation in real-time control systems: An immediate response is not realistic due to other tasks in the system and

because the request is not known in advance. Also, scheduling analysis and the design of multi-rate systems is easier for periodic tasks or quasi-periodic variants with skipped executions. Furthermore, the sensor must provide the triggering signal. This motivates the principle of *self-triggered control*, where the controller proactively computes the next execution time depending on the current system state. This can be extended to *periodic self-triggered control*, in which the jobs are restricted to a periodic grid.

In general, the studied literature on event-based control provides mathematical stability proofs and worst-case error bounds, providing techniques for a timing adaptation as in Research Question 2.4.3. However, it does not consider the IO timing variations occurring in MIMO systems. Instead, it is assumed that either all sensors are sampled synchronously or that only one sensor is sampled per period. A further limitation is that much work in the area is targeted at networked systems, where network usage is the only scarce resource and computational overhead is of little relevance.

Research Question 2.4.4. *How can timing adaptation for a given controller (as in Research Question 2.4.3) be implemented with low computational overhead, even in complex MIMO systems?*

Summary There is a large body of work on improving the coexistence of control algorithms and their computing platform. However, one aspect has so far received little attention: In a MIMO control system, each sensor and actuator may have an individual IO timing variation. This scenario poses a number of research questions that will be addressed in this doctoral thesis, ranging from design-time stability analysis to dynamic resource management.

2.5 Stability Analysis for Uncertain Timing

As noted in the previous section, the starting point for co-design is design-time analysis of the worst-case control performance under uncertain timing. This section reviews existing control-theoretic work on that topic, focusing on stability analysis of real-time control systems. This system class is also termed *sampled-data systems*. The techniques and challenges for stability analysis are representative of other aspects of design-time analysis, such as state bounds. Furthermore, the same techniques serve as the starting point for dynamic resource management and also for controller synthesis and optimization.

The following discussion is based on the reviews of Hetel, Fiter, et al., 2017; Simon et al., 2012 and an own literature survey. It starts with an overview of the major theoretical frameworks for modeling the control loop with timing effects (Section 2.5.1). Then, corresponding stability analysis techniques and the applicable timing scenarios are considered (Sections 2.5.2 and 2.5.3). Finally, the resulting difficulty for analyzing time windows in MIMO systems is discussed (Section 2.5.4).

Previous, shorter versions of this section appeared in Gaukler, Roppenecker, et al., 2020 and the corresponding report (Gaukler, Roppenecker, et al., 2019).

2.5.1 Formal Frameworks for Modeling

The starting point for stability analysis is to bring the control loop, which contains both continuous-time and discrete-time dynamics (Figure 6a), into a unified formal framework.

To simplify the following discussion, the system and controller dynamics are assumed to be linear and value quantization is neglected. The formulas are given for the simple example

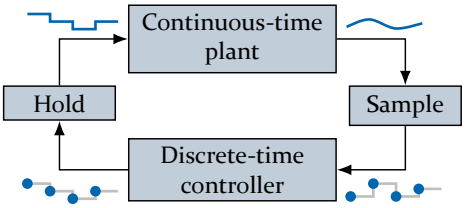
$$\dot{x} = Ax + Bu, \tag{1a}$$

$$u(t) = K_{\text{ctrl}} x(k(t)\,T) \tag{1b}$$

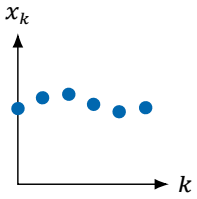
of discrete-time state-feedback control with ideal timing and a sampling period of T. Herein, the state x and actuation u are vectors and A, B, K_{ctrl} are matrices of appropriate size. The notation $k(t)$ refers to the current value of the sample index k, i.e., $k(t) \in \mathbb{N}_0$ such that $k(t)\,T \leq t < (k(t)+1)T$.

2.5.1.1 Discrete Time

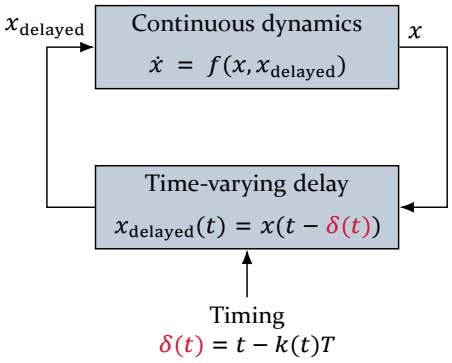
The first major distinction among possible theoretical frameworks is between continuous and discrete time. If a discretization is applied, this results in the *switched discrete-time system* $x_{k+1} = A(\sigma_k)x_k$ (Figure 6b), where the *switching signal* σ_k encodes the timing. The discretization process will be detailed later in Section 4.1.4. Two cases of switching can be distinguished (Shorten et al., 2007): The basic case is *arbitrary switching*, where σ_k is restricted to a set but otherwise uncontrollable. This corresponds to IO timing that can vary arbitrarily within given bounds. In contrast, for *restricted switching* the switching sequence obeys certain rules, e.g., (M, K)-weak execution, or can be influenced, e.g., by dynamic resource management.



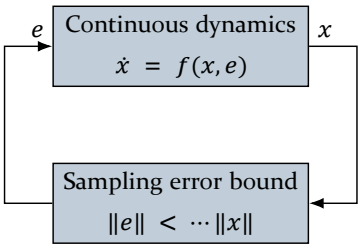
(a) Original: mixed continuous and discrete



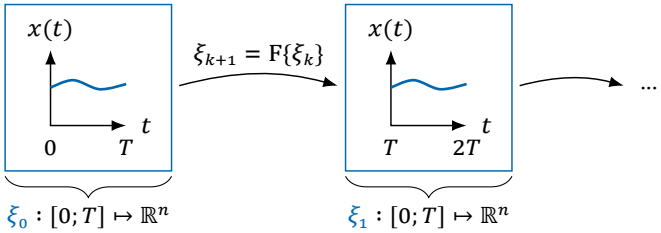
(b) Discrete-time system



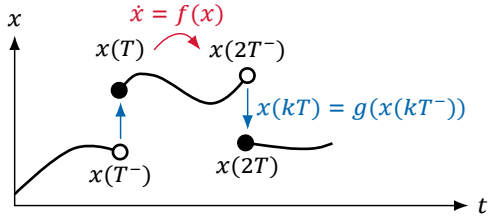
(c) Time-delay system



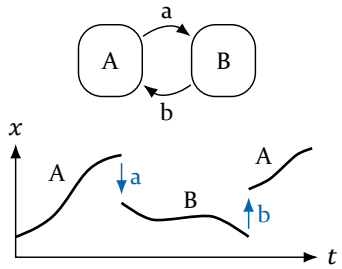
(d) Input-output approach



(e) Looped functional



(f) Impulsive system



(g) Hybrid automaton

Figure 6: Formal frameworks for modeling real-time control systems

Among the discrete-time methods, a further distinction is whether the set of possible $A(\sigma_k)$ has a finite number of elements. The finite case is easier and corresponds to a finite number of timing variants. If there are infinitely many timing variants, such as a continuous interval of delays, the analysis is more difficult. Analysis will be discussed subsequently in Section 2.5.2.1.

2.5.1.2 Continuous Time

A different theoretical perspective arises if the system is described in continuous time. Then, the sample-and-hold dynamics must be modeled explicitly, for which there are multiple prominent variants. The corresponding analysis methods will be discussed in Section 2.5.2.2.

Time-Delay Systems A first variant, depicted in Figure 6c, uses delayed differential equations, where the sampling $x(k(t)T)$ is described as a time-varying delay $x(t - \delta(t))$. Stability analysis then uses properties of the delay time $\delta(t) = t - k(t)T$, e.g., its maximum (Hetel, Fiter, et al., 2017, Sec. 4.1).

Looped Functionals A slightly different view is to consider the evolution of the continuous-time signal from one period segment to the next (Figure 6e). In informal terms, the state variable is not $x(t)$ at one time point but the whole trajectory from $x(kT)$ to $x((k+1)T)$ during one period k. Mathematically, the new state ξ_k is defined as a function on the interval $[0; T]$:

$$\xi_k(\tilde{t}) \coloneqq x(\tilde{t} + kT), \quad 0 \leq \tilde{t} \leq T. \tag{2}$$

Here, ξ_k evolves in discrete time k from one period to the next, connecting the discrete- and continuous-time views (Seuret, 2011; Seuret, 2012).

Impulsive Systems The delay operator is difficult for mathematical analysis because it leads to an infinite-dimensional system state: Formally, the behavior of the system at time t depends not only on $x(t)$ but also on the past of x back until $x(t - \delta)$. This is avoided by a different modeling variant, the impulsive differential equation (Figure 6f): The state vector x is extended to include the "memory" of the sample-and-hold mechanisms. With this new state vector, the system evolution alternates between continuous dynamics and discrete "jumps" (Hetel, Fiter, et al., 2017, Sec. 4.2.1). This will be detailed in Section 4.1.

Hybrid Systems A hybrid automaton (HA) combines an impulsive system with a discrete-event automaton (Figure 6g): Every state of the automaton corresponds to continuous-time evolution, and every transition corresponds to an impulsive "jump" (Hetel, Fiter, et al., 2017, Sec. 4.2.4). Therefore, HA can not only encode the continuous-time plant and discrete-time controller behavior but also the timing bounds and (M, K)-constraints. Furthermore, state-machine-like behavior of controller and plant can also be modeled, e.g., if an aircraft switches to a different controller for low flight heights. The topic of HA will be continued later in Section 4.2.

Input/Output Properties Many real-time control systems behave similar to a quasi-continuous execution with infinite sample rate. Accordingly, the system can be rewritten as quasi-continuous dynamics disturbed by a *sampling error* $e(t) = x(k(t)T) - x(t)$ (Figure 6d). Under appropriate formal definitions, this sampling error is generated by an operator with bounded input-output gain. The analysis then considers the operator as a black box for which only the maximum gain is known (Hetel, Fiter, et al., 2017, Sec. 4.4).

Finally, it should be noted that the discussed formal frameworks exist on a continuum and therefore not all publications fit into exactly one category.

2.5.2 Analysis Techniques

Corresponding to the formal modeling frameworks, there is a range of techniques for stability analysis, mostly dominated by variants of the Lyapunov stability theorem. Depending on the scenario (skipped execution, uncertain delays, SISO or MIMO, linear or nonlinear, etc.), these techniques vary in mathematical complexity, computational effort and precision (pessimistic vs. tight bounds). Prominent variants will be summarized in the following.

2.5.2.1 Discrete Time

Multiple standard techniques for the stability analysis of switching, i.e., time-varying, discrete-time systems exist, as detailed in Hetel, 2007, Chapters 1–2. Importantly, the stability condition for the time-invariant case, i.e., that all eigenvalues of $A(\sigma_k)$ lie in the unit circle for all possible σ_k, is *not sufficient* in the time-varying case. This means that switching between timings that are stable in isolation does not imply stability of the switched system (Schinkel et al., 2002). Instead, stricter conditions are required. This leads to the *joint spectral radius* (Jungers, 2009), which is generally more difficult to compute than the classical spectral radius, i.e., the largest magnitude of all eigenvalues.

Stability can be shown by modified variants of the discrete-time Lyapunov theorem. The basic variant is the *common Lyapunov function*, where a scalar and positive definite "energy function" $V(x)$ is shown to be decaying in every time step independent of σ_k. This can be extended to a *parameter-dependent Lyapunov function* that depends on the switching signal, e.g., $V(\sigma_{k-1}, x_k)$. In general, there is a compromise between computation time and accuracy because some "difficult" systems can only be proven stable using more complex Lyapunov functions. For linear systems, V is often chosen as a quadratic form so that the problem can be posed as linear matrix inequality (LMI). In this framework, common quadratic Lyapunov functions (CQLFs) are the basic case that is easiest to compute. The principle can be extended to either parameter-dependent quadratic or piecewise-quadratic Lyapunov functions. Both are closely related (Philippe et al., 2019).

Finitely Many Switching Variants Some timing scenarios lead to a finite number of transition matrices $A(\sigma)$. For example, Schinkel et al., 2002, show stability of a real-time control system with time-varying sampling rate, assuming a finite number of sampling rates and using a CQLF.

An (M, K)-weak execution also corresponds to a finite number of matrices. Here, a CQLF can not be directly used because skipping an execution generally results in "temporary instability". Therefore, the approach is modified towards a multi-step Lyapunov function (Blind and Allgöwer, 2015): It is shown that the function V decreases after multiple steps, even though it may increase in between. For example, one can consider the difference from one non-skipped controller execution to the next, ignoring the temporary increase at skips (Felicioni et al., 2008; Linsenmayer and Allgöwer, 2017). Graph Lyapunov functions are an alternative to the multi-step approach, where the (M, K)-constraint is represented by an automaton graph and the Lyapunov function depends on the state of the automaton (Linsenmayer and Allgöwer, 2017). Aside from Lyapunov methods, one can also reuse generic methods for computing the joint spectral radius (Vankeerberghen et al., 2014), as applied in Maggio et al., 2020 for a specific subset of (M, K)-weak execution.

Infinitely Many Switching Variants If the unknown delay can take continuous values within an interval, this results in infinitely many timing variants. Then, the discretization of the system dynamics leads to uncertain matrix-exponential terms, such as $A(\sigma) = M_1 \mathrm{e}^{M_2 \sigma}$ with the scalar delay $\sigma \geq 0$. To

avoid this difficult nonlinear parameter dependence, a common approach is to bound the resulting matrix set by other shapes, such as a convex matrix polytope conv$\{V_1, \dots, V_{n_v}\}$, where conv denotes the convex hull and $V_1, \dots, V_{n_v}$ are constant matrices, called the vertices of the polytope.

Several methods exist for this procedure (Heemels, van de Wouw, et al., 2010): Using the Jordan form (Cloosterman et al., 2009) or Cayley-Hamilton theorem results in a polytope. Using the Taylor series (Hetel, Daafouz, et al., 2006) or a grid of possible timing combinations (Heemels, van de Wouw, et al., 2010) leads to a polytope with an additional bounded error term.

A counterintuitive but helpful fact is that for stability analysis, a convex matrix polytope (without error term) is *equivalent* to just the finite set $\{V_1, \dots, V_{n_v}\}$ of its vertices. This reduces the problem to the previous case of finitely many variants (Mason et al., 2007, Remark 5; Shorten et al., 2007, p. 554 bottom).

In an approach closely related to the polytopic approximation, the dynamics matrix is rewritten such that it depends linearly on timing-dependent parameters α_i, resulting in $A_k = \sum_i \alpha_i(\sigma_k)\, M_i$. This is a special case of a linear parameter-varying (LPV) system (Hetel, Daafouz, et al., 2006; Simon et al., 2017 and the references therein).

2.5.2.2 Continuous Time

The methods for continuous-time systems are mostly an adaptation of the discrete-time case.

Time-Delay Systems For time-delay systems, the Lyapunov approach is modified to include additional terms that depend on the delayed state $x(t - \delta(t))$ (Seuret, 2011) or the time integral $\int_{t-\delta(t)}^{t} x(\tau)d\tau$ (Fridman et al., 2004). A further variant are non-monotonic Lyapunov functions, also known as Lyapunov-like functions, which are only required to decay "on average" from one period to the next and may increase temporarily in between. This also makes them suitable for impulsive systems (Briat and Seuret, 2012).

Numerical Implementation Lyapunov functions are a variant of numerical certificates. All cited Lyapunov approaches start with an ansatz (template function) whose coefficients are determined numerically to show stability. Generally, Lyapunov criteria require the positive (semi-)definiteness of functions. For quadratic forms, this results in LMIs, which are used by a large part of the literature. Efficient numerical algorithms exist for optimization

under LMI constraints (Boyd et al., 1994). A generalization to higher-order polynomials is possible by sum-of-squares (SOS)-polynomials, as used in Briat and Seuret, 2012. These result in larger LMI and therefore increased computational effort but allow for less conservative stability bounds.

A more general possibility for determining and verifying numerical certificates are satisfiability modulo theories (SMT) solvers. In principle, they can check arbitrary formulas over real-valued variables including for-all and there-exists quantifiers. As a simple example, the problem "find a such that ax^2 is positive definite" is encoded as $\exists a : \forall x \, (x = 0 \vee ax^2 > 0)$ and given to the SMT solver, which returns, e.g., "satisfied by $a = 1$". However, the greater capability of SMT solvers comes with the drawback of higher computational demand. For a practical introduction to SMT, see Yurichev, 2018.

As a rule of thumb, with more parameters in the template function, a Lyapunov approach becomes more capable but also more computationally expensive.

Hybrid Systems and Reachable Sets As HA can encode both the timing constraints and the control loop dynamics in a single description, the resulting system can be analyzed using general-purpose verification techniques (cf. page 12). For example, Frehse, Hamann, et al., 2014 use set-valued reachability analysis for a (M, K)-like scenario. However, as discussed further in Sections 4.2.6 and 4.2.10, the applicability is limited due to pessimistic overapproximations, particularly for the case of uncertain IO delays. Adhikary et al., 2020 use an SMT solver to determine whether the system reaches a specified state bound. However, their analysis can not show stability on an infinite time horizon but only determine state bounds over a finite, short time range.

To reduce the pessimism of reachability analysis tools, Bak and Johnson, 2015 suggest a pre-transformation (continuization) that removes the discrete jumps from the dynamics. This approach is explored in Section 5.2. With further specialization, it is also possible to completely avoid reachability tools. Instead, the reachable set can be obtained via analytic bounds (Huang et al., 2019) or specialized set-valued computations (Al Khatib et al., 2016; el Hakim and Bekooij, 2018). A Lyapunov approach is also possible (Möhlmann and Theel, 2013), generalizing the graph Lyapunov functions discussed earlier.

Input-Output Approach The idea of the Input-Output approach is related to continuization, as both methods are based on bounding the effects of sampling. However, typical work on the Input-Output approach uses an entirely different language: Robust stability techniques such as the small-gain theorem are used (Cervin, 2012), resulting in Lyapunov functions and LMIs related to the time-delay approach (Hetel, Fiter, et al., 2017, Section 4.4).

2.5.3 Timing Models

From a user's point of view, the existing results on sample-data systems can be categorized by the employed timing model:

A large body of research considers (M, K)-weak execution (Blind and Allgöwer, 2015; Huang et al., 2019; Linsenmayer and Allgöwer, 2017) or variants thereof (Felicioni et al., 2008; Frehse, Hamann, et al., 2014; Maggio et al., 2020) but all under the assumption of ideal IO timing.

Cervin, 2012 analyzes stability for a timing model similar to the time window considered in this doctoral thesis. The analysis is, however, restricted to the SISO case, which is easier since there are only two scalar timing uncertainties, namely sensor and actuator delay. The same holds for multiple inputs and outputs if all sensors are jointly sampled and all actuators are jointly updated. This results in a system with SISO-like timing but vector-valued signals ("quasi-SISO"). However, the quasi-SISO assumption is invalid for systems with multiple sensors or actuators that are not exactly synchronized.

The quasi-SISO case of uncertain delay or sampling rate is considered in many further publications (Adhikary et al., 2020; Al Khatib et al., 2016; Bauer et al., 2012; Briat and Seuret, 2012; el Hakim and Bekooij, 2018; Fridman et al., 2004; Hetel, Daafouz, et al., 2006; Kao and Rantzer, 2007; Seuret, 2011; Seuret, 2012; Simon et al., 2017). Notable variations include that the delays are restricted to a grid (Fontanelli et al., 2013) or may be larger than one sampling period to allow for skipped executions (Cloosterman et al., 2009). To model network-controlled systems, Bauer et al., 2012 offer the alternative model that exactly one sensor or actuator is updated in every control period, thereby transforming a MIMO system to a switched quasi-SISO one. As this scenario is tailored to networked control with severely restricted communication resources, it does not match the common scenario of an embedded system that has enough resources to query all sensors in every period.

2.5.4 Extension to MIMO Systems

Despite an extensive literature review, no publication could be found that addresses the actual MIMO case of multiple sensors and actuators with independent timing uncertainties. Filling this gap is the key contribution of this doctoral thesis.

Research Question 2.5.1. *How can existing methods for stability analysis under varying timing be adapted to MIMO systems?*

A straightforward extension of an existing approach is problematic because the computational complexity would increase drastically: For m actuators and p sensors, there are $m + p$ timing variables, as well as $(m + p)!$ possible orderings of the sensor and actuator events.

For example, consider a discretization-based approach that analyzes the transition matrix $A(\sigma_k)$: A direct numerical approach based on a grid of possible timing combinations σ_k suffers from *exponential complexity* with regard to the dimension $m + p$ of the timing parameter space. Similarly, an analytical approach which uses an explicit expression for $A(\sigma_k)$ suffers from the prohibitively large number of case distinctions corresponding to the $(m + p)!$ possible orderings of sensor and actuator times.

Related difficulties occur for other methods: For hybrid automata, the large number of orderings translates to a large number of states in the automaton, making analysis slow and pessimistic, as detailed later in Section 4.2.10. In a time-delay formulation, $m + p$ separate delay variables are required, while the reviewed literature only supports a single one. Similarly, the input-output-approach is burdened by the number of sample-and-hold operators. Impulsive models with continuous-time Lyapunov functions also suffer from the dependence on ordering and multiple timing variables.

Approach One goal of this doctoral thesis is to extend the stability analysis of real-time control systems with uncertain timing to the MIMO case. As the previous discussion shows, computational complexity is the major obstacle. Therefore, the focus of a solution must be placed on computational efficiency, even if this comes at cost of some pessimism. The intended use-case is a control system with small deviations from a periodic IO timing.

Two frameworks are chosen for the developments of the following chapters: To exploit the specific problem structure of MIMO real-time control systems, the discretization of a linear impulsive system is used. It allows to derive an explicit solution of the system dynamics and determine a CQLF via LMI. However, this approach is restricted to linear systems. Additionally, continuization of hybrid automata is investigated for a general but more complex approach. The contribution in both fields is detailed in the following.

2.6 Contribution

The overall contribution of this thesis is to advance mathematical methods for the safety verification of MIMO control systems with flexible IO timing.

System Model The system model uses periodic IO time windows, allowing for a bounded deviation from the ideal periodic timing. Chapter 3 formalizes the problem and Chapter 4 presents it in two different frameworks: linear impulsive systems (LIS) and hybrid automata (HA). In each framework, a technique for stability analysis is considered in detail:

LIS-based Analysis Section 5.1 develops an approach to stability analysis in the linear case. This is the first known stability analysis method for MIMO systems with IO time windows (Research Questions 2.2.1 and 2.3.1). To exploit the problem structure, the approach employs a discretization of LIS and a novel splitting theorem, by which the complex MIMO case is broken down into a sum of SISO timing effects. Stability is then shown by bounding a norm that is equivalent to a CQLF. The major theoretical breakthrough is that the computational complexity of the MIMO case is reduced to a feasible level (Research Question 2.5.1). With this analysis, the timing requirements can be weakened without giving up stability requirements.

HA-based Analysis For stability analysis in the nonlinear case, Section 5.2 considers continuization, a "transformation" of a real-time control system into a purely continuous-time system. The result is more suitable for general-purpose formal verification tools, as it avoids the computational complexity resulting from the large number of discrete modes. The existing technique of Bak and Johnson, 2015 is put onto a new formal basis using abstractions of hybrid automata, uncovering limitations and extending it to further controller types. While the resulting technique is not yet fully extended to MIMO IO timing, this thesis lays the foundations for future work on a more general case (Research Questions 2.1.2 and 2.2.2).

Convergence Rate Abstractions The issue of computational complexity is even more pronounced for the case of dynamic resource management. As a solution, Chapter 6 introduces the novel theoretical framework of *convergence rate abstractions*: A simple one-dimensional dynamical system represents the effects of timing deviations on the worst-case control performance. Provably safe methods for analysis at design-time and adaptation at run-time can then be constructed from this simple system.

Dynamic Resource Management Dynamic resource management using convergence rate abstractions is considered in Chapter 7, where a Simplex-like architecture for timing adaptation is presented (Research Question 2.4.2). The method is provably safe and has little run-time overhead even for complex MIMO systems (Research Questions 2.4.3 and 2.4.4). The generic concept of convergence rate abstractions can be applied to different timing scenarios, such as IO timing or skipped executions (Research Question 2.4.1).

Result Numerical experiments illustrate the theoretical results and point out that safe MIMO control systems are possible without overly strict timing requirements. Overall, the results of this doctoral thesis present a step towards the development of future real-time control systems that are safe and cheap despite complex functionality.

2.7 Previous Publications

For most parts of this thesis, previous versions have been published in conference proceedings and technical reports.

- Gaukler, Michalka, et al., 2018: LIS system model (mainly Chapter 3 and Section 4.1, also Section 4.3) and stochastic analysis (not used in this thesis). A starting point thereof was presented in my Master's thesis (Gaukler, 2015).
- Gaukler and Ulbrich, 2019: HA system model and reachability analysis (mainly Chapter 3 and Section 4.2, also Section 4.3 and Appendix A)
- Gaukler, Roppenecker, et al., 2019; Gaukler, Roppenecker, et al., 2020: LIS-based stability analysis (mainly Sections 4.1 and 5.1, also Section 2.5, Chapter 3 and Appendix B)
- Gaukler, 2020a: Continuization (mainly Sections 4.2 and 5.2, also Section 4.3 and Appendix C)

- Ulbrich and Gaukler, 2019: First concepts on dynamic resource management (high-level idea for Chapter 7)
- Gaukler, Rheinfels, et al., 2019: Theory of convergence rate abstractions (mainly Chapter 6, also Chapter 3)
- Gaukler, 2020b (internal report): Application of convergence rate abstractions (mainly Chapter 7, also Chapters 3 and 6 and Appendix E)

The bibliographical details of these works are listed on page 266. Furthermore, I advised and co-advised of a number of student works as noted on page 267. Many of these student works considered related topics and feasibility studies.

Author Contribution For all of the joint publications listed above, except Ulbrich and Gaukler, 2019, I was the lead author, invented the mathematical approach and contributed the vast majority of theoretical content (over 80 percent) and overall presentation (over 70 percent).

The main contributions of the coauthors were the following: Günter Roppenecker, Andreas Michalka and Tim Rheinfels improved the control-theoretic content. Peter Ulbrich, Tim Rheinfels and Tobias Klaus contributed the real-time systems point of view, such as related work and implementation aspects, and refined the overall presentation.

This doctoral thesis would never have come so far without the invaluable benefit of cooperation: Joint discussions shaped the research agenda and provided priceless ideas, such as the "safety net" and the "quality-of-control model", which then evolved into the concepts presented in this work.

3 Problem Statement

As outlined in the previous chapter, the design of efficient real-time systems is greatly simplified if an application tolerates a certain amount of timing deviation and ideally also skipped executions. The goal of this doctoral thesis is to enable such flexibility for MIMO control systems. This chapter lays the foundations by introducing a formal model of the control loop (Section 3.1) that includes the influence of timing and skips. Then, Section 3.2 specifies mathematical problem settings that will be solved in the subsequent chapters.

3.1 System Model

This section presents a formal description of a real-time control system with uncertain timing. Earlier variants of this formal description were published before (Gaukler, 2020b; Gaukler, Michalka, et al., 2018; Gaukler, Rheinfels, et al., 2019; Gaukler, Roppenecker, et al., 2019; Gaukler, Roppenecker, et al., 2020; Gaukler and Ulbrich, 2019). Here, they are combined and extended to allow for skipped events and nonlinear dynamics.

Notation A full list of variables and notation is given on page xiv. In particular, closed intervals are denoted by $[a; b]$, open intervals by $(a; b)$ and the Cartesian product by $\times$. Furthermore, $v^{(j)}$ is the j-th component of a vector v.

3.1.1 Control Loop

Plant The physical system to be controlled is described as a time-invariant plant, for which the following nonlinear equations and their linear special case, denoted by $\overset{\text{lin}}{=}$, are given.

$$\begin{aligned} \dot{x}_\mathrm{p}(t) &= f_\mathrm{p}(x_\mathrm{p}(t), u(t), d(t)) \overset{\text{lin}}{=} A_\mathrm{p} x_\mathrm{p}(t) + B_\mathrm{p} u(t) + G_\mathrm{p} d(t), \quad t > 0, \\ x_\mathrm{p}(0) &= x_{\mathrm{p},0}, \\ y(t) &= g_\mathrm{p}(x_\mathrm{p}(t), d(t)) \overset{\text{lin}}{=} C_\mathrm{p} x_\mathrm{p}(t) + H_\mathrm{p} d(t). \end{aligned} \tag{3}$$

The plant has state $x_\mathrm{p} \in \mathbb{R}^{n_\mathrm{p}}$, input $u \in \mathbb{R}^m$ and output $y \in \mathbb{R}^p$. Measurement noise and physical disturbance are combined in the generalized disturbance input $d \in \mathbb{R}^{n_\mathrm{dist}}$. If they are not present, it is denoted by $n_\mathrm{dist} = 0$, which means that all dependencies on $d(t)$ are omitted. In the linear case, A_p, B_p et cetera are matrices of appropriate dimension.

The initial state and disturbance are unknown but restricted to the bounded sets $\mathcal{X}_{\mathrm{p},0}$ and $\mathcal{D}$:

$$x_{\mathrm{p},0} \in \mathcal{X}_{\mathrm{p},0}, \tag{4}$$
$$d(t) \in \mathcal{D}. \tag{5}$$

To simplify the discussion, the disturbance $d(t)$ and the functions $f_\mathrm{p}(x_\mathrm{p}, u, d)$ and $g_\mathrm{p}(x_\mathrm{p}, d)$ are assumed to be differentiable.

It is assumed that every sensor and actuator has a one-dimensional signal, so that "the j-th sensor" refers to $y^{(j)}$ and "the j-th actuator" to $u^{(j)}$.

Controller The plant is controlled by the discrete-time controller

$$\begin{aligned} x_{\mathrm{d},k+1} &= f_\mathrm{d}(x_{\mathrm{d},k}, y_k) \overset{\text{lin}}{=} A_\mathrm{d} x_{\mathrm{d},k} + B_\mathrm{d} y_k, \\ u_k &= g_\mathrm{d}(x_{\mathrm{d},k}) \overset{\text{lin}}{=} C_\mathrm{d} x_{\mathrm{d},k}, \quad x_{\mathrm{d},0} = 0, \quad k \in \mathbb{N}_0 \end{aligned} \tag{6}$$

with state $x_\mathrm{d} \in \mathbb{R}^{n_\mathrm{d}}$ and nominal period $T > 0$. This formulation includes standard linear controllers, such as discretized PID or observer-based state feedback. The given form has no feedthrough, which means that the control signal u_k only requires the previous measurement y_{k-1}. This restriction is reasonable for control systems in which timing is relevant, as it permits a computation time of slightly below one control period, while with feedthrough any computation time would inevitably delay the output. For simplicity, a constant set point of $x_\mathrm{p} = 0$, $y = 0$, $u = 0$ is assumed, implying

$$f_\mathrm{d}(0, 0) = 0. \tag{7}$$

3.1.2 Timing

Nominally, the measurement vector y_k is sampled at $t = kT$. Using this data, the control signal u_{k+1} is computed and then emitted at $t = (k+1)T$. The real timing differs from these ideal times by a bounded deviation. Additionally, it is possible to individually skip sampling each sensor component, updating each actuator component and updating the controller. First, the timing deviations will be explained under the assumption that no events are skipped.

IO timing model The value $u_k^{(j)}$ of the j-th actuator is delayed by $\Delta t_{\mathrm{u},k}^{(j)}$, where negative values represent a too early output. The output uses zero-order hold, i.e., the value is constant between updates and the change is instantaneous. Formally,

$$u^{(j)}(t) = \begin{cases} u_k^{(j)}, & kT + \Delta t_{\mathrm{u},k}^{(j)} \le t < (k+1)T + \Delta t_{\mathrm{u},k+1}^{(j)} \quad \wedge \quad k \ge 1, \\ 0, & \text{else.} \end{cases} \tag{8}$$

Respectively, the sample $y_k^{(j)}$ of the j-th sensor is delayed by $\Delta t_{\mathrm{y},k}^{(j)}$:

$$y_k^{(j)} = \begin{cases} y^{(j)}(kT + \Delta t_{\mathrm{y},k}^{(j)}), & k \ge 1, \\ 0, & \text{else.} \end{cases} \tag{9}$$

The "0" case in (9) and (8) is only relevant for start-up and will be discussed later on page 46. Effects beyond timing, e.g., quantization and limited bandwidth, are not considered within this work.

Figure 7 shows the timing of input, computation and output with the corresponding data dependencies (double arrows). As a "black-box" model, the timing deviation of each sensor and actuator is unknown but bounded:

$$\underline{\Delta t}_{\mathrm{u}}^{(j)} \le \Delta t_{\mathrm{u},k}^{(j)} \le \overline{\Delta t}_{\mathrm{u}}^{(j)}, \quad \underline{\Delta t}_{\mathrm{y}}^{(j)} \le \Delta t_{\mathrm{y},k}^{(j)} \le \overline{\Delta t}_{\mathrm{y}}^{(j)} \qquad \forall j \ \forall k \ge 1. \tag{10}$$

Additional constraints, e.g., that a group of sensors is always sampled at the same time, may be introduced to make the model less pessimistic but are not considered here for the sake of simplicity. Similarly, the stochastic distribution of timing within the bounds is not considered in this work, but treated further in Gaukler, Michalka, et al., 2018.

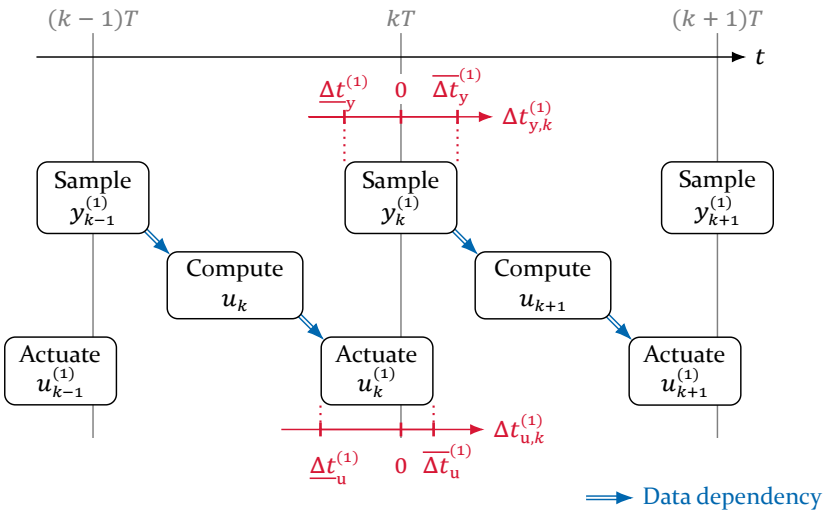


Figure 7: Timing model for IO and computation with the resulting data dependencies: The timing of sampling and actuation may deviate by a bounded amount, resulting in periodic time windows instead of the classical periodic time instants. The illustration only shows a single input and output ($m = p = 1$).

To avoid the need for extra buffers in the realization and corresponding buffer states in the resulting model, the periods k and $k+1$ are separated by a timing barrier $t_{\mathrm{b},k}$, which no event may cross:

$$\left.\begin{matrix} kT + \Delta t_{\mathrm{y},k}^{(j)} \\ kT + \Delta t_{\mathrm{u},k}^{(j)} \end{matrix}\right\} < t_{\mathrm{b},k} < \begin{cases} (k+1)T + \Delta t_{\mathrm{y},k+1}^{(j)} \\ (k+1)T + \Delta t_{\mathrm{u},k+1}^{(j)} \end{cases} \qquad \forall j \;\; \forall k \geq 1. \tag{11}$$

This barrier coincides with the controller computation and can equivalently be described by a task with zero execution time and four virtual dependencies as shown in Figure 8. To simplify the subsequent model,

$$t_{\mathrm{b},k} := \left(k + \frac{1}{2}\right) T \tag{12}$$

is assumed in the following, which means that each control period k is centered at its nominal time kT and all delays are limited to the interval $(-T/2; T/2)$. A generalization is discussed later (Remark 3.1.4 on page 49).

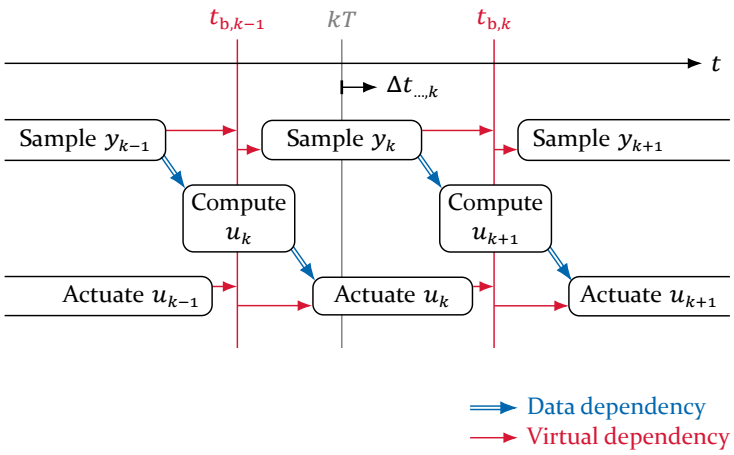


Figure 8: Timing barrier $t_{b,k}$ and its corresponding virtual dependencies. In this example, the sampling and actuation time windows are almost as large as it is possible for a given controller computation time.

Controller Computation Time For simplicity, the model assumes that the controller computation is instantaneous and occurs exactly at $t_{ctrl,k} := t_{b,k} = kT + T/2$, which does not match a real implementation. However, this assumption is effectively without loss of generality: Because updating the controller state has no direct physical impact, the computation timing does not affect the physical state, as long as the data dependencies of the controller are satisfied. This means that computation may start as soon as all measurements are available and may take until the first actuator is updated, without changing the physical behavior of the closed loop. This statement will be formalized later (page 64).

Skipping Additionally, updates may be skipped, either deliberately to save computing and communication resources, or accidentally due to deadline misses or packet loss for networked IO components. This is modeled by one binary variable $\sigma_{u,k}^{(i)} \in \{0, 1\}$ per actuator, where 1 indicates skipping the update of $u^{(i)}$ at time step k and 0 represents normal execution. In the same way, sensors can be skipped by $\sigma_{y,k}^{(i)} = 1$ and the computation of the controller state $x_{d,k+1}$ by $\sigma_{ctrl,k} = 1$.

For simplicity, this work assumes that the control algorithm is agnostic of skips. Skipped data is substituted by the most recent non-skipped value. In particular, if the controller is skipped, then x_d remains constant.

The ratio of skips is bounded by a policy that depends on the specific scenario, which is detailed later. A prominent example is the following:

Definition 3.1.1 ((M,K)-Weak Task (Hamdaoui and Ramanathan, 1995))**.** A (M,K)-weak periodic task is a weakly-hard periodic task with the following guarantee: In any K consecutive periods k, the task σ_x must be executed normally ($\sigma_{x,k}=0$) at least M times, while it may be skipped ($\sigma_{x,k}=1$) in the remaining up to $K-M$ executions:

$$\forall k \geq 0 : \quad \sum_{i=k}^{k+K-1} \sigma_{x,i} \leq K-M. \tag{13}$$

◁

Numbering As noted earlier, the timing barriers divide the time axis into control periods, whose numbering is defined as follows:

Definition 3.1.2. The *period* k, also termed *k-th period*, is the time range $\left(k-\frac{1}{2}\right)T < t \leq \left(k+\frac{1}{2}\right)T$. The *first period* is $k=1$. ◁

With this definition, the period k contains the updates of y_k, u_k and $x_{\mathrm{d},k+1}$. Therefore, all $\sigma_{\mathrm{skip},k}$ and $\Delta t_{\ldots,k}$ have their effect in this k-th period.

Startup Behavior

Definition 3.1.3. The *startup phase* is the time interval $0 \leq t \leq T/2$. ◁

Because time doesn't start at $t=-T/2$ but at $t=0$, the first half of the "0-th control period" $k=0$ is missing and $y_0^{(j)}$ would not be sampled if $\Delta t_{\mathrm{y},0}^{(j)} < 0$. To avoid this discontinuity, sampling y_0 and actuating u_0 are skipped, which is reflected in the "0" case of (8) and (9). Hence, the first proper period has index $k=1$ and starts at $T/2$. Because the startup behavior only affects a short time interval at the start, changing it is roughly equivalent to a small change of the initial set. For the linear case, it is therefore not relevant for asymptotic stability and other infinite-time properties.

Concurrent sampling The model permits that multiple input or output components are sampled at the same time. This was excluded in earlier variants, such as the stochastic average-case analysis of Gaukler, Michalka, et al., 2018, to avoid the mathematical difficulties of concurrent events, which will

be discussed later (page 60). For the worst-case verification presented here, it does not make a difference whether such concurrent sampling is excluded because the same behavior is possible with sequential sampling: The time between two samplings can be arbitrarily short, so that the physical state can be arbitrarily close (assuming f_p, g_p, f_d, g_d are linear or sufficiently regular), and the disturbance can randomly have the same value at these samplings. Therefore, the set of possible trajectories does not change by excluding or including concurrent sampling.

3.1.3 Formalization

The previous explanations are now summarized in compact notation. In the following, a colon denotes definitions, e.g., $a \coloneqq b$ means "a is defined as b".

Timing Vector In the execution of period k, the physical behavior is influenced by the input and output timing deviations

$$\Delta t_k \coloneqq \begin{bmatrix} \Delta t_{\mathrm{u},k}^{(1)} & \dots & \Delta t_{\mathrm{u},k}^{(m)} & \Delta t_{\mathrm{y},k}^{(1)} & \dots & \Delta t_{\mathrm{y},k}^{(p)} \end{bmatrix}^\top, \tag{14}$$

and the skips

$$\sigma_{\mathrm{skip},k} \coloneqq \begin{bmatrix} \sigma_{\mathrm{u},k}^{(1)} & \dots & \sigma_{\mathrm{u},k}^{(m)} & \sigma_{\mathrm{y},k}^{(1)} & \dots & \sigma_{\mathrm{y},k}^{(p)} & \sigma_{\mathrm{ctrl},k} \end{bmatrix}^\top. \tag{15}$$

These are summarized in the generalized execution influence

$$\sigma_k \coloneqq \begin{bmatrix} \Delta t_k \\ \sigma_{\mathrm{skip},k} \end{bmatrix} \in \Sigma \coloneqq (-T/2; T/2)^{m+p} \times \{0,1\}^{m+p+1}, \tag{16}$$

where $\times$ denotes the Cartesian product.

Continuous-Time State Vector For a uniform formulation, all "discrete-time" variables are treated as continuous-time signals that are updated at certain times and remain constant in between (zero-order hold). This leads to the state vector

$$x(t) = \begin{bmatrix} x_\mathrm{p}(t)^\top & x_\mathrm{d}(t)^\top & y_\mathrm{d}(t)^\top & u(t)^\top \end{bmatrix}^\top \in \mathbb{R}^n \tag{17}$$

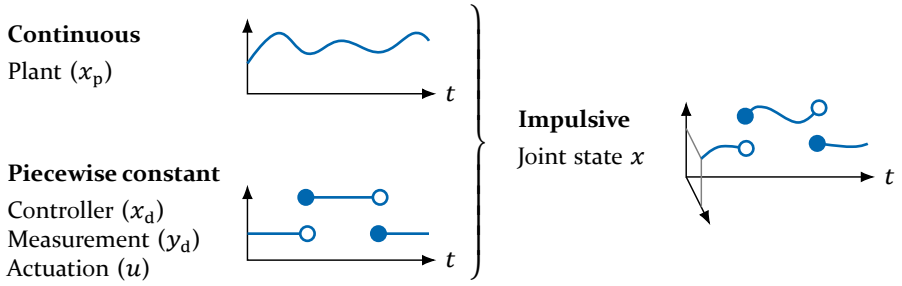


Figure 9: The joint state vector x combines the continuous plant dynamics with the piecewise-constant dynamics of controller, measurement and actuation. This leads to impulsive dynamics that admit both discrete jumps and continuous evolution.

of dimension $n = n_p + n_d + p + m$ consisting of of plant state x_p, controller state x_d, sampled measurement y_d and actuation u. Time starts at $t = 0$ with $x(0) = \left[x_{p,0}^\top \; 0^\top \; 0^\top \; 0^\top\right]^\top$. The discrete-time states x_d, y_d, u are piecewise constant, starting at zero and changing only at certain events: Measurement and actuation states are updated as soon as a sampling or update takes place. As noted earlier, it is assumed without loss of generality that the controller state is updated exactly at the timing barrier. All signals are defined as right continuous ($x(t) = x(t^+)$, where t^+ denotes the limit from above). The combination of continuous plant dynamics and piecewise-constant discrete-time dynamics results in impulsive dynamics for the joint state vector, as illustrated in Figure 9.

In the startup phase ($k = 0$), all sensors and actuators are skipped by definition. The controller has no effect due to (7). Regular execution starts with $k = 1$.

Events The k-th control period ($k \geq 1$) contains the following events:

- The i-th sensor, $i = 1, ..., p$, is sampled only if $\sigma_{y,k}^{(i)} = 0$:

$$y_d^{(i)}(t) = y^{(i)}(t^-) \quad \text{at } t = t_{y,k}^{(i)} \coloneqq kT + \Delta t_{y,k}^{(i)} \quad \text{if } \sigma_{y,k}^{(i)} = 0. \tag{18}$$

  The limit $y^{(i)}(t^-)$ exists because $d(t)$ is assumed to be differentiable.

- The j-th actuator, $j = 1, ..., m$ is updated only if $\sigma_{u,k}^{(j)} = 0$:

$$u^{(j)}(t) = u_k^{(j)} = g_d(x_d(t^-))^{(j)} \quad \text{at } t = t_{u,k}^{(j)} \coloneqq kT + \Delta t_{u,k}^{(j)} \quad \text{if } \sigma_{u,k}^{(j)} = 0. \tag{19}$$

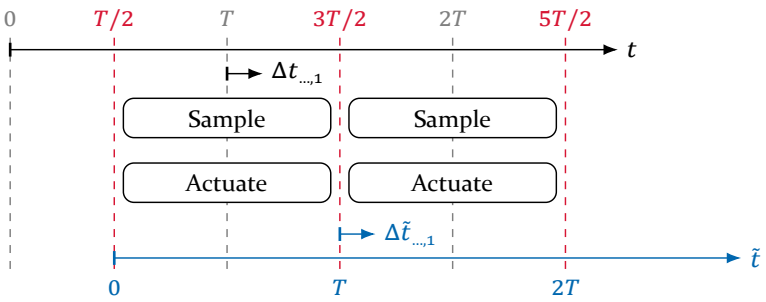


Figure 10: Transformation of the IO timing model by shifting the time axis to $\tilde{t} = t - T/2$.

- Then, the controller state is updated to $x_{\mathrm{d},k+1}$ at the timing barrier $t_{\mathrm{b},k}$ only if $\sigma_{\mathrm{ctrl},k} = 0$:

$$x_{\mathrm{d}}(t) = f_{\mathrm{d}}(x_{\mathrm{d}}(t^-), y_{\mathrm{d}}(t^-)) \quad \text{at } t = t_{\mathrm{ctrl},k} = (k+1/2)T \quad \text{if } \sigma_{\mathrm{ctrl},k} = 0. \tag{20}$$

  For comparison, the discrete-time equivalent is

$$x_{\mathrm{d},k+1} = \begin{cases} f_{\mathrm{d}}(x_{\mathrm{d},k}, y_k), & \sigma_{\mathrm{ctrl},k} = 0, \\ x_{\mathrm{d},k}, & \text{else.} \end{cases} \tag{21}$$

Continuous and Discrete Time For a given time t, there is no general mapping to a unique discrete-time index k. A notable exception is the time $t_{\mathrm{b},k}^- = (k+1/2)T^-$ just before the end of the period k: Then, sampling and actuation of the current period has finished, but the controller is not yet updated, so all discrete-time states are at the value with index k:

$$x(t_{\mathrm{b},k}^-) = \begin{bmatrix} x_{\mathrm{p}}(t_{\mathrm{b},k}^-)^\mathsf{T} & x_{\mathrm{d},k}^\mathsf{T} & y_k^\mathsf{T} & u_k^\mathsf{T} \end{bmatrix}^\mathsf{T}. \tag{22}$$

Remark 3.1.4 (Equivalence of Timing Models). In Section 3.1.2, the boundary between two control periods was chosen as $t_{\mathrm{b},k} = (k+1/2)T$, leading to a symmetric timing model with deviations within $(-T/2; T/2)$.

As illustrated in Figure 10, this is not a restriction: Relabeling the time axis to $\tilde{t} = t - T/2$ can equivalently transform the problem to a timing model with deviations within $(0; T)$. The only difference is the startup phase, corresponding to a change of the initial state.

In general, choosing the symmetric timing model with $t_{\mathrm{b},k} = kT + T/2$ is almost without loss of generality, as its results hold for any other timing model with a deviation range of $(c; c + T)$ for any constant $0 \leq c \leq T$, up to a change of the initial state. ◁

3.1.4 Continuous Approximation

For illustration, it is possible to show a purely continuous variant of the problem: Assuming ideal timing and a negligibly small period T, the discrete-time difference quotient of the controller state x_d is approximated by a continuous-time differential equation. With $t = kT$, this yields

$$\dot{x}_\mathrm{d}(t) \approx \frac{x_{\mathrm{d},k+1} - x_{\mathrm{d},k}}{T} = \frac{f_\mathrm{d}(x_{\mathrm{d},k}, y_k) - x_{\mathrm{d},k}}{T} \overset{\text{lin}}{=} \frac{A_\mathrm{d} - I}{T} x_\mathrm{d}(t) + \frac{B_\mathrm{d}}{T} y(t). \quad (23)$$

The resulting closed loop does not require states for measurement and actuation and is given by

$$\begin{bmatrix} \dot{x}_\mathrm{p}(t) \\ \dot{x}_\mathrm{d}(t) \end{bmatrix} \overset{\text{lin}}{\approx} \begin{bmatrix} A_\mathrm{p} & B_\mathrm{p} C_\mathrm{d} \\ B_\mathrm{d} C_\mathrm{p}/T & (A_\mathrm{d} - I)/T \end{bmatrix} \begin{bmatrix} x_\mathrm{p}(t) \\ x_\mathrm{d}(t) \end{bmatrix} + \begin{bmatrix} G_\mathrm{p} \\ B_\mathrm{d} H_\mathrm{p}/T \end{bmatrix} d(t). \quad (24)$$

Note that the stability of this approximation is neither sufficient nor necessary for the stability of the original control loop. However, modeling the approximation error as bounded disturbance leads to a method for safety analysis that will be explored later in Section 5.2.

3.1.5 Example Systems

Appendix A provides data for several examples ranging from a minimal, one-dimensional system to the simplified model of a quadrotor helicopter with three-axis angular rate control. They will be used throughout this work to compare and illustrate analysis techniques.

The names of the systems (e.g., A1) are consistent throughout this thesis and the corresponding previous publications (cf. page 266). Letters indicate the base system and numbers denote variants such as increased timing deviation.

Figure 11 shows the effect of uncertain timing on the initial response of the aforementioned quadrotor control. As detailed in Appendix A.3, this system is based on a quasi-continuous controller design with a sample rate chosen such that the initial response deviates by approximately 20 percent from that of a controller with infinite sample rate.

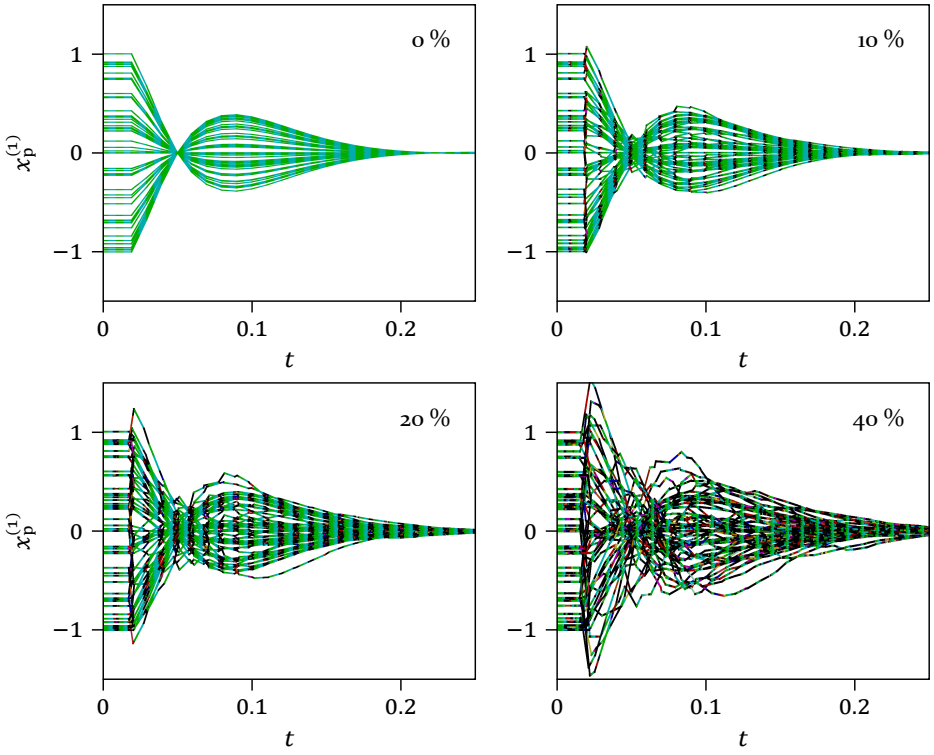


Figure 11: Simulation of an example system for different timing deviations. The timing deviation of $\pm c$ percent of the control period is varied from $c = 0$ to $c = 40$. The linearized three-axis angular rate control of a quadrotor helicopter (Example D3 from Appendix A.3) is simulated with random initial states, uniformly random time-varying sampling and actuation timing and zero disturbance. Each plot shows multiple simulation runs with different random values. The plots show the state $x_p^{(1)}(t)$, which is the normalized angular velocity of the helicopter around the x-axis. The colors depict the stages of the sample-compute-actuate cycle.

It can be seen that increasing the timing deviation gradually disturbs and destabilizes the system. Still, a certain amount of uncertainty can be accepted, which is beneficial for the design of the real-time computing system. The simulations also illustrate that the large majority of random trajectories is much less affected by timing than a small number of outliers. Hence, random simulations can easily miss the worst case. In contrast, the formal verification methods investigated in the following chapters guarantee full coverage of all possible timing combinations.

3.2 Problem Settings

For safety-critical real-time control, it is desirable to automatically validate the correctness of the system, e.g., to prove that an unmanned aerial vehicle (UAV) does not crash. This can be formalized by requirements on the model of the control loop, such as state bounds or stability. However, Section 2.5 showed that for the important MIMO case it is not yet possible to translate these requirements from control engineering to the language of real-time systems design. Therefore, one has to resort to vague statements such as "sufficiently good timing" or to the unreasonable demand of ideal timing. The vision of this doctoral thesis is to provide clear answers in terms of time windows and permitted skips, so that the controller only demands as much timing precision and resources as it actually needs. In the following, this vision is translated into formal problems.

Given A real-time control system with uncertain but bounded IO timing and disturbance is given, as modeled in Section 3.1.

Requirements For disturbance up to the specified maximum amplitude, the system must obey a physical safety requirement, which is modeled by a bound on the physical state. As an alternative requirement, exponential stability can be considered, which mandates exponential convergence of the state magnitude if there is no disturbance.

Goal Two different co-design scenarios are considered (cf. Section 2.4). For *design-time analysis*, the goal is a worst-case proof: Prove that the requirements hold for any execution sequence within given bounds for the timing deviation and skip rate. This can be used to determine fixed deadlines and release times of the real-time system.

In contrast, *dynamic resource management* occurs at run-time: The largest permissible range of timing and skips for the next period must be computed, given the actual execution in the previous periods. This dynamic strategy can allow for higher peak timing deviations than possible with a fixed design-time bound. However, it incurs computational overhead at run-time.

These concepts are formalized in the following.

3.2.1 Safety Requirement

Physical safety requirements, such as that a UAV holds its position with an accuracy of one meter, are expressed by a virtual safety output

$$s_k \coloneqq C_\mathrm{s} x_\mathrm{p}(kT - T/2) \in \mathbb{R}^{n_\mathrm{s}} \tag{25}$$

whose magnitude must be below a given bound:

$$|s_k| \leq |s|_\mathrm{max} \quad \forall k \geq 1. \tag{26}$$

The specification is restricted to discrete time points $t = kT - T/2$ at the start of each control period to simplify the derivations. In future work, this may be extended by considering the possible overshoot within one control period.

Neglecting sampling, this specification is equivalent to defining an ellipsoid

$$\{x_\mathrm{p} \in \mathbb{R}^{n_\mathrm{p}} \mid x_\mathrm{p}^\mathsf{T} C_\mathrm{s}^\mathsf{T} C_\mathrm{s} x_\mathrm{p} < |s|_\mathrm{max}^2\}. \tag{27}$$

of safe states x_p. In the literature, the term *practical stability* is used to denote similar properties where the system state is bounded to a set, while in this work, the term stability is reserved for exponential convergence.

There are no further restrictions on the specification. For example, non-measurable states can also be included, e.g., $C_\mathrm{s} = I$ denotes a bound $|x_\mathrm{p}| \leq |s|_\mathrm{max}$ on the whole physical state vector, regardless if it is measurable. Also, the ellipsoid can be unbounded. For example, $C_\mathrm{s} = e_1^\mathsf{T}$ models the restriction $|x_\mathrm{p}^{(1)}| < |s|_\mathrm{max}$ on the first component of the physical state, while the safe set is unbounded in all other directions.

3.2.2 Stability Requirement

For analysis techniques that do not consider disturbance, an appropriate goal is to show exponential stability:

Definition 3.2.1. A dynamical system with state x and initial value $x(t_0)$ admits *continuous-time globally uniform exponential stability*, denoted as CGES(λ, α), if and only if there exist constants $\alpha \in [1; \infty)$ and $\lambda < 0$ such that for all possible trajectories

$$|x(t)| \leq \alpha |x(t_0)| \mathrm{e}^{\lambda(t-t_0)} \quad \forall t \geq t_0,\ \forall x(t_0) \in \mathbb{R}^n. \tag{28}$$

◁

If not mentioned otherwise, this definition refers to the closed control loop without disturbance and measurement noise: The state vector x per (17) must decay exponentially for all permitted timing sequences.

3.2.3 Design-Time Analysis

The subsequent chapters present multiple variants of design-time analysis:

- Section 5.1: CGES analysis for uncertain timing and no skips.
- Section 5.2: Safety analysis and state bounds for ideal timing, with an outlook on a generalization.
- Section 6.6: CGES analysis for a (M, K) skip rate, assuming ideal timing and that either all or no events are skipped in one period.

3.2.4 Dynamic Resource Management

As discussed in Section 2.4, the goal of dynamic resource management is to provide dynamically relaxed real-time constraints for the control loop while strictly guaranteeing the safety specification.

Given After the period $k-1$, the execution σ_{k-1} of that period is known, including skips and timing deviation. In this thesis, perfect measurement of these execution conditions is assumed for simplicity, which can however be relaxed with minor changes. The measurement requires an appropriate precision timer mechanism in the real-time operating system.

Goal The goal is to determine a range Σ_k of permissible executions $\sigma_k \in \Sigma_k$ for the coming period k, i.e., bounds on the timing deviation Δt_k and possible skips $\sigma_{\text{skip},k}$. This range must guarantee the safety specification (26) and should be large according to some metric such as the possible reduction in processor load.

The decision policy must be implemented with little computational effort so that its benefits outweigh the run-time cost. In particular, this excludes storing the full information about all previous executions, as this would require unbounded memory. Another difficulty is the high number of decision variables for systems with multiple sensors and actuators.

An implicit requirement is *recursive feasibility*: A decision taken now must not render it impossible to guarantee safety in the future. This requires predictive decisions, as for systems with mechanical inertia the effects of bad execution on physical safety will usually appear with significant delay. For example, a UAV has limited acceleration and therefore does not instantaneously jump out of place if the actuation timing is bad. Instead, the velocity error gradually increases, which slowly accumulates as position error until the UAV is about to violate a required safe position range. At this point, switching back to good actuation timing can no longer prevent the violation: As the existing velocity must be gradually reversed first, the position overshoots beyond the safe set.

The idea of dynamic resource management is implemented later:

- Chapter 6: For uncertain timing and skips, a time-varying bound on $|s_k|$ is determined by a dynamical model called *convergence rate abstraction*.
- Chapter 7: Based on this bound, decisions for skipping events or for adjusting time windows are computed.

3.3 Concluding Remarks

This chapter formalized the influence of IO timing and skipped executions on the behavior of a MIMO real-time control system. Multiple problems were posed: First, the design-time analysis of worst-case stability and safety. Second, dynamic resource management, which decides at run-time how much timing deviation is permitted without violating the safety specification. These will be addressed in the remainder of this doctoral thesis.

The model is restricted to systems whose timing differs from a periodic grid by less than half of a period. It avoids some details to allow for feasible analysis. Future research could consider extending the model and the corresponding analysis: Particularly for network-controlled systems, interesting extensions include larger delays and more general multi-rate timing schemes. Another aspect is non-ideal sample-and-hold due to the finite rise-time of electrical circuits. Furthermore, the black-box model of time windows could be replaced by a model of the scheduling mechanism to allow for more precise analysis.

4 Theoretical Framework

The system model of Section 3.1 offers a precise description but is difficult to handle in formal analysis. This chapter presents two frameworks in which the model can be reformulated and analyzed: Linear impulsive systems (Section 4.1) are specialized to the linear case and have restricted expressiveness. In contrast, hybrid automata (Section 4.2) are more general, so that they support nonlinear systems and can include timing specifications or skip bounds.

Figure 12 gives an overview of how these frameworks will be applied in the subsequent chapters of this thesis: For the linear case, the linear impulsive system model is converted to a discrete-time system, whose stability is analyzed using a Lyapunov Function. This leads to design-time stability analysis for uncertain timing.

For the nonlinear case, the system is modeled as a hybrid automaton, from which in theory an upper bound on the set of reachable (possible) states x can be directly determined by reachability analysis. In practice, however, this bound turns out to be too pessimistic, which can be solved by first applying continuization, a transformation that leads to a continuous-time system. The resulting bound on the reachable set of x can be used to prove the safety specification at design time.

An alternative approach to obtaining upper bounds are convergence rate abstractions, which can be determined from a Lyapunov function. In the linear case, this Lyapunov function results from the previous stability analysis. The obtained bounds are time-varying and can be used at run-time for dynamic resource management but also at design-time for (M, K)-weak execution.

4.1 Linear Impulsive Systems

Linear impulsive systems (LIS) combine continuous-time dynamics and discrete events. After introducing notation and definitions in Section 4.1.1, the formalism of LIS will be defined and applied to the system model (Sections 4.1.2 and 4.1.3). Finally, a time discretization is applied to derive a discrete-time model (Section 4.1.4).

Previous versions of this section were published in Gaukler, Michalka, et al., 2018; Gaukler, Roppenecker, et al., 2019; Gaukler, Roppenecker, et al., 2020.

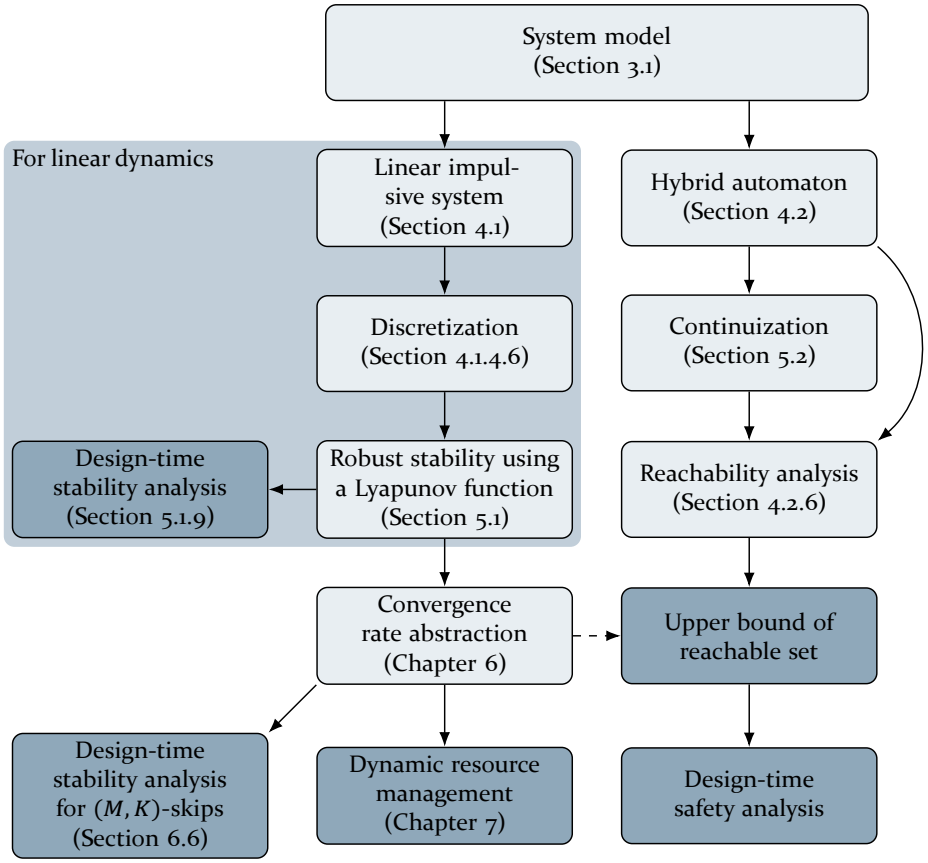


Figure 12: Overview of this thesis, showing methods (light) and their results (dark)

4.1.1 Preliminaries and Notation

For formal clarity and to allow referencing in subsequent proofs, this section presents relevant definitions and results from linear algebra, mostly matching the standard work of Bernstein, 2009.

For a matrix $A \in \mathbb{R}^{n\times n}$, the eigenvalues are denoted $\lambda_i\{A\}$ and the spectral radius is $\rho\{A\} \coloneqq \max_i |\lambda_i|$. A full list of notation can be found on page xiv.

Definition 4.1.1 (Norm)**.** The function $\|\cdot\| : \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ is a vector norm on $\mathcal{S} = \mathbb{R}^n$ or a matrix norm on $\mathcal{S} = \mathbb{R}^{m\times n}$ if and only if $\forall x, y \in \mathcal{S}, \alpha \in \mathbb{R}$

$$\|x\| \begin{cases} > 0, & x \neq 0, \\ = 0, & x = 0, \end{cases} \tag{29a}$$

$$\|\alpha x\| = |\alpha|\|x\| \tag{29b}$$
$$\text{and} \quad \|x + y\| \leq \|x\| + \|y\| \tag{29c}$$

(Bernstein, 2009, pp. 597, 601). ◁

Definition 4.1.2 (Euclidean Vector Norm)**.** For $x \in \mathbb{R}^n$, $|x| \coloneqq \sqrt{x^\top x}$ denotes the Euclidean norm, which is a vector norm. ◁

Theorem 4.1.3 (Equivalence of Norms)**.** *All norms $\|\cdot\|_a$, $\|\cdot\|_b$ defined on the same space ($\mathbb{R}^n$ or $\mathbb{R}^{m\times n}$) are equivalent up to a bounded factor:*

$$\forall \|\cdot\|_a, \|\cdot\|_b \quad \exists c_1, c_2 > 0 : \quad \forall x \quad c_1\|x\|_b \leq \|x\|_a \leq c_2\|x\|_b \tag{30}$$

(Bernstein, 2009, Theorem 9.1.8, Definition 9.2.1).

Definition 4.1.4 (Submultiplicative Matrix Norm)**.** A matrix norm $\|\cdot\|$ on $\mathbb{R}^{n\times n}$ is submultiplicative if and only if

$$\|XY\| \leq \|X\|\|Y\| \quad \forall X, Y \in \mathbb{R}^{n\times n} \tag{31}$$

(Bernstein, 2009, p. 604). ◁

In this thesis, submultiplicativity is only considered for square matrices. Note that in some literature, submultiplicativity is required in the definition of a matrix norm, while here it is not.

Definition 4.1.5 (Equi-Induced Matrix Norm)**.** Every vector norm $\|\cdot\|_v$ on $\mathbb{R}^n$ leads to a corresponding *equi-induced matrix norm* $\|\cdot\|'_v$ on $\mathbb{R}^{n\times n}$,

$$\|A\|'_v \coloneqq \max_{x\in\mathbb{R}^n\setminus\{0\}} \frac{\|Ax\|_v}{\|x\|_v} = \max_{x\in\mathbb{R}^n \text{ with } \|x\|_v=1} \|Ax\|_v, \tag{32}$$

which is submultiplicative (Bernstein, 2009, pp. 607 f.). The prefix "equi-" denotes that A is square. ◁

Definition 4.1.6 (Spectral Norm)**.** The spectral norm

$$\|A\|_2 \coloneqq \rho\{A^\top A\} = \max_{x\in\mathbb{R}^n\setminus\{0\}} \frac{|Ax|}{|x|} = \max_{x \text{ with } |x|=1} \underbrace{\sqrt{x^\top A^\top A x}}_{|Ax|} \tag{33}$$

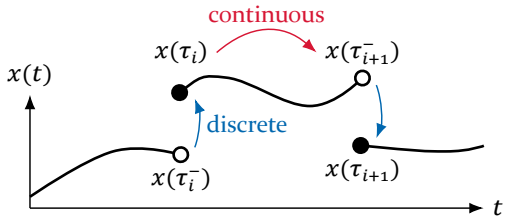


Figure 13: Trajectory of an exemplary LIS.

of $A \in \mathbb{R}^{m\times n}$ is the maximum singular value of A. It describes the maximum growth of the euclidean norm due to multiplication with A. For square matrices ($m = n$), it is the equi-induced matrix norm of the euclidean vector norm and therefore submultiplicative (Bernstein, 2009, pp. 328, 603, 607–609). ◁

Theorem 4.1.7 (Norm Bound of Matrix Exponential)**.** *If $\|\cdot\|$ is a submultiplicative matrix norm on $\mathbb{R}^{n\times n}$ with $\|I\| = 1$, then*

$$\|e^{A\delta}\| \overset{(*)}{\leq} e^{\|A\delta\|} \overset{(29b)}{=} e^{\|A\|\|\delta|} \quad \forall A \in \mathbb{R}^{n\times n}, \delta \in \mathbb{R} \tag{34}$$

The requirement $\|I\| = 1$ is fulfilled for all equi-induced norms, which includes the spectral norm. (Eq. (*) is due to Bernstein, 2009, Prop. 11.1.2, Thm. 9.4.2.)

4.1.2 Definition

A simple definition of a linear impulsive system with state $x \in \mathbb{R}^n$ is

$$\dot{x}(t) = A_{\text{cont}}x(t), \quad t \neq \tau_i, \quad t > \tau_0, \tag{35a}$$
$$x(t) = E_i x(t^-), \quad t = \tau_i, \quad i \in \mathbb{N}_1, \tag{35b}$$
$$x(\tau_0) = x_0, \qquad \tau_0 < \tau_1 < \tau_2 < \dots. \tag{35c}$$

As illustrated in Figure 13, $A_{\text{cont}} \in \mathbb{R}^{n\times n}$ models continuous dynamics, which are interrupted by discrete events $E_i \in \mathbb{R}^{n\times n}$ at $t = \tau_i$. The trajectory is right-continuous, i.e., $x(t^+) = x(t)$, matching the model of Section 3.1.

Extension to Concurrent Events This definition can not handle concurrent events $\tau_i = \tau_{i+1}$, which is a problem for the basic case of ideal timing: In this case, all measurements and actuator updates occur at the same time $t = kT$. Therefore, the definition must be extended such that $\tau_{i+1} = \tau_i$ is permitted and leads to the same result as the right-side limit $\tau_{i+1} \to \tau_i^+$.

Definition 4.1.8 (LIS with Concurrent Events)**.** A more appropriate generalized definition is the following: A LIS is given by

- an initial time τ_0 and state x_0,
- continuous dynamics A_{cont},
- a number N_{ev} of events that may be infinite,
- a series of discrete events $i = 1, \dots, N_{\text{ev}}$ with event matrix E_i and time τ_i, ordered by $\tau_0 \leq \tau_1 \leq \dots \leq \tau_{N_{\text{ev}}}$.

To simplify the notation, either the series of events is infinite ($N_{\text{ev}} = \infty$) or a pseudo-event at time $\tau_{N_{\text{ev}}+1} \coloneqq \infty$ is introduced.

The solution of the LIS at time $t \geq \tau_0$ is defined as

$$\begin{aligned} x(t) &= \mathrm{e}^{A_{\text{cont}}(t-\tau_N)} E_N \mathrm{e}^{A_{\text{cont}}(\tau_N-\tau_{N-1})} \\ &\quad \cdot E_{N-1} \mathrm{e}^{A_{\text{cont}}(\tau_{N-1}-\tau_{N-2})} \dots E_1 \mathrm{e}^{A_{\text{cont}}(\tau_1-\tau_0)} x_0 \qquad (36) \\ &= \mathrm{e}^{A_{\text{cont}}(t-\tau_N)} \left(\widetilde{\prod_{i=1}^{N}} E_i \mathrm{e}^{A_{\text{cont}}(\tau_i-\tau_{i-1})} \right) x_0 \qquad (37) \end{aligned}$$

with N depending on t such that $\tau_N \leq t < \tau_{N+1}$. Note the distinction between N and N_{ev}. $\widetilde{\prod}$ is the reversed product, given by the subsequent definition. ◁

Importantly, this result allows to compute a discrete-time dynamics matrix that accounts for timing uncertainties and skipped events.

Definition 4.1.9 (Reversed Product)**.** For $a, b \in \mathbb{Z}$, the reversed product $\widetilde{\prod}$ is defined as

$$\widetilde{\prod_{i=a}^{b}} X_i \coloneqq \prod_{i=-b}^{-a} X_{-i} = \begin{cases} X_b X_{b-1} \dots X_{a+1} X_a, & a \leq b, \\ I, & a > b. \end{cases} \qquad (38)$$

◁

Remark 4.1.10 (Interpreting a LIS as an Algorithm). The solution (37) of the LIS can be restated as the following algorithm, which is later shown to represent a hybrid automaton.

1. Start at $i = 0$, $t = \tau_0$, $x(\tau_0) = x_0$.

2. Compute $x(t)$ for $\tau_i < t \leq \tau_{i+1}$ as solution of $\dot{x}(t) = A_{\text{cont}} x(t)$ with known initial value $x(\tau_i)$. (For concurrent events, i.e., $\tau_{i+1} = \tau_i$, this step has no effect.)
3. Set $x(\tau_{i+1}) \coloneqq E_{i+1} x(\tau_{i+1})$ and then set $i \coloneqq i + 1$. Go back to step 2. ("Set" refers to overwriting the previous value, analogous to updating a variable in usual, imperative programming languages.) ◁

Proof Sketch. Follow the algorithm to see that it is consistent with the evolution of (37). Special treatment is required for the case of concurrent events, where $x(\tau_{i+1})$ overwrites $x(\tau_i)$ and correspondingly $N(t)$ never equals i. □

4.1.3 System Model

A subset of the system model defined in Chapter 3 can be rewritten in the framework of linear impulsive systems, similar to the derivations in Rheinfels, 2019 and own prior work (Gaukler, 2015; Gaukler, Michalka, et al., 2018; Gaukler, Roppenecker, et al., 2019; Gaukler, Roppenecker, et al., 2020).

Assumptions Assume linear dynamics without disturbance and measurement uncertainty ($n_{\text{dist}} = 0$). Disturbance will be added later in Section 4.1.4.4.

Notation In the following, all dynamics matrices are given as 4×4 block matrices, partitioned according to the four parts of the state vector

$$x(t) = \begin{bmatrix} x_{\text{p}}(t) \\ x_{\text{d}}(t) \\ y_{\text{d}}(t) \\ u(t) \end{bmatrix} \in \mathbb{R}^n, \quad n = n_{\text{p}} + n_{\text{d}} + p + m. \tag{39}$$

with dimensions n_{p}, n_{d}, p, m.

Continuous Dynamics The plant dynamics $\dot{x}_\mathrm{p} = A_\mathrm{p}x_\mathrm{p} + B_\mathrm{p}u$ are continuous. All other variables are constant between the discrete events, e.g., $\dot{x}_\mathrm{d} = 0$:

$$A_\mathrm{cont} = \begin{bmatrix} A_\mathrm{p} & 0 & 0 & B_\mathrm{p} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \mathrm{e}^{A_\mathrm{cont}\delta} = \begin{bmatrix} \mathrm{e}^{A_\mathrm{p}\delta} & 0 & 0 & \tilde{B}(\delta) \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

$$\forall \delta \in \mathbb{R}, \text{ with } \tilde{B}(\delta) \coloneqq \int_0^\delta \mathrm{e}^{A_\mathrm{p}\xi}\,\mathrm{d}\xi\, B_\mathrm{p}. \tag{40}$$

Discrete Events By Definition 3.1.2, the k-th control period is the time range $(k - 1/2)T < t \leq (k + 1/2)T$. Within this period, all sensors and actuators are updated near $t = kT$: In LIS notation, the actuator update $u^{(i)}(t^+) = (C_\mathrm{d}x_\mathrm{d}(t))^{(i)}$ is

$$E_{\mathrm{u},i} \coloneqq I + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \underset{m\times m}{S_i}\, C_\mathrm{d} & 0 & -\underset{m\times m}{S_i} \end{bmatrix}, \quad t^{(i)}_{\mathrm{u},k} = kT + \Delta t^{(i)}_{\mathrm{u},k}, \tag{41}$$

with the selector matrices

$$S_i \coloneqq e_i e_i^\mathsf{T} = \mathrm{diag}(\underbrace{0, \ldots, 0}_{i-1 \text{ times}}, 1, 0, \ldots, 0) \tag{42}$$

of appropriate dimension. Similarly, measurement is modeled by

$$E_{\mathrm{y},i} \coloneqq I + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \underset{p\times p}{S_i}\, C_\mathrm{p} & 0 & -\underset{p\times p}{S_i} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad t^{(i)}_{\mathrm{y},k} = kT + \Delta t^{(i)}_{\mathrm{y},k}. \tag{43}$$

The index "k" of the event times will later be omitted for better readability.

Just before the end of the control period, at $t = (k + 1/2)T$, the new controller state is computed instantaneously from the recent measurements:

$$E_{\text{ctrl}} \coloneqq \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & A_{\text{d}} & B_{\text{d}} & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}, \quad t_{\text{ctrl},k} = (k + 1/2)T. \tag{44}$$

Effect of Controller Timing As claimed in Section 3.1.2, the actual timing of the controller computation may deviate from the assumed $t_{\text{ctrl},k} = (k + 1/2)T$ without changing the physical behavior, as long as the order of events remains unchanged. This serves as a first example of how the LIS model encodes the problem structure:

Consider the solution given by (37), let $\delta > 0$ and see that

$$E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}\delta} \overset{(40),(44)}{=} \mathrm{e}^{A_{\text{cont}}\delta} E_{\text{ctrl}}. \tag{45}$$

This proves that shifting the controller computation forward from $t_{\text{ctrl},k}$ to $t_{\text{ctrl},k} + \delta$ does not change the state $x(t_{\text{ctrl},k} + \delta)$, as long as δ is small enough so that the order of events is preserved. Between $t_{\text{ctrl},k}$ and $t_{\text{ctrl},k} + \delta$, this change also has no physical effect because it only changes x_{d}, as can be seen from the sparsity pattern of $\mathrm{e}^{A_{\text{cont}}}$ and E_{ctrl}.

By an analogous argument, the same result follows for shifting backwards.

4.1.4 Discretization

In general, neither the plant nor the controller is stable in isolation. Stability of the real-time control system therefore only comes from the combination of discrete and continuous evolution. This motivates that stability analysis is applied to the transition from one period to the next, considering

$$x_k \coloneqq x(t_k), \quad \text{where} \quad t_k \coloneqq t_{\text{b},k-1} \overset{(12)}{=} (k - 1/2)T, \tag{46}$$

as discrete-time state. This leads to a linear discrete-time system $x_{k+1} = A_k x_k$. Note that disturbance is not considered for now.

4.1.4.1 Choice of Sampling Time

By (46), the discretization time t_k is chosen as the timing barrier, i.e., just after the controller execution $x_{\mathrm{d},k}$. The benefit of this choice is that the transition matrix $A_k = A(\sigma_k)$ depends only on the execution σ_k of the *current* period, while for $t_k = kT$ the timing σ_k would have effects on A_k and A_{k-1}. In cases where no skips occur, A_k only depends on the timing deviations Δt_k and is abbreviated as

$$A(\Delta t) \coloneqq A(\sigma)\Big|_{\sigma=\left[\begin{smallmatrix}\Delta t\\ 0\end{smallmatrix}\right]} \tag{47}$$

with slight abuse of notation.

Discussion A downside of the chosen discretization time is that x_0 refers to $t_0 = -T/2 < 0$ and therefore a special treatment of the case $k = 0$ is required. Regular discrete-time evolution starts from $k = 1$ at the state $x_1 = \mathrm{e}^{A_{\mathrm{cont}}T/2}x(0)$. As a simplifying assumption, this is neglected, starting discrete-time evolution at $k = 0$ with $x_0 \approx x(0)$. In accordance with (6), (8) and (9), all states except that of the plant are initialized as zero, so

$$x_0 = \begin{bmatrix} x_{\mathrm{p},0} \\ 0 \end{bmatrix}. \tag{48}$$

As noted in Section 3.1.3, the mapping from the time t to the index k of $x_{\mathrm{d},k}$, y_k and $u_{\mathrm{d},k}$ is nontrivial. With the chosen discretization times,

$$x_k = \begin{bmatrix} x_{\mathrm{p}}\left(kT - \frac{T}{2}\right) \\ x_{\mathrm{d},k} \\ y_{k-1} \\ u_{k-1} \end{bmatrix} \tag{49}$$

holds, assuming there are no skips and $k \geq 1$.

4.1.4.2 Transition Matrix

Using (37) with $t = t_{k+1}$ and $\tau_0 = t_k$, the transition matrix can be determined as

$$x_{k+1} = \underbrace{\mathrm{e}^{A_{\mathrm{cont}}(t_{k+1}-\tau_N)}\left(\overleftarrow{\prod_{i=1}^{N}} E_i \mathrm{e}^{A_{\mathrm{cont}}(\tau_i-\tau_{i-1})}\right)}_{A(\sigma_k)} x_k \tag{50}$$

with the following parameters: The period starts at the time

$$\tau_0 \coloneqq t_k \overset{(46)}{=} t_{\mathrm{b},k-1} \overset{(12)}{=} kT - T/2, \tag{51}$$

just after the previous controller update has been completed. The period contains one event for the controller and for each sensor and actuator, except for events that were skipped. It ends at

$$\tau_N = t_{\mathrm{b},k} \overset{(12)}{=} kT + T/2 \tag{52}$$

just after the controller update to $x_{\mathrm{d},k+1}$ and contains $N \coloneqq N_{\mathrm{ev},k}$ events.

Formally, the set of events (E_i, τ_i) in the period is

$$\begin{aligned}
\mathcal{E}_k &\coloneqq \left\{(E_i, \tau_i) | i = 1, \dots, N_{\mathrm{ev},k}\right\} \text{ with } \tau_i :\leq \tau_{i+1} && (53)\\
&= \left\{(E_{\mathrm{u},i}, t_{\mathrm{u},k}^{(i)}) | i = 1, \dots, m, \quad \sigma_{\mathrm{u},k}^{(i)} = 0\right\}\\
&\quad \cup \left\{(E_{\mathrm{y},i}, t_{\mathrm{y},k}^{(i)}) | i = 1, \dots, p, \quad \sigma_{\mathrm{y},k}^{(i)} = 0\right\}\\
&\quad \cup \left\{(E_{\mathrm{ctrl}}, t_{\mathrm{ctrl},k}) | \text{ if } \sigma_{\mathrm{ctrl},k} = 0\right\}, && (54)
\end{aligned}$$

$$|\mathcal{E}_k| \coloneqq N_{\mathrm{ev},k} \coloneqq m + p + 1 - \underbrace{\left(\sum_{i=1}^{m} \sigma_{\mathrm{u},k}^{(i)} + \sum_{i=1}^{p} \sigma_{\mathrm{y},k}^{(i)} + \sigma_{\mathrm{ctrl},k}\right)}_{\text{number of skipped events}}, \tag{55}$$

which means that events in each period are numbered as $i = 1, \dots, N_{\mathrm{ev},k}$ according to their temporal order and that all events occur exactly once.

While the order of events with equal time τ_i is ambiguous, this is no problem since the following theorem guarantees that all possible orders lead to the same trajectory. Thus, an arbitrary order can be chosen without loss of generality.

Theorem 4.1.11. *The order of actuation and/or measurement events occurring at the same time $\tau_i = \tau_{i+1}$ does not change the system dynamics.*

Proof. Consider the trajectory (37) of the linear impulsive system. If the i-th and $(i+1)$-th event occur at the same time $\tau_i = \tau_{i+1}$, this yields a trajectory $x(t) = \cdots E_{i+1}E_i \cdots$. Reversing the order of these events changes the trajectory to $x(t) = \cdots E_iE_{i+1} \cdots$. As will be shown later in (347), $E_{i+1}E_i = E_iE_{i+1}$ holds for all measurement and actuation event matrices E_i, E_{i+1}, so the trajectory remains unchanged. □

If the controller is not skipped ($\sigma_{\text{ctrl},k} = 0$), the transition matrix becomes

$$A(\sigma_k) = E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev},k}} - \tau_{N_{\text{ev},k}-1})} \overbrace{\prod_{i=1}^{N_{\text{ev},k}-1}} E_i \mathrm{e}^{A_{\text{cont}}(\tau_i - \tau_{i-1})}. \tag{56}$$

4.1.4.3 Discretization and Stability

Stability of the discretized system is easier to analyze but still *equivalent* to the desired continuous-time stability:

Definition 4.1.12. A discrete-time dynamical system with state x_k admits discrete-time globally uniform exponential stability, denoted DGES(ρ, α), if and only if there exist constants $\alpha \in [1; \infty)$ and $\rho \in (0; 1)$ such that

$$|x_k| \leq \alpha |x_0| \rho^k \quad \forall k \geq 0, \forall x_0 \in \mathbb{R}^n \tag{57}$$

for all trajectories. ◁

If not otherwise mentioned, DGES in this thesis refers to the discretized control loop without disturbance,

$$x_{k+1} = A(\sigma_k) x_k, \tag{58}$$

for all permissible execution sequences $\sigma_{0,1,\ldots}$.

Theorem 4.1.13. *For the given linear real-time control system, CGES ⇔ DGES.*

The proof requires an intermediate result, which is illustrated in Figure 14:

Lemma 4.1.14. *During one control period, the growth of the state x of the closed control loop is bounded: There exist constants $\bar{\alpha} \geq 1, \bar{\lambda} \in \mathbb{R}$ such that for all periods $k \geq 0$ and for all $x(t_k) \in \mathbb{R}^n$,*

$$|x(t_k + \delta)| \leq \bar{\alpha} \mathrm{e}^{\bar{\lambda}\delta} |x(t_k)| \quad \forall \delta \in [0; T). \tag{59}$$

Note that this is not a stability result: Any sampled-data control effectively runs in open loop between the sampling instants, so $\bar{\lambda} \geq 0$ if the uncontrolled plant is unstable. To see that this must be true, consider the case when the plant state is nonzero, i.e., $x_{\text{p}}(t_k) \neq 0$, and all other entries of $x(t_k)$ are zero.

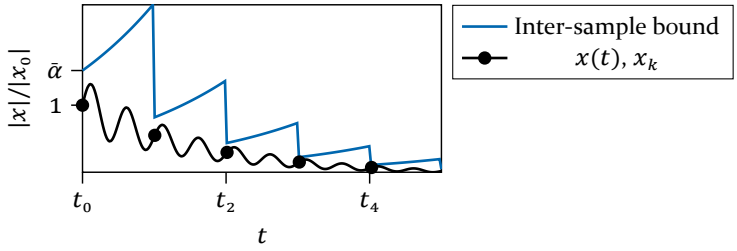


Figure 14: Lemma 4.1.14 bounds the overshoot between samples, so that discrete-time stability leads to continuous-time stability.

Proof. The result is trivially true for $\delta = 0$. Next, assume $0 < \delta < T$. Because all the event matrices from Section 4.1.3 are constant and finite, there is a finite $\alpha_{\text{ev}} \geq 1$ such that

$$\alpha_{\text{ev}} \geq \|E\|_2 \quad \forall E \in \{E_{\text{ctrl}}, E_{\text{u},1}, \dots, E_{\text{u},m}, E_{\text{y},1}, \dots, E_{\text{y},p}\}. \tag{60}$$

Consider (37) with $N \in \{0, \dots, m+p\}$ as the number of events in $(t_k; t_k + \delta]$. Note that by (53), the events are numbered such that the first event after $t = \tau_0 \coloneqq t_k$ has the number $i = 1$. By the properties of the spectral norm (Definition 4.1.1 and Theorem 4.1.7),

$$|x(t_k+\delta)| \overset{(37)}{=} \left| \mathrm{e}^{A_{\text{cont}}(t_k+\delta-\tau_N)} \left(\widetilde{\prod_{i=1}^{N}} E_i \mathrm{e}^{A_{\text{cont}}(\tau_i-\tau_{i-1})} \right) x(t_k) \right| \tag{61}$$

$$\overset{(31),(34)}{\leq} \mathrm{e}^{\|A_{\text{cont}}\|_2 (t_k+\delta-\tau_N)} \left(\widetilde{\prod_{i=1}^{N}} \|E_i\|_2 \, \mathrm{e}^{\|A_{\text{cont}}\|_2 (\tau_i-\tau_{i-1})} \right) |x(t_k)| \tag{62}$$

$$\overset{(60)}{\leq} \mathrm{e}^{\|A_{\text{cont}}\|_2 (t_k+\delta-\overset{t_k}{\tau_0})} (\underbrace{\alpha_{\text{ev}}}_{\geq 1})^{\overset{0\leq N\leq m+p}{\tilde{N}}} |x(t_k)| \tag{63}$$

$$\leq \underbrace{\mathrm{e}^{\|A_{\text{cont}}\|_2 \, \delta}}_{\mathrm{e}^{\lambda\delta}} \underbrace{\alpha_{\text{ev}}^{m+p}}_{\bar{\alpha}} |x(t_k)|. \tag{64}$$

□

Proof of Theorem 4.1.13 (CGES ⇔ DGES). The proof is similar to Al Khatib et al., 2016, Prop. 2 and a special case of the results of Nešić et al., 1999.

"⇒": Assume CGES(λ, α) and let $\rho = \mathrm{e}^{\lambda T}$. Then, the system is DGES(ρ, α):

$$|x_k| = |x(t_k)| \overset{\text{CGES, (46)}}{\leq} \alpha |x(t_0)| \mathrm{e}^{\lambda k T} = \alpha |x_0| \rho^k. \tag{65}$$

"⇐": Assume DGES(ρ, α), which implies $0 < \rho < 1$. Let $\lambda = \log(\rho)/T$, so $\lambda < 0$ and $\rho = \mathrm{e}^{\lambda T}$. Assume $t \geq t_0$, since both CGES and DGES only refer to this time range.

Denote rounding down by $\lfloor x \rfloor$. Define $k(t) \coloneqq \left\lfloor (t - t_0)/T \right\rfloor$ as the integer k for which $t_{k(t)} \leq t < t_{k(t)+1}$. This implies $k(t) \leq (t - t_0)/T$ and therefore $\rho^{k(t)} = \mathrm{e}^{\lambda T k(t)} \leq \mathrm{e}^{\lambda (t - t_0)}$. Because Lemma 4.1.14 bounds the ratio between $|x(t)|$ and the previous discrete-time sample $|x(t_{k(t)})|$, the system is CGES(λ, α_c):

$$|x(t)| \overset{(59)}{\leq} |x(t_{k(t)})| \bar{\alpha} \mathrm{e}^{\bar{\lambda} T} \overset{\text{DGES}}{\leq} \alpha |x(t_0)| \underbrace{\rho^{k(t)}}_{\leq \mathrm{e}^{\lambda (t - t_0)}} \bar{\alpha} \mathrm{e}^{\bar{\lambda} T} \tag{66}$$

$$\leq \underbrace{\alpha \bar{\alpha} \mathrm{e}^{\bar{\lambda} T}}_{=: \alpha_c} \mathrm{e}^{\lambda (t - t_0)} |x(t_0)|. \tag{67}$$

□

4.1.4.4 Adding Disturbance

The previous derivations assumed zero disturbance and measurement noise ($n_{\text{dist}} = 0$). While this assumption does not matter for the question of exponential stability, the impact of noise is important for the analysis of state bounds. Under simplifying assumptions stated in the following, the discretized system becomes

$$x_{k+1} = A(\sigma_k) x_k + G(\sigma_k) d_k, \quad d_k \in \mathcal{D}. \tag{68}$$

In the general case, the disturbance matrix $G(\sigma_k)$ can be determined by extending the discretization of $A(\sigma_k)$ to uncertain inputs, which will be outlined later (Section 4.1.4.5).

For a simplified approximation, three assumptions are used that lead to

$$G(\sigma_k) = \begin{bmatrix} \int_0^T \mathrm{e}^{A_\mathrm{p} \tau} G_\mathrm{p} \, \mathrm{d}\tau \; (*) \\ 0 \; (**) \\ 0 \; (***) \\ 0 \end{bmatrix}: \tag{69}$$

1. There is no measurement noise ($H_\mathrm{p} = 0$).
2. The disturbance $d(t)$ is constant within every period, i.e.,

$$d(t) = d_k \quad \forall t \in [t_k,\, t_{k+1}) \quad \forall k. \tag{70}$$

3. The "inter-sampling" impact (**) and (***) of the current disturbance $G_\mathrm{p} d_k$ on the y_d- and x_d-components of x_{k+1} is neglected. This will be detailed later.

Under these assumptions, G is independent of σ_k. To explain the third assumption, consider the effect of the disturbance d_k during one period from t_k until t_{k+1}. The effect accumulates continuously in x_p: For $t_k \le t < t_{k+1}$ and, to simplify the discussion, $u = 0$, the explicit solution of

$$\dot{x}_\mathrm{p}\big|_{u=0} = A_\mathrm{p} x_\mathrm{p} + G_\mathrm{p} d \tag{71}$$

is

$$x_\mathrm{p}(t)\big|_{u=0} = \mathrm{e}^{A_\mathrm{p}(t-t_k)} x_\mathrm{p}(t_k) + \int_0^{t-t_k} \mathrm{e}^{A_\mathrm{p}\tau} G_\mathrm{p}\, \mathrm{d}\tau\, d_k. \tag{72}$$

For $t = t_{k+1}$, this leads to the direct effect of d_k on $x_\mathrm{p}(t_{k+1})$, corresponding to (*) in (69). Besides this effect, there are neglected inter-sampling effects of d_k on x_d (**) and y_d (***): For ideal timing, y_d is sampled in the middle of the period using $x_\mathrm{p}(kT)$, which includes the effect (***) of d_k in the first half of the period ($t_k < t < kT$). At the end of the period ($t = t_{k+1}$), the controller state x_d is updated using the value stored in y_d, therefore including the dependence (**) on d_k.

4.1.4.5 Outlook on a Generalization

Lifting the previous simplifying assumptions is tedious but possible by a "pessimistic discretization" of $d(t)$, as sketched in the following: Using set-valued reachability analysis for the behavior within one period, the change Δx of the state x caused by the disturbance $G d(t)$ within one period, i.e.,

$$\Delta x \coloneqq x(t_{k+1}) - x(t_{k+1})\big|_{d(t)=0 \text{ for } t_k \le t \le t_{k+1}} \tag{73}$$

can be bounded by a set

$$\Delta x \in \{\tilde{G}\tilde{d}_k \mid \tilde{d}_k \in \tilde{\mathcal{D}}\}. \tag{74}$$

Herein, $\tilde{G}$ is a shape matrix, possibly the identity matrix, and $\tilde{\mathcal{D}}$ a set. Then, the pair $(\tilde{G}, \tilde{\mathcal{D}})$ is a pessimistic replacement for $(G(\sigma_k), \mathcal{D})$. The details are left for future work.

4.1.4.6 Summary of the Discrete-Time System Model

Using the LIS framework, a discrete-time model of the real-time control system was derived that is parameterized by the timing in the current control period. The benefit is that the problem structure is encoded in the sparsity pattern of the transition matrices, which will be used later to simplify the stability analysis. The downside is that the approach is limited to the linear case and that the discretization of continuous disturbance is difficult.

The resulting model can be summarized as follows: Using the results and approximations of Section 4.1.4.4 and the previous sections, the control loop with disturbance d_k and uncertain execution σ_k per (16) is discretized as

$$x_{k+1} = A(\sigma_k)x_k + Gd_k, \quad d_k \in \mathcal{D}, \tag{75a}$$

$$s_k = C_\mathrm{s} x_\mathrm{p}(t_k) = \underbrace{\begin{bmatrix} C_\mathrm{s} & 0 \end{bmatrix}}_{=:\bar{C}_\mathrm{s}} x_k \tag{75b}$$

with G per (69) and A per (50) in the general case or per (56) if the controller is not skipped.

This model forms the basis of many subsequent derivations. It inherits the assumptions of the previous sections, which can be summarized as:

- linear dynamics,
- the approximation of initial state and disturbance (Section 4.1.4), and
- no measurement noise (Section 4.1.4.4).

4.2 Hybrid Automata

The expressiveness of LIS is limited: First, they are linear, though generalizations to nonlinear dynamics exist. Second, the event matrices and times are considered as known a priori. Therefore, it is fundamentally impossible to specify timing bounds, skip rates or more complicated policies inside the formalism of LIS. This complicates modeling and analysis because a mixture of LIS and other formalisms must be considered.

A solution to this problem are hybrid automata (HA), which will be introduced in this section: They combine a discrete-event automaton with classical continuous-time, continuous-valued dynamical behavior. HA are therefore a powerful formal model for the mixture of continuous-time and discrete-time dynamics of real-time control systems.

Parts of this section were published previously: The formalism and definitions are an extended version of Gaukler, 2020a, and the system model and experiments were presented in Gaukler and Ulbrich, 2019.

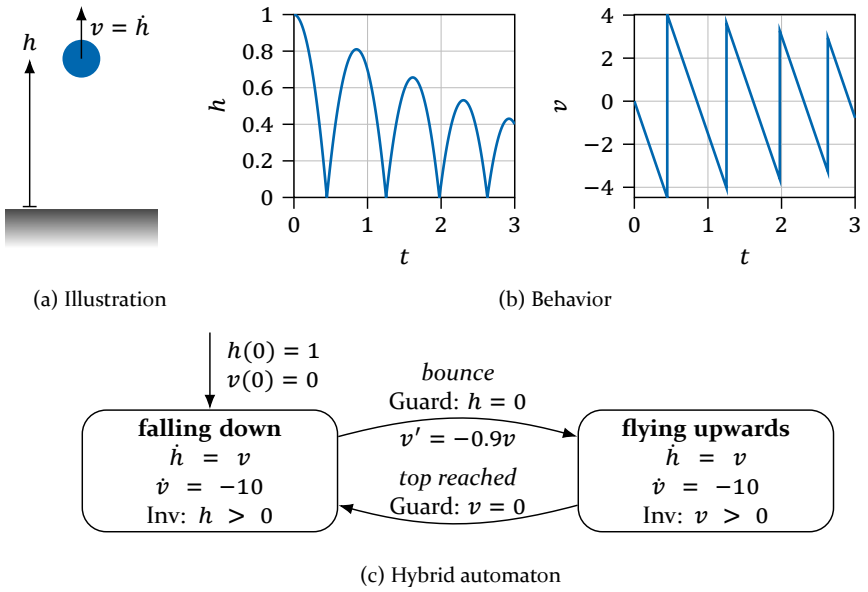
4.2.1 Introduction

As an introduction to hybrid automata, Figure 15 shows the simple model of a bouncing ball (Althoff, 2015). The formal details of the notation will be gradually introduced in the following and can be ignored for now. The ball has height h above ground and upward velocity $v = \dot{h}$. Starting with $h = 1$ and $v = 0$, it falls down and accelerates by $\dot{v} = -10$ as long as it does not touch the floor ($h > 0$). Here, $h > 0$ is an *invariant condition* of the *location* (discrete-valued state) "falling down". Then, the ball bounces at $h = 0$, modeled by an instantaneous change of velocity: Formally, $h = 0$ is the *guard condition* of the transition "bounce", with the *transition function* $v' = -0.9v$. Note that v' denotes the value after the transition and should not be confused with the derivative $\dot{v}$. After this transition, the ball is moving upwards, which for illustration is modeled as a separate location. The flight upwards continues while $v > 0$ (invariant condition), until at $v = 0$ (guard condition) the location changes from "flying upwards" to "falling down". This transition does not affect v and h, as no transition function is given.

In this example, all dynamics and transitions are linear if, for the sake of explanation, the constant term -10 is considered as a state variable g with dynamics $g(0) = -10,\ \dot{g} = 0$. Still, this system can not be modeled by a LIS, as the times of the *bounce* transitions depend on the initial state.

Next, the formalism of hybrid automata according to Henzinger, 2000 is introduced, starting with an informal summary analogous to Frehse, 2015.

Informal Summary The elements of hybrid automata are named in Figure 16. In each location (discrete-valued state of the discrete-event automaton), the state variables of the continuous system evolve according to an ordinary differential equation (ODE). A transition to another location *may* be taken while the guard condition of the transition is true (*may-semantics*). If the invariant condition of the current location is violated, a transition *must* be



(a) Illustration (b) Behavior

(c) Hybrid automaton

Figure 15: Bouncing ball as an example of a hybrid automaton

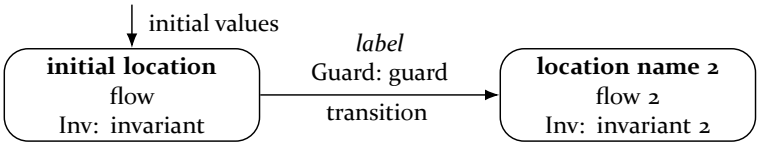


Figure 16: Legend for hybrid automata

taken immediately. At the transition, the continuous state changes instantaneously as given by the jump transition, e.g., $x' = 2x$ denotes that x jumps to twice its previous value. The labels of transitions are used for synchronization in the composition of multiple automata, which will be discussed later.

4.2.2 Definition

The previous informal definition is not strict enough for formal proofs. Hence, a formal definition is introduced that defines the behavior of the hybrid automaton by a chain of discrete and continuous transitions:

Definition 4.2.1 (Hybrid Automaton)**.** The semantics of hybrid automata in this doctoral thesis are based on the work of Henzinger (Henzinger, 2000, Definitions 1.4 and 1.5) with some minor simplifications.

A hybrid automaton is specified by the following information:

- A finite set $\mathcal{Loc}$ of discrete-valued *locations* ("modes", "discrete-valued states")
- A vector $x \in \mathbb{R}^n$ of continuous-valued *state variables* (abbreviated as "states" with some abuse of terminology)
- Discrete transitions ("jumps") from $i \in \mathcal{Loc}$ to $j \in \mathcal{Loc}$ with
  - a non-numeric *transition label* $\alpha_{i,j} \notin \mathbb{R}$, e.g., a letter of the alphabet, whose value is only relevant for the composition operation that is discussed later,
  - a *guard condition* $guard_{i,j}(x) \in \{true, false\}$ to describe if the transition is enabled, and
  - a *transition function* $trans_{i,j}(x) \subseteq \mathbb{R}^n$ to describe the set of possible new values x' of the continuous-valued state x
- Continuous dynamics in each location $i \in \mathcal{Loc}$ with
  - an *invariant condition* $inv_i(x) \in \{true, false\}$ and
  - a *flow* (dynamics) $flow_i(x, \dot{x}) \in \{true, false\}$, e.g., an ordinary differential equation or a differential inclusion.
- An *initial set* $\mathcal{Init} \subseteq \mathcal{Loc} \times \mathbb{R}^n$ of location-state pairs.

The *discrete transition* from $x = x$ in location i to $x = x'$ in location j is possible (formally, the relation $(i, x) \xrightarrow{\alpha_{i,j}} (j, x')$ is true) if and only if $guard_{i,j}(x)$ is true and $x' \in trans_{i,j}(x)$.

The *continuous transition* in location i from x to x' with duration $\delta \geq 0$ is possible (formally, the relation $(i, x) \xrightarrow{\delta} (i, x')$ is true) if and only if there is a solution (witness) $\xi(t) : [0; \delta] \mapsto \mathbb{R}^n$ such that

- $\xi(0) = x$ and $\xi(\delta) = x'$,
- $flow_i(\xi(t), \dot{\xi}(t))$ and $inv_i(\xi(t))$ are true on $(0; \delta)$ and
- $\xi(t)$ is differentiable on its domain $[0; \delta]$. Informally, $\xi(t)$ can be drawn as a smooth curve from $\xi(0)$ to $\xi(\delta)$. (For a precise definition of differentiability on a closed set, refer to Bernstein, 2018, Definition 12.5.3.)

In particular, the invariant may be violated at the start and end of the transition, and $(i, x) \xrightarrow{0} (i, x)$ is always true. Note that here, t refers to the local time since the start of the transition.

A *trajectory* of the automaton is defined as a chain of transitions

$$(l_0, x_0) \xrightarrow{\beta_1} (l_1, x_1) \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_n} (l_n, x_n) \tag{76}$$

of arbitrary length $n \geq 0$ with $(l_0, x_0) \in \mathcal{I}nit$ and $inv_{l_0}(x_0) = true$. The restriction to $inv_{l_0}(x_0) = true$ ensures compatibility with the definition of Henzinger but is not relevant in this doctoral thesis. Note that β_i refers to the label α of a discrete or the duration δ of a continuous transition.

The *duration* of the trajectory is defined as the sum

$$\sum_{i=1}^{n} \begin{cases} \beta_i, & \beta_i \in \mathbb{R} \text{ (continuous transition)}, \\ 0, & \text{else (discrete transition)} \end{cases} \tag{77}$$

of the durations of all continuous transitions in the trajectory.

Should any of the involved functions become undefined, e.g., if *flow* contains a division by zero, then this is treated as *false* or the empty set {}, forcing an end of the trajectory (deadlock) if no other transition is possible. ◁

Definition 4.2.2 (Graphical Notation for HA)**.** HA are denoted graphically per Figure 16 on page 73. Disabled transitions (guard=*false*), as well as irrelevant labels and location names may be omitted. State variables not mentioned in a transition function remain unchanged, e.g., an empty transition function denotes the identity transition $x' = x$. ◁

An exemplary trajectory of the bouncing-ball example (Figure 15) is

$$\left(\mathrm{d}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) \xrightarrow{0.44} \left(\mathrm{d}, \begin{bmatrix} 0 \\ -4.5 \end{bmatrix}\right) \xrightarrow{\text{bounce}} \left(\mathrm{u}, \begin{bmatrix} 0 \\ 4.0 \end{bmatrix}\right) \xrightarrow{0.41} \left(\mathrm{u}, \begin{bmatrix} 0.81 \\ 0 \end{bmatrix}\right) \xrightarrow{\text{top reached}} \left(\mathrm{d}, \begin{bmatrix} 0.81 \\ 0 \end{bmatrix}\right).$$

Herein, the numbers are rounded and the location names are abbreviated as “d” (falling down) and “u” (flying up). The state evolution is piecewise differentiable, where a continuous transition represents a differentiable piece. Jumps are only possible by discrete transitions.

The following exemplary trajectories illustrate that continuous transitions do not need to be of maximal duration:

$$\begin{aligned} &\left(\mathrm{d}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right), \\ &\left(\mathrm{d}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) \xrightarrow{0} \left(\mathrm{d}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) \xrightarrow{0} \left(\mathrm{d}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right), \\ &\left(\mathrm{d}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) \xrightarrow{0.1} \left(\mathrm{d}, \begin{bmatrix} 0.95 \\ -1.0 \end{bmatrix}\right) \xrightarrow{0.01} \left(\mathrm{d}, \begin{bmatrix} 0.94 \\ -1.1 \end{bmatrix}\right). \end{aligned}$$

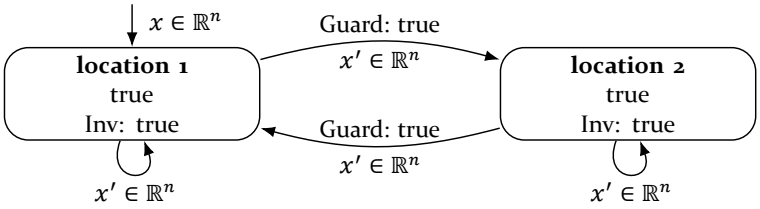


Figure 17: Automaton with two locations that permits all behavior of $x(t)$. Changing this automaton will restrict the possible behavior.

4.2.3 Dynamic Phenomena

To understand hybrid automata, it is helpful to have the right mental model. Many phenomena seem counterintuitive if one considers HA as dynamical systems that *generate* an output trajectory. A more fitting mental model is that an automaton *restricts* the set of valid trajectories, analogous to how traffic rules restrict the set of legal driving trajectories. Loosely speaking, an "empty" HA as in Figure 17 permits all trajectories, analogous to a country without any traffic rules. Adding details to the HA, such as dynamics or invariants instead of the default value *true*, constrains the set of trajectories. This correctly explains several important effects:

Nondeterminism The bouncing-ball example in Figure 15 is deterministic: The automaton has a unique trajectory, up to formal changes such as ending earlier or splitting continuous transitions. However, hybrid automata also support nondeterministic behavior to express uncertainty. Then, there is a set of possible behaviors, just like traffic rules allow for multiple legal driving routes. In continuous transitions, such uncertainty can occur both for the initial values and the flow. For the bouncing ball, this could be $h(0) \in [1; 2]$ for unknown initial height and $\dot{v} \in [-9; -10]$ for uncertain friction.

Uncertain discrete transitions are less intuitive but very powerful: Recall that discrete transitions *may* be taken as soon as a guard condition is true and *must* be taken immediately if the invariant is false. Therefore, the transition may or may not be taken if guard and invariant are both true. For the bouncing ball, replacing the guard $h = 0$ by $h < 0.1$ represents an uneven surface with a height between 0 and 0.1: The ball *may* bounce as soon as $h < 0.1$ and *must* bounce at latest at $h = 0$, since then the invariant $h > 0$ has just become false.

These possibilities for nondeterminism are a significant distinction to practical implementations of hybrid systems: For example, the Stateflow modeling framework contained in MATLAB/Simulink uses *must-semantics*, i.e., a possible transition must be taken immediately (Bak, Beg, et al., 2017).

Deadlock The definition of transitions has the counterintuitive result that trajectories may get stuck in a "deadlock" from which neither continuous evolution nor a discrete transition is possible. For a physical system, this can not occur, as physical time never stops. A real-world analogy is a car that has entered a dead-end street in which turning and reverse driving are not permitted, so there is no legal way to get out. As an academic example, consider the bouncing ball in Figure 15 and change the invariant $h > 0$ to $h > -1$. Now, it is possible that the ball does not bounce but instead keeps falling down below the floor until the invariant becomes false at $h = -1$. Then, there is no possible transition, as the guard condition $h = 0$ is not satisfied, so the trajectory ends. To be precise, the trajectory can still be continued by continuous transitions of zero duration but this does not advance time.

Zeno Behavior As the trajectory of the bouncing ball converges towards $h = 0$, the rate of discrete transitions per time grows without bound. This phenomenon is named Zeno behavior (Frehse, 2015). In some automata, it can occur that time does not progress beyond a certain point, although there is no deadlock: Change the bouncing transition to $v' = 0$. Then, the automaton will reach $h = v = 0$, where all guard conditions are constantly true and all invariants are constantly false, so it must take discrete transitions in an infinite loop without any continuous evolution. A possible interpretation of this paradoxical result is that the model is wrong and should be fixed by adding a third location "resting on the ground" with dynamics $\dot{v} = \dot{h} = 0$ that is activated if $h = v = 0$.

4.2.4 Composition and Interconnection

The basic operation for combining two automata H_1 and H_2 is the composition or parallel product: The trajectories of the composition $H_1 \parallel H_2$ are trajectories consistent with both H_1 and H_2. Consistency loosely means that if a state variable or transition label is present in the respective sub-automaton, then it must obey the dynamics of this sub-automaton along the trajectory. If there

are no such shared variables or transition labels, $H_1 \parallel H_2$ merely joins two independent dynamics into one state vector. Else, $H_1 \parallel H_2$ is the intersection of behaviors, e.g., a discrete transition is taken if the respective guard conditions are true in both H_1 and H_2.

In the analogy of traffic rules, each automaton represents one traffic rule, and the composition is the result of obeying both traffic rules: The possible driving trajectories are the intersection of those of each rule. If the rules apply to independent variables, such as velocity and permitted lanes, this intersection boils down to joining the possible ranges of each variable, so that, e.g., a given velocity range is permitted on all given lanes.

A detailed formalism for the composition and interconnection of HA, termed hybrid I/O automata, is given by Lynch et al., 1996. To avoid excessive formalism, this doctoral thesis instead uses a simplified definition.

Definition 4.2.3 (Composition of Hybrid Automata)**.** The trajectories of the composition $H_1 \parallel H_2$ of two hybrid automata H_1, H_2 are defined as follows. Herein, the superscript $\blacksquare^{H_i}$ annotates the respective automaton.

If $(l_1^{H_1}, x_1^{H_1}) \overset{\beta^{H_1}}{\longrightarrow} (l_2^{H_1}, x_2^{H_1})$ is a transition in H_1 and $(l_1^{H_2}, x_1^{H_2}) \overset{\beta^{H_2}}{\longrightarrow} (l_2^{H_2}, x_2^{H_2})$ a transition in H_2, then $(l_1, x_1) \overset{\beta}{\longrightarrow} (l_2, x_2)$ is a transition of $H_1 \parallel H_2$ if and only if the following conditions are all true:

1. The location names are concatenated: $l_i = (l_i^{H_1}, l_i^{H_2})$ for $i = 1, 2$.
2. The states are consistent: $x_i^{H_j} = p_j(x_i)$ for $i = 1, 2$ and $j = 1, 2$, with a projection function p_j that selects the state variables occurring in H_j.
3. The labels are consistent:
   - β is a label of H_1 or H_2 or both. (Formally, β is a label of H_i if and only if $\beta \in \mathbb{R}_{\geq 0}$ or there is an $\alpha_{j,k}^{H_i} = \beta$.)
   - If β is a label of H_1, then $\beta^{H_1} = \beta$, else $\beta^{H_1} = 0$. The same holds for H_2.
4. The states are also consistent during continuous evolution. Formally, if $\beta \in \mathbb{R}_{\geq 0}$, the witnesses $\xi^{H_1}(t)$ and $\xi^{H_2}(t)$ of the transition can be combined into an overall witness $\xi(t)$ such that $\xi^{H_j}(t) = p_j(\xi(t))$ for all $t \in [0; \beta]$ and $j = 1, 2$.

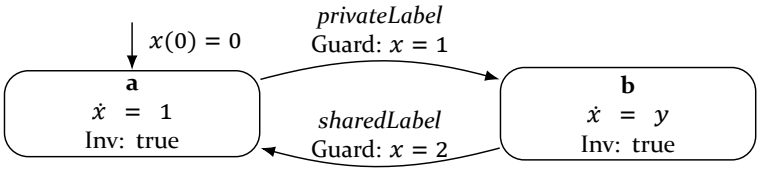
This condition is required to exclude the following case: For simplicity, let H_1 and H_2 have the same state variables. If H_1 and H_2 evolve continuously from x_0 to x_1 in the same time δ but on a different continuous-time "trajectory", e.g., with $\xi^{H_1}(\delta/2) \neq \xi^{H_2}(\delta/2)$, both still have the same transition $(l_0, x_0) \xrightarrow{\delta} (l_1, x_1)$. The inconsistency is only revealed by checking the witnesses.

Furthermore, the start (l_0, x_0) of the trajectory must be consistent with initial sets of H_1 and H_2 analogous to conditions 1 and 2. ◁

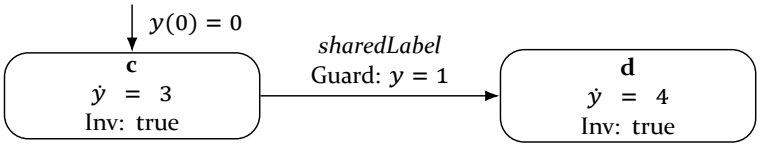
An example is illustrated in Figure 18. Note that while H_1 has a transition from a to b and H_2 from c to d, there is no joint transition from (a, c) to (b, d) as the automata "disagree" on the transition label.

Interconnection For efficient modeling, it is useful to model a system as a network of connected subsystems, where each subsystem is an HA. In the above definition of composition, the connection is implicit by the names of variables and transition labels. This is problematic if the same automaton is used as a template for multiple components, e.g., to model multiple sample-and-hold components. Hence, this work uses a graphical notation by means of block diagrams as in Figure 18d, inspired by the SpaceEx graphical model editor (Frehse, Le Guernic, et al., 2011). Blocks represent hybrid automata, and connections denote shared variables or labels. All other variables and labels are considered internal. Explained in formal terms, the composition of all network components is applied after renaming the variables and labels such that two of them have the same name if and only if they are connected.

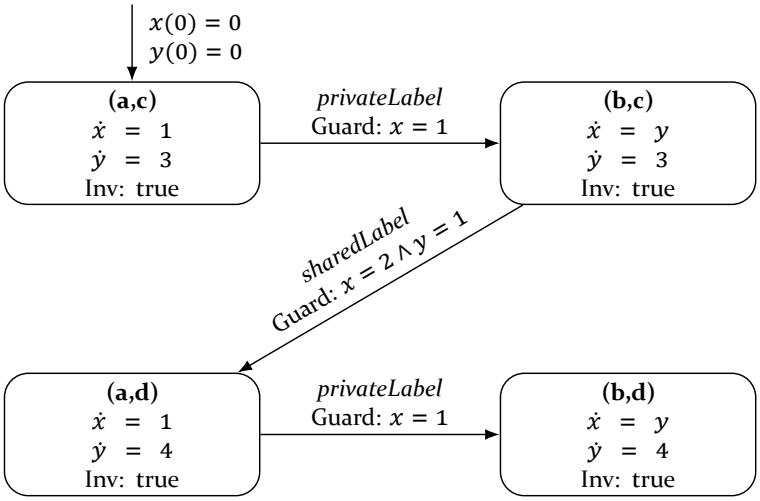
Signal Flow In classical block diagrams, as used in graphical modeling techniques such as MATLAB/Simulink simulations, connections have a clear direction of signal flow. This is not necessarily the case when interconnecting hybrid automata: For example, a transition with a shared label can only occur if both automata "allow" it. This mutual synchronization corresponds to bidirectional signal flow. In Figure 18, H_1 prevents the transition from (a, c) to (a, d), while H_2 prevents the transition from (b, d) to (a, d). For shared state variables, there is typically one automaton "writing" to the variable by defining a differential equation, while other automata only "read" it in their dynamics, so then the signal flow is clear. The case of shared writing is theoretically possible but does not occur in this work.



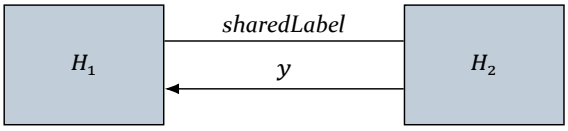
(a) H_1



(b) H_2



(c) $H_1 \parallel H_2$



(d) Block diagram of $H_1 \parallel H_2$

Figure 18: Academic example of the composition of hybrid automata

4.2.5 Reachable Set

Safety analysis often considers the set of states that an automaton can reach:

Definition 4.2.4 (Reachable Set)**.** The reachable set of an automaton H is the set of all location-state-pairs (l, x) to which there is a trajectory:

$$\text{Reach}(H) \coloneqq \{(l, x) \mid \text{there is a trajectory of } H \text{ that ends with } (l, x)\} \quad (78)$$

◁

To compare two automata, one can define equivalence based on the trajectories:

Definition 4.2.5 (Equivalence)**.** Two hybrid automata H_1, H_2 are equivalent if and only if every trajectory $(l_0, x_0) \xrightarrow{\beta_1} \ldots \xrightarrow{\beta_n} (l_n, x_n)$ of H_1 is also a trajectory of H_2 and vice versa. ◁

These definitions are too strict for comparing the physical behavior of a feedback control system: After transforming the state vector or renaming locations, the automaton has the same physical behavior, but according to the above definition it is no longer equivalent and has a different reachable set. This is remedied by the following specialized definitions:

Definition 4.2.6 (Reachable Set of a Continuous Variable)**.** Let H be a hybrid automaton with $x \in \mathbb{R}^n$ as the vector of continuous state variables. Then, "$x = \xi$ is reachable by H at t", abbreviated as $\xi \in \text{Reach}_x(H, t)$, if and only if there is a trajectory of H that has a duration of t and ends with (l, ξ), where the location l is irrelevant. ◁

For comparison with other literature, in the notation of Henzinger, 2000, Def. 1.4 and 1.5, this definition reads

$$\text{Reach}_x(H, t) \coloneqq \Big\{ x \mid \exists Tr, j : \quad Tr = \langle \alpha_i, (l_i, x_i) \rangle_{i \geq 1} \text{ is an initialized trajectory of } H, \quad x_j = x, \quad \sum_{i=1}^{j} \mathit{duration}(\alpha_i) = t \Big\} \quad (79)$$

with "initialized trajectory" and "duration" as defined there.

Definition 4.2.7 (Generalized Notation for the Reachable Set)**.** To extend the previous notation, denote the reachable set of a function $z = h(x)$ of the state vector $x \in \mathbb{R}^n$ as

$$R_z(H,t) \coloneqq \{h(\xi) \mid \xi \in \mathrm{Reach}_x(H,t) \cap \mathrm{dom}\, h\}. \tag{80}$$

Herein, $\mathrm{dom}\, h \subseteq \mathbb{R}^n$ denotes the domain of h, i.e., undefined results are excluded.

The reachable set over a time range $\mathcal{T}$ is defined as

$$\mathrm{Reach}_x(H,\mathcal{T}) \coloneqq \bigcup_{t\in\mathcal{T}} \mathrm{Reach}_x(H,t), \tag{81}$$

and the reachable set over all time as

$$\mathrm{Reach}_x(H) \coloneqq \mathrm{Reach}_x(H,\mathbb{R}_{\geq 0}). \tag{82}$$

◁

Definition 4.2.8 (Continuous Equivalence)**.** Two hybrid automata H_1, H_2 are x-equivalent ($H_1 =_x H_2$) if and only if the corresponding reachable sets for $x(t)$ are the same for all times t:

$$H_1 =_x H_2 \quad :\Leftrightarrow \quad \mathrm{Reach}_x(H_1,t) = \mathrm{Reach}_x(H_2,t)\ \forall t. \tag{83}$$

Matching (80), this notation includes the case that H_1 and H_2 have different state vectors x and $\tilde{x}$ with a given, possibly non-invertible transformation $x = q(\tilde{x})$. ◁

Loosely speaking, x-equivalence considers all discrete transition labels and locations as hidden internal behavior and only compares for equality with respect to a given "output" variable $x(t)$.

4.2.6 Numerical Simulation and Reachability Analysis

In principle, the trajectories of hybrid automata can be determined using numerical simulation. However, nondeterminism is difficult to handle since it typically leads to infinitely many alternative trajectories (Frehse, 2015). Randomized simulations are possible but can generally not cover the whole reachable set. For safety analysis, this under-approximation is not acceptable. As discussed in Section 2.2, *reachability analysis* solves this issue, as it computes a guaranteed *outer approximation of the reachable set*.

Set-Valued Reachability Analysis In the last decades, numerous software programs for reachability analysis of hybrid systems have been developed (Althoff, Bak, et al., 2020; Geretti et al., 2020). As a detailed discussion of their inner workings would be beyond the scope of this work, this section instead summarizes a single popular approach, namely iterative *set-valued* computation. The basic algorithms will be sketched only as far as it is required to understand the fundamental problems, summarizing the tutorial by Frehse, 2015 and the works of Althoff, 2010; Althoff and Kochdumper, 2018.

The following computations operate on sets, which should ideally be computed exactly. However, for an implementation with finite computing time and memory, one has to resort to outer approximations, here denoted by the operator "Approx". Let S be any set. Ideally, $\text{Approx}\, S = S$, while in reality $\text{Approx}\, S \supseteq S$. Practical examples are bounding to multidimensional intervals ("box"), ellipsoids or convex polytopes.

Continuous Reachability A first subproblem is to determine the reachable set for a purely continuous system, i.e., for a hybrid automaton with a single location and no discrete transitions: For a given time horizon t_H, determine

$$\text{Reach}_x\left(\left(\begin{array}{c} x(0) = \mathcal{X}_0 \\ \hline \dot{x} \in \mathcal{F}(x) \\ \text{Inv: } x \in \mathcal{S} \end{array}\right), [0; t_H]\right). \tag{84}$$

This is possible by set-valued numerical integration of the ODE. For illustration, a simplified approach using set iterations is explained in the following:

Denote by $\mathcal{X}_i$ the over-approximated set of states x reachable at $t = i\epsilon$, where $\epsilon > 0$ is a short time step. To simplify the discussion, assume that the set $\mathcal{X}_0$ of initial states is a convex polytope (e.g., multidimensional interval).

Repeat the following for every time step $i = 0, 1, \ldots, t_H/\epsilon$:

1. For every vertex (corner) of the current set $\mathcal{X}_i$, use standard numerical integration to compute the successor state after one time step ϵ. For nondeterministic dynamics, neglect the nondeterminism in this step and choose an "average" trajectory.
2. Compute $\mathcal{X}_{i+1}$ as the convex hull of all successors. In special cases, this yields the exact set of successor states $\text{Reach}_x(\ldots, (i+1)\epsilon)$. In the general case, enlarge the set appropriately to guarantee an outer approximation despite nondeterminism, numerical error and further effects.

3. Then, remove states that violate the invariant, except for states at the edge of the invariant.
4. If the set has become too complex, e.g., if the number of vertices of a polytope is too large for an efficient computation, apply a simplifying outer approximation.

Finally, compute the union of all intermediate sets $\mathcal{X}_i$ and enlarge it to account for the discretization error, i.e., the gap between sample times $t = i\epsilon$.

Practical implementations are more sophisticated to obtain reasonable accuracy and computational speed. Details can be found in the literature mentioned earlier. Nevertheless, the sketched approach explains the working principle and difficulties of many existing software tools.

Hybrid Reachability Building upon the previous algorithm, the reachable set $\mathrm{Reach}(H)$ of a hybrid automaton can be computed iteratively. Denote by $\mathcal{X}_k^l$ the set of reachable x in location l at the k-th iteration of the algorithm.

1. For every location l, let $\mathcal{X}_0^l$ be the set of x such that (l, x) is an initial state of the automaton. Set $k = 0$.
2. For every location l, compute the set of states reachable by only continuous evolution for some time horizon $\delta_h > 0$, ideally $\delta_h \to \infty$:

$$\mathrm{Cont}_l(\mathcal{X}_k^l) \coloneqq \mathrm{Approx}\,\{x' \mid x \in \mathcal{X}_k^l,\ (l,x) \xrightarrow{\delta} (l,x') \text{ is a continuous transition of } H,\ 0 \le \delta \le \delta_h \quad \}. \tag{85}$$

   This is the same as in the purely continuous case discussed before.

3. For every combination of locations j and l, compute the set of states resulting from a discrete transition from j to l:

$$\mathrm{Disc}_{j\to l}(\mathcal{X}_k^j) \coloneqq \mathrm{Approx}\,\{x' \mid x \in \mathcal{X}_k^j,\ (j,x) \xrightarrow{\alpha} (l,x') \text{ is a discrete transition of } H \quad \}. \tag{86}$$

   The set is empty if there is no such transition. Implementing the computation is a straightforward application of set-based arithmetic, e.g., interval arithmetic.

4. For every location l, compute the union of the previously reachable states and the newly reached states

$$\mathcal{X}_{k+1}^{l} = \text{Approx}\left(\mathcal{X}_{k}^{l} \cup \text{Cont}_{l}(\mathcal{X}_{k}^{l}) \cup \bigcup_{j \in \mathcal{L}oc} \text{Disc}_{j \to l}(\mathcal{X}_{k}^{j}) \right). \tag{87}$$

5. If nothing changed in this iteration, i.e., if $\mathcal{X}_{k+1}^{l} = \mathcal{X}_{k}^{l}$ for all l and the current k, then the algorithm has reached a fixed point, so $\mathcal{X}_{k}^{l} = \mathcal{X}_{k+1}^{l} = \cdots = \mathcal{X}_{\infty}^{l}$. The algorithm has computed an upper approximation of the (infinite-time) reachable set:

$$\text{Reach}(H) \subseteq \{(l, x) | l \in \mathcal{L}oc, x \in \mathcal{X}_{k}^{l}\}. \tag{88}$$

   Else, increment k and go back to step 2.

The result (88) is an outer approximation of the reachable set over an infinite time horizon. With minor modifications, it is also possible to only consider a finite time horizon.

Wrapping Effect A key problem of iterative algorithms using set-valued outer approximations is the *wrapping effect* (Kühn, 1998; Lohner, 2001): In every iteration, the approximation is applied. The resulting approximation error increases the set of states, which causes even more approximation error in the next iteration. Therefore, analysis will fail even for a stable system if the system is not "stable enough" to counter the "instability" caused by the approximation error. A classical example is a discrete-time system involving a rotation matrix (Lohner, 2001): The discrete-time system

$$x_{k+1} = \underbrace{0.9 \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}}_{A} x_k, \quad \phi = -\frac{\pi}{4}, \quad x_0 \in \underbrace{[-11; -9] \times [-1; 1]}_{\mathcal{X}_0} \tag{89}$$

is stable and therefore the reachable set $\mathcal{X}_k$ of states x_k converges to zero. However, as illustrated in Figure 19, an upper approximation $\bar{\mathcal{X}}_i$ diverges if it is computed using the interval bound

$$\bar{\mathcal{X}}_0 \coloneqq \mathcal{X}_0, \quad \bar{\mathcal{X}}_{i+1} \coloneqq \square(A\bar{\mathcal{X}}_i). \tag{90}$$

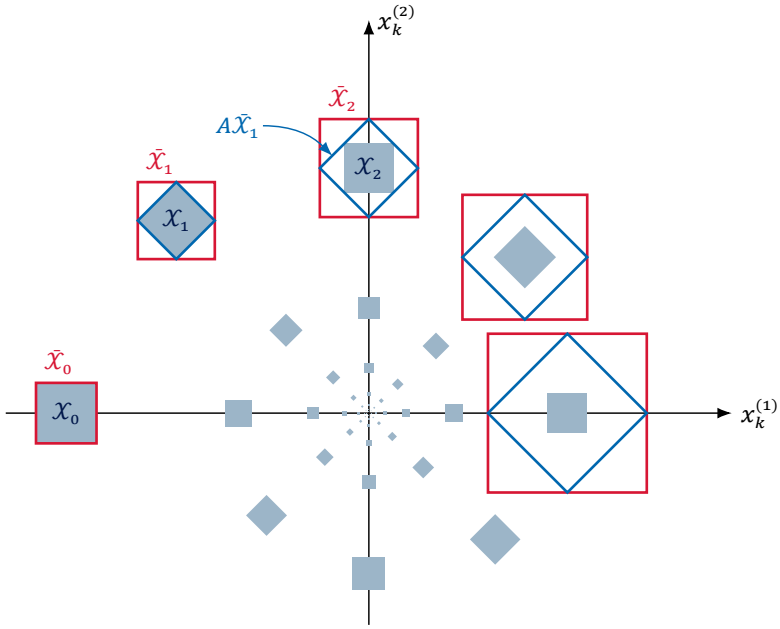


Figure 19: Graphical example of the wrapping effect (Equations (89) and (90)): The approximation error causes the computation of outer bounds $\bar{\mathcal{X}}_i$ (red) to diverge, although the actual reachable set $\mathcal{X}_i$ (shaded) converges.

Herein, the operator $\square$ denotes the enclosing multidimensional interval and $A\bar{\mathcal{X}}_i$ the linear transformation $\{Ax \mid x \in \bar{\mathcal{X}}_i\}$ of the set $\bar{\mathcal{X}}_i$. In this simple example, the wrapping effect can be avoided by using polytopes instead of intervals, so that $A\bar{\mathcal{X}}_i$ can still be represented exactly. However, in the general case there is no cure for the wrapping effect, only a number of often intricate tricks to reduce the symptoms (Girard et al., 2006; Lohner, 2001).

Wrapping Effect in Reachability Analysis A common problem in the practical application of reachability analysis is that the sequence of sets $\mathcal{X}_k^l, \mathcal{X}_{k+1}^l, \ldots$ diverges. From this result, it is impossible to tell whether the system is unstable or the divergence is only due to the wrapping effect.

In set-valued reachability analysis, there are two main sources of approximation error: First, the computation of continuous successors Cont_l may be subject to a wrapping effect, which can be avoided for linear time-invariant dynamics (Girard et al., 2006) but not in general (Althoff, 2010). Second, the approximation error occurs once per iteration of the reachability algorithm. The latter problem is most pronounced for trajectories with many discrete transitions, which is a significant hurdle for the analysis of real-time control

systems as considered in this thesis: As discussed later, sampling, actuation and controller computation are modeled as discrete events. Therefore, there are multiple discrete events within one control period, while the continuous evolution between them is too short for significant convergence of the state. As will be seen in the subsequent experiments, this unfortunate combination is an ideal breeding ground for large wrapping error. Countermeasures and workarounds will be discussed later in Section 5.2.

4.2.7 Abstraction

If direct safety analysis of a given system is computationally infeasible, an *abstraction*, i.e., a pessimistic model simplification, can be a viable alternative. By a very general definition, "abstraction" means that the representation of a problem is mapped into a new representation that is easier to handle but still preserves desirable properties (Giunchiglia and Walsh, 1992). For example, the nonlinear system

$$\dot{x} = -x + 0.1 \sin x \tag{91}$$

is abstracted by the following linear system with bounded disturbance:

$$\dot{x} = -x + 0.1d,\ |d| \leq 1 \tag{92}$$

Analysis of this simplified system is easier because it is linear. Because (91) $\Rightarrow$ (92), the "abstract" reachable set of (92) is a superset of the original reachable set of (91).

This idea is formalized by a relaxed variant of continuous equivalence, where "=" is replaced with "$\supseteq$":

Definition 4.2.9 (Continuous Abstraction)**.** Let H_1, H_2 be hybrid automata. H_2 x-abstracts H_1 ($H_2 \supseteq_x H_1$) if and only if all values of the continuous variable x reachable by H_1 are also reachable by H_2 at the same time t:

$$H_2 \supseteq_x H_1 \quad :\Leftrightarrow \quad \text{Reach}_x(H_2, t) \supseteq \text{Reach}_x(H_1, t)\ \forall t. \tag{93}$$

A mirrored version of this relation is defined as $H_1 \subseteq_x H_2 :\Leftrightarrow H_2 \supseteq_x H_1$. $\triangleleft$

Every automaton H_2 with $H_2 \supseteq_x H_1$ is a safe, i.e., pessimistic, approximation of H_1: If H_2 stays within a given safe x-region, then H_1 will certainly do, while the converse is not guaranteed.

Note that any reachability analysis that uses outer approximations can also be interpreted as the exact reachability analysis of an abstraction of the original automaton. Then, the approximations are considered to be a part of the dynamics of the abstract automaton.

Lemma 4.2.10 (Operator Properties of Continuous Equivalence and Abstraction)**.** *The operators $=_x$ and $\supseteq_x$ obey the same rules as $=$ and $\supseteq$. In particular,*

$$H_1 \supseteq_x H_2 \supseteq_x H_3 \quad \Rightarrow \quad H_1 \supseteq_x H_3, \tag{94}$$

$$H_1 =_x H_2 \quad \Leftrightarrow \quad H_1 \supseteq_x H_2 \wedge H_2 \supseteq_x H_1 \tag{95}$$

$$\textit{and} \quad H_1 =_x H_2 \quad \Leftrightarrow \quad H_2 =_x H_1. \tag{96}$$

Proof. This follows immediately from inserting the definitions of $=_x$ and $\supseteq_x$ and applying basic properties of $=$ and $\supseteq$. □

4.2.8 Idealized System Model

For ideal timing, the real-time control system is modeled by the simple HA of Figure 20. A timer variable $\tau = t \bmod T$ increases from 0 to T according to $\dot{\tau} = 1$ and then resets to $\tau = 0$. At this reset, which occurs at $t = kT, k \geq 1$, the controller state x_d is updated from the previous sample $y_\mathrm{d}(t^-) = y((k-1)T)$ and the new sample $y_\mathrm{d}(t) = y(kT)$ is stored. This one-step delay of y is required to match the fact that $x_{\mathrm{d},k+1}$ is computed from y_k.

For the assumed ideal timing, the output u is not needed as state variable because it can be directly computed from the controller state as $u = g_\mathrm{d}(x_\mathrm{d})$. The treatment of d will be discussed later in the full system model.

For simplicity, this model deviates from the timing of x_d that was originally defined in Section 3.1: Here, x_d is updated at $t = kT$, whereas originally it is defined to be updated at $t = (k + 1/2)T$. The trajectories of x_p, y_d and u remains unchanged.

4.2.9 Full System Model

If timing is considered, the resulting behavior is more complicated and therefore best modeled by the interconnection of multiple components: The controller, a global clock, the physical plant and multiple sample-and-hold (SH) elements. For clarity, the following first describes the model without skips.

A previous version of this section was published in Gaukler and Ulbrich, 2019.

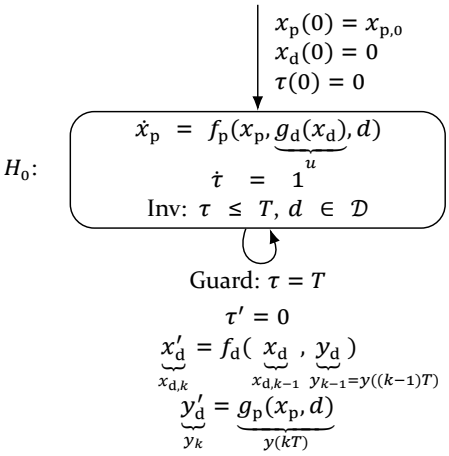


Figure 20: Hybrid Automaton representing the control loop for ideal timing

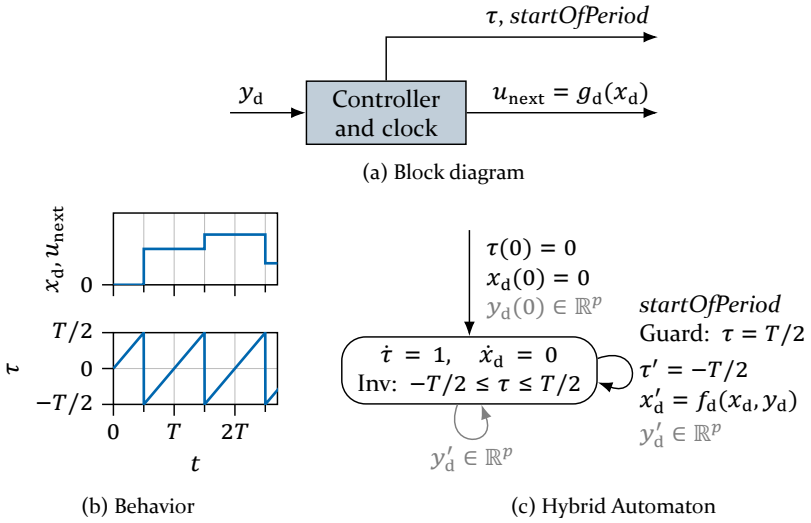
Controller and Clock A block diagram representation of controller and clock is shown in Figure 21a. The input is the sampled measurement y_d. As detailed in the following, the outputs are a clock signal τ, a label *startOfPeriod*, which corresponds to the start of the control period, and the "next" (most recently computed) control signal $u_{next} = g_d(x_d)$.

Figure 21b depicts that the periodic sawtooth clock signal

$$\tau = ((t + T/2) \bmod T) - T/2, \tag{97}$$

is the signed distance $t - kT$ to the nearest nominal sampling point kT. This signal is generated by the automaton in Figure 21c, which also includes the controller: τ continuously increases ($\dot{\tau} = 1$) until $\tau = T/2$ is reached. Then, τ is reset to $-T/2$ and the new controller state x_d is computed from the recent measurement y_d. This event is given a synchronization label *startOfPeriod*, which will later be used to force the reset of all sample-and-hold automata. It can be seen that this event occurs periodically and exactly at the timing barrier $t = (k + 1/2)T$ between two control periods.

The following paragraphs discuss subtleties required for formal correctness.



(a) Block diagram

(b) Behavior

(c) Hybrid Automaton

Figure 21: Specification of controller and clock

The gray-colored self-transition models that y_d is an external input. Unlike the extensive formalism of Lynch et al., 1996, the definition of HA in this work does not provide special support for this case. Hence, $y_d(t)$ is formally treated a state variable of the HA with arbitrary dynamics. At any time, a transition allows jumps of y_d, and in all modes the continuous dynamics do not specify a differential equation for y_d. After the composition, $y_d(t)$ will inherit its dynamics from the SH subsystem that provides the signal.

For the sake of readability, the representation slightly deviates from the definitions: First, u_{next} is not modeled inside the HA but only as a substitution in the block diagram. Second, although *startOfPeriod* is a label and therefore has bidirectional signal flow (cf. Section 4.2.4), it is depicted as an output because all other automata sharing this label are designed so that they do not prevent occurrence of the transition.

Generic Sample-and-Hold (SH) The connection between discrete-time controller and continuous-time plant can be modeled by SH elements as shown in Figures 22a and 22b. All SH elements use the clock τ and label *startOfPeriod* provided by the "controller and clock" component. The output b is updated to the value of the input a once per period, within $\underline{\Delta t} \le \tau \le \overline{\Delta t}$, and held constant in between ($\dot{b} = 0$). In accordance with the timing model, the parameters $\overline{\Delta t}$ and $\underline{\Delta t}$ are restricted to $-T/2 < \underline{\Delta t} \le \overline{\Delta t} < T/2$.

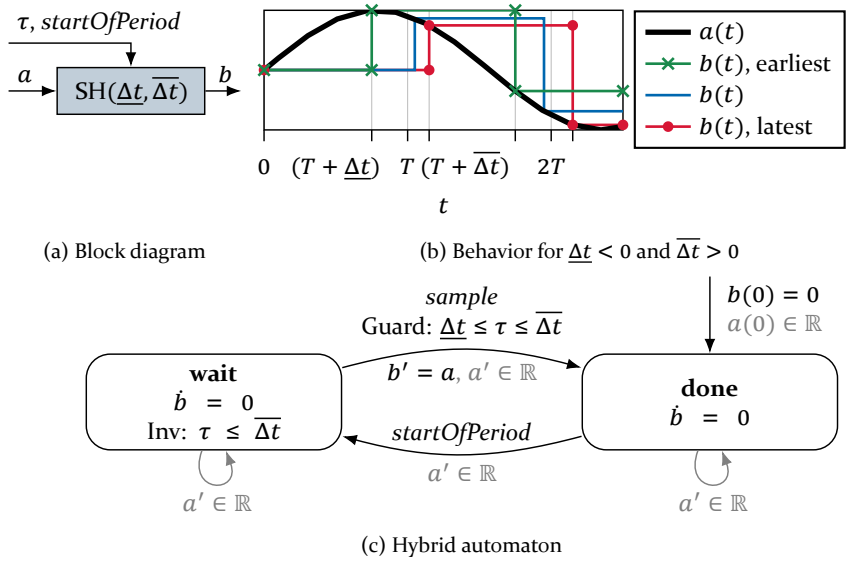
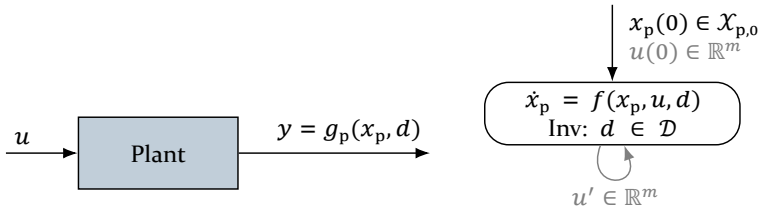


Figure 22: Specification of sample-and-hold (SH) element

Figure 22c shows the corresponding automaton. The gray-colored parts model that $a(t)$ is an external input. The SH operation is modeled using may-semantics: Sampling *may* happen while $\underline{\Delta t} \leq \tau \leq \overline{\Delta t}$ but *must* happen before $\tau > \overline{\Delta t}$. The SH automaton is initialized at location "done" to match to the specified behavior of not sampling in the startup phase. The *startOfPeriod* synchronization label, which occurs at the transition from $\tau = T/2$ to $\tau = -T/2$, is used to reset the automaton at the beginning of each period.

Plant The plant is described the equations given in Section 3.1, which lead to the hybrid automaton of Figure 23. At its core, the plant dynamics are a continuous-time ODE. Nevertheless, the automaton contains a discrete transition to support the discontinuous input u due to sample-and-hold. To simplify the notation, d is considered as state variable of the automaton, leading to the unusual result that there are no dynamics for d, equivalent to $\dot{d} \in \mathbb{R}^{n_{\text{dist}}}$. No extra transition is required for d because it is assumed to be differentiable. For simplicity, the output equation $y = g_{\text{p}}(x_{\text{p}}, d)$ is noted directly in the block diagram.

Interconnection As shown in Figure 24, the closed loop is modeled by connecting the controller and plant with one sample-and-hold block per input and output component. All timing-dependent components share the common clock τ and the corresponding *startOfPeriod* synchronization label.



(a) Block diagram (b) Hybrid automaton

Figure 23: Plant as HA

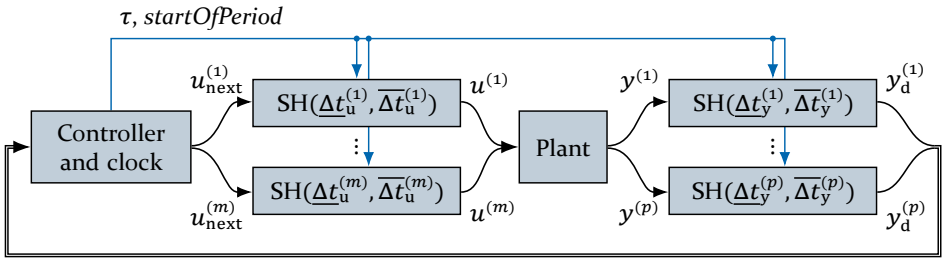


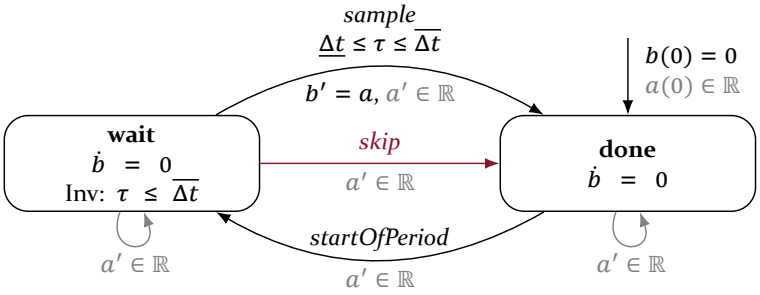
Figure 24: Block diagram of closed control loop

Alternative Startup Behavior A different startup behavior could be modeled by starting at "wait" and $\tau = -T/2$, which would be equal to starting at $t = T/2$ with the first regular period $k = 1$. It is not generally possible to initialize $\tau = 0$ and still use "wait" as initial location, because the invariant $\tau \leq \overline{\Delta t}$ would be violated at the start if $\overline{\Delta t} < 0$.

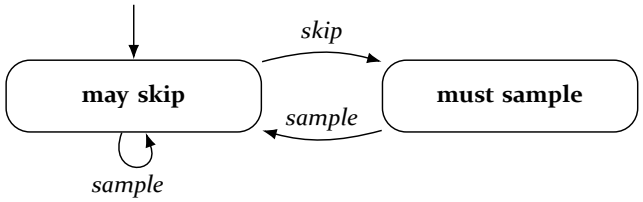
Skipping Skipping events is not modeled in the previous automata but could be implemented as illustrated in Figure 25: For every skippable event, an appropriate bypass transition must be added to the corresponding HA, as exemplified for SH in Figure 25a.

Note that this automaton has two transitions from "wait" to "done". Strictly speaking, this is not supported by the definition of HA in Definition 4.2.1. However, can be easily solved by extending the definition or by introducing an additional location.

At first, this would always allow skipping. Therefore, a skip policy is modeled as HA and connected to the SH element by the *skip* and *sample* labels. For example, (M, K)-weak execution (Definition 3.1.1) can be represented as automaton, whose size however grows rapidly with increasing K (Horssen et al., 2016). Figure 25b shows a $(1, 2)$ policy, i.e., one out of any two consecutive samples may be skipped.



(a) SH automaton modified to support skipping



(b) $(1, 2)$-weak skip policy as HA

Figure 25: Modified HA to support skipping

4.2.10 Experiments with Direct Reachability Analysis

In principle, the HA presented in the previous section can be directly used as input for existing software for reachability analysis. In the following, this approach is put to the test for a number of examples, ranging from simple first-order systems up to the angular rate control of a linearized quadrotor helicopter. The results will show that unmodified reachability analysis is of limited use here, particularly due to the wrapping effect. Hence, specialized analysis methods are required, which will be the subject of Chapter 5.

A previous version of this section was published in Gaukler and Ulbrich, 2019, which contains further implementation details omitted here. The source code for the following experiments is referenced in Appendix F.

Implementation To apply existing software for reachability analysis, the example systems from Appendix A were generated as HA in the format of the analysis tool *SpaceEx* (Frehse, Le Guernic, et al., 2011). The HA models depicted earlier can be entered into SpaceEx with minimal modifications. The gray-colored parts were omitted because SpaceEx explicitly supports inputs (“uncontrolled variables”).

SpaceEx was chosen for two reasons: First, it is a mature tool that performs well on benchmark examples from current literature (Althoff, Bak, et al., 2020), despite its last release being from 2015. Second, its model format is supported as a de-facto standard for the exchange of HA: Using the *HyST* conversion tool (Bak, Bogomolov, et al., 2015) and its *HybridPy* interface, an HA in SpaceEx format can be sent to multiple other state-of-the-art tools, particularly *Flow** (Chen et al., 2013) and *Hylaa* (Bak and Duggirala, 2017). However, no useful results could be obtained with these two tools, and an exhaustive survey of other tools would go beyond the scope of this thesis. The following experiments are therefore limited to SpaceEx. Nevertheless, the difficulty of the problem generally translates to other tools.

For the goal of safety analysis, the reachable set $\mathrm{Reach}_x(H)$ over an infinite time horizon is considered. This is the case that SpaceEx is designed for. For better visualization, an additional variant of the model is used. Herein, the time-dependent reachable set $\mathrm{Reach}_x(H, t)$ is computed by adding a state t with dynamics $t(0) = 0, \dot{t} = 1$.

As ground truth for comparison, random trajectories were simulated using the *PySim* simulator included in HyST and an adapted HA model. These simulations provide an *inner* approximation of the reachable set.

For SpaceEx, the best settings were determined experimentally. These are `scenario = stc`, `directions = oct` (sets are generalized octagons) and `set-aggregation = "none"` (no over-approximation at discrete transitions). It must be noted that for the scenario used here, SpaceEx uses *unsound numerics*, i.e., the effect of numerical error on the reachability computations is not considered, so that the computed set may be slightly too small. This simplification is employed in many reachability tools to improve performance.

Results Figure 26 compares simulated trajectories (left) with the outer approximation of the reachable set $\mathrm{Reach}_x(H, t)$ determined by SpaceEx (right). If SpaceEx successfully found a finite bound on the *infinite-time* reachable set $\mathrm{Reach}_x(H)$, this is marked by "✔" (right), else by "✗".

The figure starts with the first-order system A1 ($n_\mathrm{p} = n_\mathrm{d} = m = p = 1$) and variants thereof: For small timing deviation (A1), the approximated reachable set matches the simulations within some pessimism. Hence, reachability analysis would be practical for this system. With increasing timing deviation

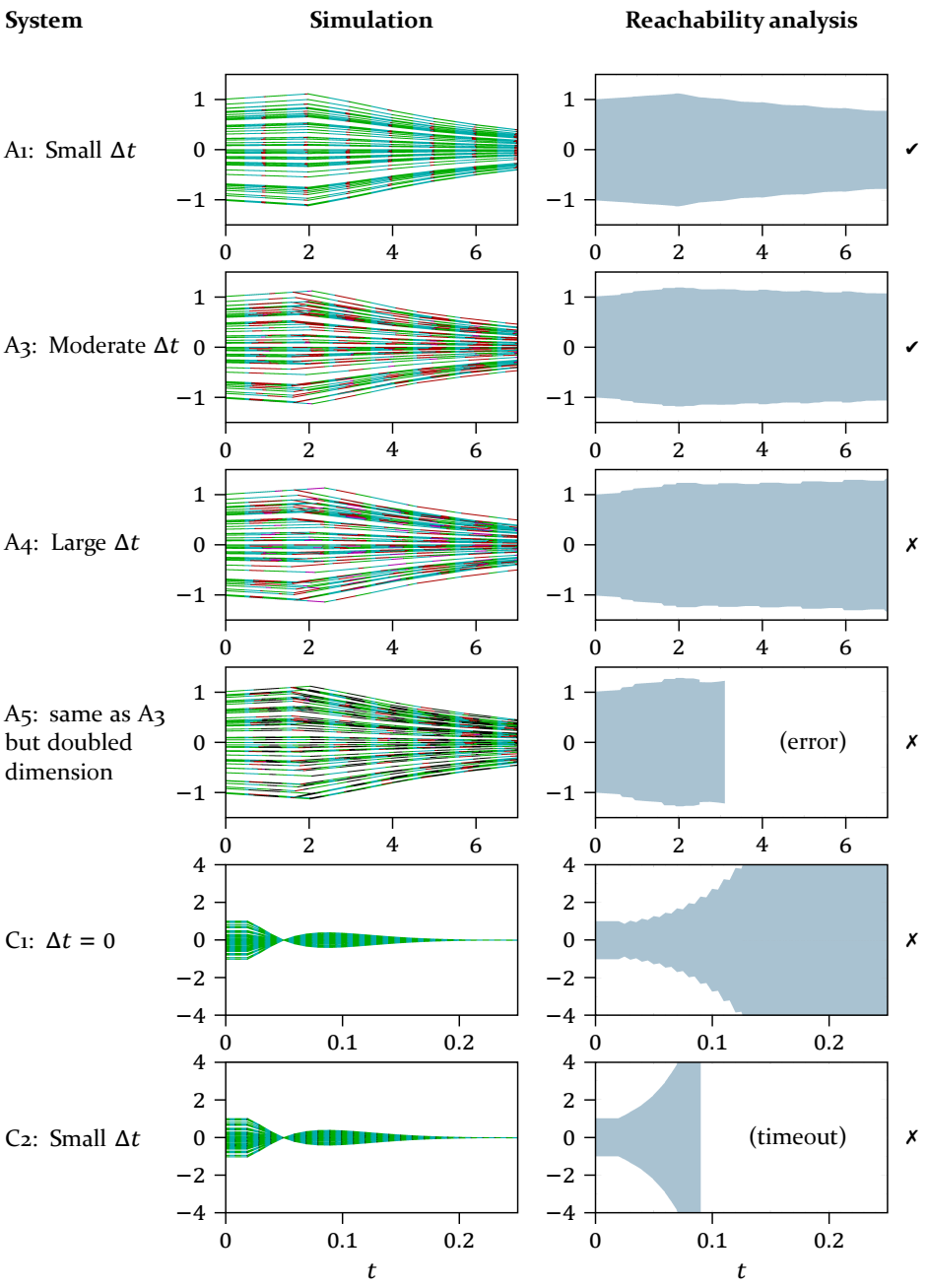


Figure 26: Comparison of random simulations (left) and the overapproximated reachable set over t computed by SpaceEx (right) for several example systems. The plots show $y^{(1)}(t)$. The varying line color in the left plot depicts the different states of the hybrid automaton. The mark indicates whether SpaceEx successfully computed a finite reachable set (✔) or not (✗).

(A3, A4), the result becomes worse until eventually safety can no longer be shown because the reachable set diverges over time (A4). This may be due to the wrapping effect or due to actual instability. The latter, however, seems less likely because the simulation does not show instability.

Next, systems of higher dimension are considered. Doubling the dimension of A3 by block-diagonal repetition leads to system A5 ($n_\mathrm{d} = n_\mathrm{p} = m = p = 2$), which has the same behavior but is more difficult to analyze unless the block structure is considered explicitly. Analysis of this system fails with an internal error in SpaceEx, hence the depicted set is incomplete.

Similar problems are experienced for examples inspired by real-world applications: Example C represents the angular rate control for one of the three axes of rotation of a quadrotor helicopter ($n_\mathrm{p} = m = p = 1, n_\mathrm{d} = 2$). Even with ideal timing (C1), analysis is not successful with SpaceEx. Increasing the timing uncertainty (C2) yields an even worse result. The computation was aborted early because it did not finish within several hours. Yet, it can be clearly seen that the reachable set diverges.

Not depicted here are examples D1 and D2, which model a three-axis angular rate control ($n_\mathrm{d} = 6, m = 4, p = 3, n_\mathrm{p} = 3$) and are an extension of examples C1 and C2. As can be expected, the results are again worse than for example C.

Discussion In summary, directly using the HA model for reachability analysis is possible in theory but difficult in practice. For many examples, safety can not be shown, as the approximated reachable set diverges over time. In general, this can be due to either actual instability or the wrapping effect. Because systems C1, C2, D1 and D2 will be proven stable later in this thesis, the issue is clearly due to the wrapping effect.

It is important to state that better results may be obtained with other tools or other settings. An exhaustive search was, however, not possible within the scope of this doctoral thesis. Still, the results represent the general difficulty posed to any reachability analysis tool by real-time control systems with uncertain timing. As will be seen later, an appropriate preprocessing can yield good results with the same tool and settings as used here.

In theory, the reachable set for *perfect* timing, linear dynamics and zero disturbance, as in examples C1 and D1, can be computed exactly: Because the transitions occur at fixed times, the solution can be computed analogous to the discretization of a LIS: If the LIS solution is denoted $x(t) = \Phi(t)x_0$, then the reachable set is $\mathrm{Reach}_x(\cdot, t) = \Phi(t)\mathcal{X}_0$. Accordingly, in reachability analysis one could exploit that a discrete transition from x to Mx at a fixed time can

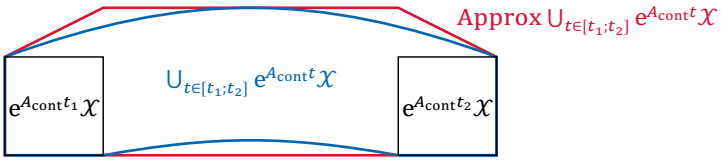


Figure 27: In usual implementations, the reachable set over a time interval can not be represented exactly. In this simplified illustration, the curved shape of the reachable set (blue) can not be described exactly by a polytope with a finite number of vertices. An outer approximation (red) inevitably leads to pessimism.

be computed exactly without the Approx operator. For example, if polytopes are used as numerical set representation, the set $\mathcal{X}$ can be transformed to $M\mathcal{X}$ without any approximation step and with the same amount of required memory. In principle, this special handling can be enabled in SpaceEx by the option `--map-zero-duration-jump-sets`, however, then the analysis fails with an error related to numerical difficulties that will be discussed soon.

The theoretical benefit of an exact computation for ideal timing does not extend to uncertain timing, as then the union over some time interval, roughly $\bigcup_{t\in[t_1;t_2]} e^{A_{\text{cont}}t}\mathcal{X}$, must be computed but can usually not be represented exactly. This is sketched in Figure 27. The difficulties observed in the experiment correspond to the fundamental challenges of numeric reachability analysis discussed earlier, particularly the pessimism for systems with frequent discrete transitions. These problems increase with system dimension and timing uncertainty.

A probable cause of the error occurring for example A5 is that set representations become numerically ill-conditioned: For example, after sampling the output, y_d and x_p are linearly dependent, up to a small deviation due to timing uncertainty and, in general, also measurement noise and nonlinearity. In a geometrical interpretation, the reachable set almost collapses to a planar set with zero volume. Correspondingly, the matrices representing the set, e.g., as basis vectors of a polytope (Althoff, 2010, Chapter 2), are close to a rank defect, causing numerical difficulties in computing set unions and intersections. This structural challenge was reproduced in a simplified implementation by Lappe, 2018a, Section 4.4.

4.3 Concluding Remarks

Uncertain input/output timing adversely affects the performance of real-time control systems but is virtually unavoidable in modern computing systems. To ensure safety, one must prove a bound on the worst-case behavior, such as the maximum position error of a quadrotor helicopter, in the presence of timing uncertainty. To address this challenge, this chapter presented two possible formal frameworks that capture both the discrete-time and continuous-time aspects of real-time control systems: hybrid automata (HA) and linear impulsive systems (LIS). Previous versions of parts of this section were published in Gaukler and Ulbrich, 2019 and Gaukler, 2020a.

From the perspective of formal analysis, HA are ideal because they can express both the dynamics and the timing of the real-time control system in an unambiguous, machine-readable notation. This format can be used for formal proofs as well as numerical analysis. However, from a practical perspective, the expressiveness of HA comes with the drawback of complicated definitions and analysis methods that often fail to produce a useful result. In theory, safety verification of HA is easy, since powerful generic tools for set-valued reachability analysis exist. However, practical experiments suggest that direct verification is only possible for simple examples.

This limitation can be explained by the structural problem of set-based reachability analysis: Because it typically performs overapproximations at discrete transitions to keep computational effort acceptable, it is best suited for systems with few discrete transitions and stable continuous dynamics; the opposite of this is the case for real-time control systems. Hence, model reduction to purely continuous dynamics will be explored in the following chapter.

In comparison, LIS are highly limited but allow for an explicit discretization. This leads to a linear discrete-time system with model uncertainty, for which existing methods can be reused. However, all difficult parts are excluded from the LIS model: Nonlinearities are not allowed and the translation of uncertain timing to a sequence of events is not included in the LIS but defined by additional equations. As will be seen in the next chapter, analysis of a pure LIS with explicitly given events would be easy but the necessary co-analysis of LIS and event model requires substantial effort. In the following, two important connections between LIS and HA are discussed.

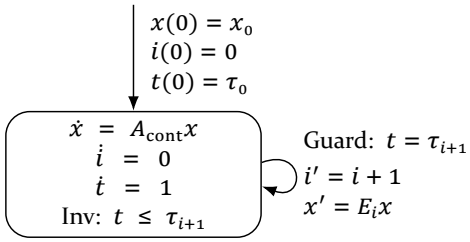


Figure 28: Linear impulsive system formalized as hybrid automaton

4.3.1 Interpretation of Impulsive Systems as Hybrid Automata

Linear impulsive systems are a special case of hybrid automata, as a LIS per Definition 4.1.8 and Remark 4.1.10 is equivalent to the hybrid automaton in Figure 28. Unlike in LIS, the transitions of HA can not depend on time, which is solved by introducing t as an extra state variable with slight abuse of notation.

4.3.2 Time Semantics

Hybrid automata and linear impulsive systems employ a different notion of time, as will be illustrated by the example in Figure 29. To avoid ambiguities, the following discussion uses a generalized time (t, j) as in Michel et al., 2001, Chapter 8: In addition to the classical time variable t, an event number $j \in \mathbb{N}_0$ is used. This is useful if multiple events occur at the same time, e.g., here $x(t_1, 1)$ can be distinguished from $x(t_2, 2)$ even if $t_1 = t_2$.

In hybrid automata, time does not elapse during discrete transitions. For example, if a discrete transition occurs at t_3, "$x(t_3)$" is ambiguous as it refers to both the value $x(t_3, 2)$ before and $x(t_3, 3)$ after a transition. Formally, the set $\text{Reach}_x(H, t_3)$ will then contain both values $x(t_3, 2)$ and $x(t_3, 3)$, so there is no unique function $x(t)$.

In this respect, HA are different from LIS, where, for the simple definition

$$\dot{x}(t) = A_{\text{cont}}x, \quad t \neq t_i, \tag{98a}$$

$$x(t_i) = E_i x(t_i^-), \quad i \in \mathbb{N}_1, \tag{98b}$$

$x(t_i) = x(t_i, i)$ is the unique value after the transition, while the value before is referenced by the left-side limit $x(t_i^-) = x(t_i, i-1)$. While this notation may be more intuitive, it is impossible if multiple transitions occur at one time, which in the example here occurs for $t_1 = t_2$. Hence, the extended

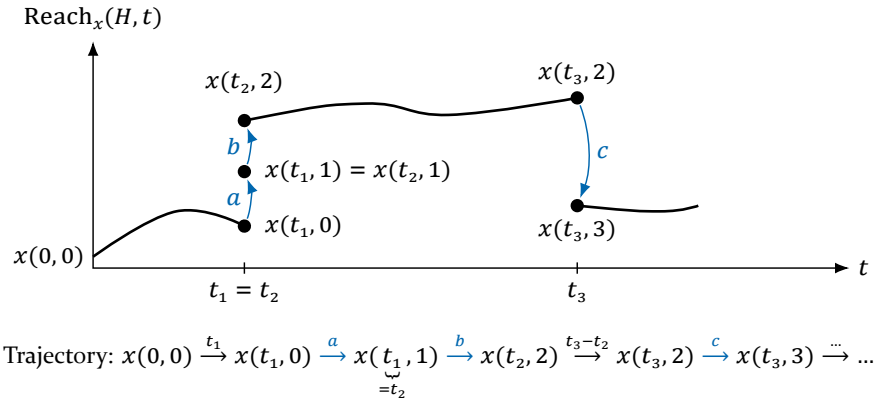


Figure 29: Reachable set and trajectory of a hybrid automaton with two discrete events at the same time t_1. In this illustration, the state at time t after j events is denoted $x(t,j)$. Filled dots denote that the reachable set contains multiple elements at times t_1 and t_3.

definition of LIS with concurrent events (Definition 4.1.8 and Remark 4.1.10) instead overwrites $x(t_i)$ with the new value after the event: For $t_1 = t_2$, there is no time t between the first and second event. Therefore, $x(t_2) = x(t_2, 2)$, while the intermediate value $x(t_1, 1)$ never occurs in $x(t)$ for any t. This is avoided in HA by considering trajectories as a sequence of transitions and not as a function $x(t)$ of time.

4.3.3 Summary

Real-time control systems can be modeled as hybrid automata or linear impulsive systems, where none of these two frameworks is clearly better than the other. Instead, they are complementary and start from different directions towards the goal of successful analysis with moderate complexity: LIS start with low expressiveness, restricted to linear dynamics and fixed event times, and add complexity via the event model. HA instead start with the full expressiveness of an almost arbitrary hybrid system, which must be reduced in most cases to allow for successful analysis. Hence, the next chapter presents an analysis method for each of the two modeling frameworks.

5 Design-Time Analysis

Due to the timing variations discussed in Section 2.3, the practical implementation of a controller with period T will often result in a *nearly-periodic* system in which the sensor and actuator times do not lie on the intended periodic grid $t = kT$, $k \in \mathbb{N}_1$ but in a small *time window* around these points, as defined in Section 3.1. If this window is too large, the resulting dynamics may violate safety or stability goals. Hence, analysis is required to determine the permissible amount of timing uncertainty. If analysis succeeds, the constraints on the real-time system can be relaxed without giving up safety.

In this chapter, two approaches for design-time analysis are presented: a stability analysis using LIS (Section 5.1) and a safety analysis using HA (Section 5.2). Then, Section 5.3 summarizes and compares both approaches.

5.1 Stability Analysis using Linear Impulsive Systems

This section explores stability analysis in the framework of linear impulsive systems (LIS). The results of this section were previously published in Gaukler, Roppenecker, et al., 2020 and the corresponding technical report (Gaukler, Roppenecker, et al., 2019).

Outline First, the problem statement, notation and definitions are introduced. Then, Section 5.1.3 presents an overview of the approach and the derivations that follow in the subsequent sections. Finally, Section 5.1.9 shows an experimental evaluation and Section 5.1.10 discusses the findings.

5.1.1 Problem Statement

Given A linear MIMO real-time control system (per Section 3.1) that is exponentially stable for ideal timing is executed with uncertain timing.

Goal As formalized in Section 3.2, the goal is to prove exponential stability (CGES) of the real-time control system for moderate timing uncertainties and no skipped executions. Given the challenging computational complexity of the MIMO case (cf. Section 2.5.4), the focus is on an efficient solution that scales well to systems with a large number of inputs and outputs, even if this scalability makes the result more pessimistic and therefore the approach is only applicable to small timing uncertainties.

Assumptions In this section, only the linear case is examined and it is assumed that no events are skipped. Because exponential stability considers autonomous systems, disturbance and measurement noise are set to zero.

Also, the assumptions of the LIS-based discretization (Section 4.1.4.6) are inherited: The system is modified to start at $t = t_0 = -T/2$ with a complete control period $k = 0$. This modification is not relevant for exponential stability, as it effectively only changes the initial state.

5.1.2 Preliminaries and Notation

Definition 5.1.1 (Matrix-Valued Limit)**.** Analogous to the classical epsilon-delta-definition (Stover, 2019), define a matrix- or vector-valued limit as

$$\lim_{x\to a} f(x) = y \quad :\Leftrightarrow \quad \Big(\forall \epsilon > 0 \quad \exists \delta(\epsilon) > 0 \text{ such that } \forall x \text{ with } \|x-a\|_X < \delta : \quad \|f(x)-y\|_Y < \epsilon \Big), \tag{99}$$

where $\|\cdot\|_X$ and $\|\cdot\|_Y$ are arbitrary norms. ◁

Due to the equivalence of norms (Theorem 4.1.3), the result y is independent of the chosen norms $\|\cdot\|_X$ and $\|\cdot\|_Y$.

Positive definite functions and matrices are denoted by $f(x) \succ 0$ and $F \succ 0$:

Definition 5.1.2 (Positive Definiteness)**.** For functions $f, g : \mathbb{R}^n \mapsto \mathbb{R}$, define

$$f(x) \succ g(x) \quad :\Leftrightarrow \quad \left(\begin{cases} f(x) > g(x), & x \neq 0 \\ f(x) = g(x), & x = 0 \end{cases} \quad \forall x \in \mathbb{R}^n \right). \tag{100}$$

For the symmetric matrices $F = F^\top, G = G^\top \in \mathbb{R}^{n\times n}$, define the abbreviation

$$F \succ G \quad :\Leftrightarrow \quad x^\top F x \succ x^\top G x. \tag{101}$$

To define negative definiteness ($\prec$), the relation $>$ is replaced by $<$. ◁

The restriction to symmetric F and G simplifies the further derivations but does not restrict the results because only the symmetric part $(F + F^\top)/2$ of a matrix F contributes to the quadratic form: For all $y \in \mathbb{R}$, $y = (y + y^\top)/2$, so

$$\underbrace{x^\top F x}_{\in\mathbb{R}} = \frac{x^\top F x + (x^\top F x)^\top}{2} = x^\top \left(\frac{F + F^\top}{2} \right) x \quad \forall x \in \mathbb{R}^n, F \in \mathbb{R}^{n\times n}. \tag{102}$$

Theorem 5.1.3 (Cholesky Decomposition)**.** *Any $P \succ 0$ can be uniquely decomposed into $P =: P^{1/2}(P^{1/2})^\top$ such that $P^{1/2}$ is lower triangular with positive diagonal entries and therefore also invertible (Bernstein, 2009, Fact 8.9.38).*

5.1.3 Approach

This section presents the high-level structure of the proposed approach. Details are given in the subsequent sections.

Discretization The approach starts with the time discretization as given in Section 4.1.4. In the following, the subscript k of timing variables $\Delta t_{\dots}$ is often omitted. The timing-dependent transition matrix is denoted $A(\Delta t)$.

As shown in Theorem 4.1.13, CGES and DGES are equivalent for the given system. Because there are no skips, the timing uncertainty in every period is the same, so DGES is a question of *robust stability* of the uncertain system

$$x_{k+1} = A_k x_k, \quad A_k \in \mathcal{A} \subset \mathbb{R}^{n\times n} \tag{103}$$

for all $A_0, A_1, \dots \in \mathcal{A}$. Here,

$$\mathcal{A} = \{A(\Delta t) \mid \underline{\Delta t}^{(i)}_{\{\mathrm{u,y}\}} \le \Delta t^{(i)}_{\{\mathrm{u,y}\}} \le \overline{\Delta t}^{(i)}_{\{\mathrm{u,y}\}}\} \tag{104}$$

is the set of possible transition matrices A_k for all possible values of the timing variables $\Delta t^{(i)}_{\{\mathrm{u,y}\}}$. The following approach will show DGES by the CQLF

$$V_P(x) \coloneqq x^\top P x \succ 0 \quad \text{with} \quad V_P(A_k x) \prec V_P(x) \quad \forall A_k \in \mathcal{A}, \tag{105}$$

where the parameter $P \in \mathbb{R}^{n\times n}$, $P \succ 0$ needs to be determined.

Difficulty As discussed in Section 2.5.4, two obstacles have to be overcome to make discretization-based stability analysis computationally feasible: The dimension $m + p$ of the timing vector Δt can lead to exponentially growing computational complexity. Furthermore, an explicit expression for A_k contains $(m + p)!$ case distinctions due to the number of possible event orderings.

Decomposition (Section 5.1.4) To avoid these difficulties, the dynamics are broken up into a sum using a novel decomposition:

Theorem 5.1.4 (Decomposition). *If there are no skips, the transition matrix $A(\Delta t)$, which depends on $m+p$ scalar timing variables $\Delta t^{(i)}_{\{\mathrm{u,y}\}}$, can be split into a sum of functions of one scalar parameter each:*

$$A(\Delta t) = A(\Delta t = 0) + \sum_{i=1}^{m} \Delta A_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)}) + \sum_{j=1}^{p} \Delta A_{\mathrm{y},j}(\Delta t_{\mathrm{y}}^{(j)}) + \sum_{i=1}^{m}\sum_{j=1}^{p} \Delta A_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}), \tag{106}$$

where $A(\Delta t = 0)$ is the nominal case and $\Delta A_{\dots}$ are "deviations" that obey $\lim_{|\Delta t|\to 0} \Delta A_{\dots} = 0$.

Proof. The proof and expressions for $\Delta A_{\dots}$ will be given in Section 5.1.4. □

Loosely interpreted, $\Delta A_{\mathrm{u},i}$ is the deviation of A resulting from the timing of the i-th actuator, $\Delta A_{\mathrm{y},j}$ corresponds to the j-th sensor, and $\Delta A_{\mathrm{uy},i,j}$ to the influence of actuator i on sensor j.

Stability by Norm Bounding (Sections 5.1.5 to 5.1.7) As assumed in the problem setting, the nominal case (ideal timing $\Delta t = 0$) is stable and therefore achieves DGES with $\rho < 1$. The resulting safety margin $1 - \rho > 0$ can be used to prove stability up to a certain amount of timing deviation. For this, a matrix norm corresponding to a CQLF is introduced:

Definition 5.1.5 (P-ellipsoidal Norm). Let $V_P(x) = x^\top P x$, $P \in \mathbb{R}^{n\times n}$, be a positive definite (Lyapunov candidate) function, i.e., $P \succ 0$. Then, the *P-ellipsoidal norm* is defined as the matrix norm induced by $\sqrt{V_P(x)}$:

$$\|A\|_P \coloneqq \max_{x\neq 0} \sqrt{\frac{V_P(Ax)}{V_P(x)}}. \tag{107}$$

◁

This norm $\|A\|_P$ represents the worst-case decay of $V_P(x)$ for a time-invariant system $x_{k+1} = Ax_k$:

$$V_P(Ax) \leq \|A\|_P^2 V_P(x) \quad \forall x. \tag{108}$$

In general, norm bounds can be highly pessimistic. However, with the right P, this norm can *accurately* capture stability of the *nominal* case $x_{k+1} = A(\Delta t = 0)x_k$, for which $\rho\{A(\Delta t = 0)\} < 1$ is the minimal possible stability factor ρ for DGES. In other words, DGES of any linear *time-invariant* system is accurately captured by corresponding a *quadratic* Lyapunov function.

Theorem 5.1.6 (Accuracy in the Nominal Case)**.** *There exists P such that $\rho_0 \coloneqq \|A(\Delta t = 0)\|_P$ is arbitrarily close to $\rho\{A(\Delta t = 0)\}$.*

Proof. This will be detailed in Section 5.1.5, Theorem 5.1.17. □

This limit $\rho_0 = \rho\{A(\Delta t = 0)\}$ is in fact the best ρ for DGES of the time-invariant system $x_{k+1} = A(\Delta t = 0)x_k$:

Theorem 5.1.7. *For $x_{k+1} = Ax_k$, the stability factor ρ in DGES(ρ, α) is at least the spectral radius of A: $\rho \geq \rho\{A\}$.*

Proof. See Appendix B.1. □

To check stability under uncertain timing, choose any $P \succ 0$ for which $\rho_0 < 1$. This exists by Theorem 5.1.6; the implementation is discussed later. Then, stability can be shown if the summands $\Delta A_{\dots}$ in (106), which represent timing deviation, are small enough:

Theorem 5.1.8 (Norm Bounding)**.** *The control loop is DGES if*

$$\begin{aligned}\Bigg(&\underbrace{\|A(\Delta t = 0)\|_P}_{=\rho_0} + \sum_{i=1}^{m} \|\Delta A_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)})\|_P \\ &+ \sum_{j=1}^{p} \|\Delta A_{\mathrm{y},j}(\Delta t_{\mathrm{y}}^{(j)})\|_P \\ &+ \sum_{i=1}^{m}\sum_{j=1}^{p} \|\Delta A_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)})\|_P \qquad \Bigg) < 1 \\ &\forall \Delta t_{\{\mathrm{u,y}\}}^{(i)} \in \left[\underline{\Delta t}_{\{\mathrm{u,y}\}}^{(i)};\ \overline{\Delta t}_{\{\mathrm{u,y}\}}^{(i)}\right]. \end{aligned} \tag{109}$$

Proof. Consider $\|A(\Delta t)\|_P$ and apply Theorem 5.1.4 and the triangle inequality (29c) to see that (109) implies $\|A(\Delta t)\|_P < 1$ for all possible Δt. This leads to DGES, as it will be detailed in Section 5.1.5, Theorem 5.1.19. □

For the practical implementation, upper bounds for $\|\Delta A_{\dots}\|_P$ are computed in Section 5.1.6 and P is determined using LMIs in Section 5.1.7.

Benefits and Drawbacks Due to the norm bounds, the approach entails some conservatism, which will later be evaluated by experiments in Section 5.1.9. On the other hand, the chosen method is particularly well-suited for analyzing MIMO systems with moderate timing uncertainty:

Theorem 5.1.9 (Stability implies timing robustness)**.** *If the nominal case $\Delta t = 0$ is stable, then Theorem 5.1.8 can show stability for some nonzero (possibly small) time window.*

Proof. Consider the summands $\rho_0 + \sum \|\Delta A_{\dots}\|_P$ in (109). Assume a timing bound $|\Delta t| < \delta$ with sufficiently small $\delta > 0$.

For the first summand ρ_0, the assumption of nominal stability $\rho\{A(\Delta t = 0)\} < 1$ means that due to Theorem 5.1.6, it is possible to choose P such that $\rho_0 < 1$.

The next step is to bound the remaining part of the sum below $1 - \rho_0$. Due to Theorem 5.1.4, $\Delta A_{\dots} \to 0$ for $\Delta t \to 0$. By the definition of a matrix-valued limit (Definition 5.1.1), this implies that for any desired bound $\epsilon > 0$ on the norm $\|\Delta A_{\dots}\|_P$ of the deviations $\Delta A_{\dots}$ from the nominal case, there is a corresponding timing bound $\delta(\epsilon) > 0$ such that $(|\Delta t| < \delta \Rightarrow \|\Delta A_{\dots}\|_P < \epsilon)$. Choose ϵ small enough such that the condition of Theorem 5.1.8 is satisfied. Then, by the previous results, $\delta(\epsilon) > 0$ and the system is stable for $|\Delta t| < \delta(\epsilon)$. □

Because nominal stability ($\rho_0 < 1$) must hold for the result of any controller design method, this has two important consequences:

- In theory, the approach is always guaranteed to return some nonzero timing range. In practice, numerical issues of the implementation may prevent success if ρ_0 is very close to 1.
- Any real-time control system of the considered form that is stable for ideal timing is also stable for a small amount of timing deviation, even if timing or robustness were not considered in the design.

Remark 5.1.10 (Complexity). With increasing number of sensors and actuators, checking Theorem 5.1.8 requires only a *quadratically* increasing number of matrix norm computations. The approach therefore avoids the exponential or worse growth caused by other methods (cf. Section 2.5.4). In detail, the computation consists of determining P, ρ_0, and then $p + m + mp$ bounds on one-dimensional functions $\|\Delta A_{\dots}(\delta)\|_P$, where δ is a bounded scalar variable.

While mp grows only quadratically, a further factor of growth is the increasing dimension $n \times n$ of A and P. Hence, the overall growth in computational complexity is beyond quadratic. Further speedup could be gained by exploiting the sparsity of A and $\Delta A_{\dots}$. ◁

Remark 5.1.11 (Interpretability). Because each summand $\|\Delta A_{\dots}\|_P$ in Theorem 5.1.8 only refers to the timing of at most one sensor and one actuator, its maximum loosely corresponds to the amount of instability caused by the timing of one sensor, actuator or sensor-actuator-pair. This gives important hints on the timing sensitivity, which can be used to improve the design of the real-time system, e.g., to give priority to sensors with high sensitivity. ◁

The following sections present the low-level details of every analysis step. Section 5.1.9 then shows experimental results.

5.1.4 Decomposition

Now, Theorem 5.1.4 is proven, which states that the transition matrix A_k can be split into summands that depend on at most two timing variables. The proof is summarized in the following. A detailed version is detached to Appendix B.2.

Proof sketch of Theorem 5.1.4. Consider the transition matrix (56) of the k-th control period, as given in Section 4.1.4:

$$A(\Delta t) \overset{(56)}{=} E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev}}} - \tau_{N_{\text{ev}}-1})} \underbrace{\overbrace{\prod_{i=1}^{N_{\text{ev}}-1}}^{\sim} E_i \mathrm{e}^{A_{\text{cont}}(\tau_i - \tau_{i-1})}}_{=:X}. \tag{110}$$

The factor X in the above transition matrix only contains measurement and actuation events: In the following analysis of X, all matrices E_i are either $E_i = E_{\text{u},\dots}$ or $E_i = E_{\text{y},\dots}$. X can be rewritten as

$$X = \overbrace{\prod_{i=1}^{N_{\text{ev}}-1}}^{\sim} ((E_i - I) + I)\mathrm{e}^{A_{\text{cont}}(\tau_i - \tau_{i-1})}. \tag{111}$$

Expanding this product leads to a sum

$$\begin{aligned} X = {} & I\mathrm{e}^{\cdots} I\mathrm{e}^{\cdots} I\mathrm{e}^{\cdots} \cdots + (E_{\dots} - I)\mathrm{e}^{\cdots} I\mathrm{e}^{\cdots} I\mathrm{e}^{\cdots} \cdots + \dots \\ & + (E_{\dots} - I)\mathrm{e}^{\cdots} (E_{\dots} - I)\mathrm{e}^{\cdots} (E_{\dots} - I)\mathrm{e}^{\cdots} \cdots . \end{aligned} \tag{112}$$

This sum is simplified using structural properties of the event matrices E_i. In particular, all summands containing three or more factors $(E_{\dots} - I)$ are zero. Further rewriting leads to the desired result

$$A_k = A(\Delta t = 0) + \sum_{i=1}^{m} \Delta A_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)}) + \sum_{j=1}^{p} \Delta A_{\mathrm{y},j}(\Delta t_{\mathrm{y}}^{(j)}) + \sum_{i=1}^{m}\sum_{j=1}^{p} \Delta A_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}) \tag{113}$$

with

$$A(\Delta t = 0) = A_k|_{\Delta t_{\mathrm{y}}=0, \Delta t_{\mathrm{u}}=0} \tag{114}$$

$$\overset{(369)}{=} E_{\mathrm{ctrl}} \mathrm{e}^{A_{\mathrm{cont}} T/2} \left(I + \sum_{i=1}^{m} (E_{\mathrm{u},i} - I) + \sum_{j=1}^{p} (E_{\mathrm{y},j} - I) \right) \cdot \mathrm{e}^{A_{\mathrm{cont}} T/2}, \tag{115}$$

$$\Delta A_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)}) \overset{(371)}{=} E_{\mathrm{ctrl}} \mathrm{e}^{A_{\mathrm{cont}} T/2} (\mathrm{e}^{-A_{\mathrm{cont}} \Delta t_{\mathrm{u}}^{(i)}} - I)(E_{\mathrm{u},i} - I), \tag{116}$$

$$\Delta A_{\mathrm{y},j}(\Delta t_{\mathrm{y}}^{(j)}) \overset{(373)}{=} E_{\mathrm{ctrl}} (E_{\mathrm{y},j} - I) \mathrm{e}^{A_{\mathrm{cont}} T/2} (\mathrm{e}^{A_{\mathrm{cont}} \Delta t_{\mathrm{y}}^{(j)}} - I), \tag{117}$$

$$\Delta A_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}) \overset{(377)}{=} \begin{cases} 0, & \Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)} \leq 0, \\ E_{\mathrm{ctrl}} (E_{\mathrm{y},j} - I)(\mathrm{e}^{A_{\mathrm{cont}} (\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)})} - I) \cdot (E_{\mathrm{u},i} - I), & \text{else.} \end{cases} \tag{118}$$

Note that all $\Delta A_{\dots}$ are of the same form: $M_1(\mathrm{e}^{A_{\mathrm{cont}} \delta(\Delta t)} - I)M_2$ or zero. A detailed proof is given in Appendix B.2. □

The proof was validated by a numerical experiment: For examples A1, B1, C1 and D1, changed to random timing within $\pm 0.9999 T/2$, the values of $A(\Delta t)$ determined by the product (110) and by the sum (113) match. To handle numerical uncertainty, the results were computed in interval arithmetic and "matching values" was defined as that the resulting intervals intersect. The source code for the experiment is referenced in Appendix F.

Conjecture 5.1.12 (Decomposition with Skips)**.** *An analogous result to Theorem 5.1.4 holds if events are skipped.*

Proof Sketch. Instead of omitting the event, skipping can equivalently be modeled by using an identity event matrix: All events are executed, but the sensor event matrix $E_{\mathrm{y},i}$ is changed to

$$E_{\mathrm{y},i} = \begin{cases} \text{original } E_{\mathrm{y},i}, & \sigma_{\mathrm{y}}^{(i)} = 0, \\ I, & \sigma_{\mathrm{y}}^{(i)} = 1. \end{cases} \tag{119}$$

The same is applied for the actuators. The timing deviation of any of these skipped events can be chosen arbitrarily, as it will cancel out later.

Furthermore, two skipped events occurring at the same time t will lead to a duplicate identity-event $(E_i, \tau_i) = (I, t)$. The definition of the event count and set of events in (53) – (55) must be changed to support this case, e.g., using a multi-set that supports duplicate elements.

Now, the formulas of the case without skipping can be reused, because formally, all events are executed. The proof does not depend on the value of E_{ctrl}. Also, the event matrix properties used in the proof still hold for I instead of the original $E_{\{\mathrm{u},\mathrm{y}\},i}$. Therefore, the proof holds unchanged, except for substituting new event matrices and adding the dependency on skips in $\Delta A_{\dots}$. □

5.1.5 P-ellipsoidal Norm

This section derives results to use the P-ellipsoidal matrix norm for stability analysis. The proofs mainly build upon the close relation between the P-ellipsoidal matrix norm and the Lyapunov candidate function $V_P(x) \coloneqq x^\top P x$.

Theorem 5.1.13 (Existence of Quadratic Lyapunov Function)**.** *If and only if the* time-invariant *system $x_{k+1} = Ax_k$ is exponentially stable (i.e., DGES), there exists a quadratic Lyapunov function $V_P(x) \coloneqq x^\top P x \succ 0$ with $V_P(Ax) \prec V_P(x)$. Herein, $P \in \mathbb{R}^{n\times n}$ is a solution of*

$$A^\top P A - P = -Q, \qquad P \succ 0 \tag{120}$$

for any $Q \in \mathbb{R}^{n\times n}$ with $Q \succ 0$. (Bernstein, 2009, Proposition 11.10.5).

Theorem 5.1.14. *The P-ellipsoidal norm (Definition 5.1.5) is a submultiplicative matrix norm with $\|I\|_P = 1$.*

Proof. Since $P \succ 0$, $\sqrt{V_P(x)} = \sqrt{x^\top P x}$ is a vector norm (Bernstein, 2009, Fact 9.7.30). The P-ellipsoidal norm $\|\cdot\|_P$ is its equi-induced matrix norm, therefore submultiplicative with $\|I\|_P = 1$ due to Definition 4.1.5 and Theorem 4.1.7. □

Note that for all $P \succ 0$, $\|A\|_P < 1$ is equivalent to $V_P(Ax) \prec V_P(x)$. The P-ellipsoidal norm can therefore be interpreted as the matrix norm which is equivalent to a quadratic Lyapunov function. This norm is also related to the joint spectral radius, a generalization of the spectral radius to time-varying systems (Jungers, 2009, Section 2.3.7; Blondel et al., 2005).

The vector norm $\sqrt{V_P(x)}$ can also be seen as the euclidean norm after applying a coordinate transformation:

$$\sqrt{V_P(x)} = \sqrt{x^\top P x} = \sqrt{x^\top P^{1/2} (P^{1/2})^\top x} = |\underbrace{(P^{1/2})^\top x}_{z}|, \tag{121}$$

where $P^{1/2}$ is the Cholesky decomposition of P per Theorem 5.1.3. If V_P is a Lyapunov function, the transformed system is contractive, i.e., $|z_{k+1}| = \sqrt{V_P(x_{k+1})} \leq \sqrt{V_P(x_k)} = |z_k|$. The coordinate transformation of (121) leads to an explicit expression for the P-ellipsoidal norm:

Theorem 5.1.15. $\|A\|_P = \|(P^{1/2})^\top A (P^{1/2})^{-\top}\|_2$, *where* $^{-\top}$ *denotes the transposed inverse.*

Proof. Rewrite the P-ellipsoidal norm as

$$\|A\|_P \overset{\text{Definition 5.1.5, (121)}}{=} \max_{x \neq 0} \frac{|(P^{1/2})^\top A x|}{|(P^{1/2})^\top x|} \tag{122}$$

and change variables to z with $x = (P^{1/2})^{-\top} z$:

$$\|A\|_P = \max_{z \neq 0} \frac{|(P^{1/2})^\top A (P^{1/2})^{-\top} z|}{|z|} \tag{123}$$

$$\overset{\text{Definition 4.1.6}}{=} \|(P^{1/2})^\top A (P^{1/2})^{-\top}\|_2 . \tag{124}$$

□

For stability analysis by norm bounding, the norm should be as small as possible. The following two theorems compare the smallest possible value of $\|A\|$ for the P-ellipsoidal norm and for other submultiplicative norms.

Theorem 5.1.16 (Spectral Radius Bound via Matrix Norms)**.** *Every submultiplicative matrix norm* $\|\cdot\|_S$ *is at least as large as the spectral radius (Bernstein, 2009, Proposition 9.3.2):*

$$\|A\|_S \geq \rho\{A\} \quad \forall A \in \mathbb{R}^{n \times n} \quad \forall \|\cdot\|_S \tag{125}$$

The P-ellipsoidal norm $\|A\|_P$ can get arbitrarily close to this smallest possible value if P is optimized for a specific A. For stability analysis, this means that the P-ellipsoidal norm can correctly capture the stability of the nominal case.

Theorem 5.1.17 (Extremal Quadratic Lyapunov Function)**.** *If the time-invariant system $x_{k+1} = Ax_k$ is stable, i.e., $\rho\{A\} < 1$, then there exists a quadratic Lyapunov function $V_P(x)$ that proves a stability factor $\bar{\rho}$ arbitrarily close to the spectral radius $\rho\{A\}$:*

$$\forall A \in \mathbb{R}^{n\times n} \text{ with } \rho\{A\} < 1 \quad \forall \bar{\rho} \in (\rho\{A\}; 1)$$
$$\exists P \succ 0 : \quad \|A\|_P = \max_{x\neq 0} \sqrt{\frac{V_P(Ax)}{V_P(x)}} \leq \bar{\rho}. \quad (126)$$

Proof. See Appendix B.1. □

The previous theorem leaves an infinitesimal difference between the spectral radius and the best possible P-ellipsoidal norm:

Remark 5.1.18 (Extremal P-ellipsoidal Norm). In the general case, there is no lower P-ellipsoidal norm than the one guaranteed by Theorem 5.1.17. Especially, it is not generally possible to find P such that $\|A\|_P = \rho\{A\}$. ◁

Proof. See Appendix B.1. □

Finally, the following theorem relates norm bounds and exponential stability.

Theorem 5.1.19 (Robust Stability from Norm Bounds)**.** *Let $A_k = \sum_{i=0}^{N} A_{k,i} \in \mathbb{R}^{n\times n}$ with fixed N. Then, the system $x_{k+1} = A_k x_k$ is DGES$(\bar{\rho}, C)$ for some C if there are a submultiplicative matrix norm $\|\cdot\|_S$ and a bound $0 \leq \bar{\rho} < 1$ such that $\sum_i \|A_{k,i}\|_S \leq \bar{\rho}\ \forall k$.*

Proof. Assume $\sum_i \|A_{k,i}\|_S \leq \bar{\rho} < 1\ \forall k$. The triangle inequality (29c) leads to

$$\|A_k\|_S = \left\| \sum_{i=0}^{N} A_{k,i} \right\|_S \overset{(29c)}{\leq} \sum_{i=0}^{N} \|A_{k,i}\|_S \leq \bar{\rho}. \quad (127)$$

Due to the equivalence of norms (Theorem 4.1.3), there is a finite $\alpha > 0$ such that $\|M\|_2 \leq \alpha \|M\|_S$ for all $M \in \mathbb{R}^{n \times n}$. This leads to

$$|x_{k+1}| = \left| \left(\overleftarrow{\prod_{j=0}^{k}} A_j \right) x_0 \right| \overset{(33)}{\leq} \left\| \overleftarrow{\prod_{j=0}^{k}} A_j \right\|_2 |x_0| \tag{128}$$

$$\overset{(30)}{\leq} \alpha \left\| \overleftarrow{\prod_{j=0}^{k}} A_j \right\|_S |x_0| \overset{(31)}{\leq} \alpha \overleftarrow{\prod_{j=0}^{k}} \left\| A_j \right\|_S |x_0| \overset{(127)}{\leq} \alpha \bar{\rho}^k |x_0| \quad \forall x_0 \in \mathbb{R}^n. \tag{129}$$

This holds for all $k \geq 0$ and therefore proves DGES$(\bar{\rho}, \alpha)$. □

5.1.6 Norm Bounding of Timing-Dependent Summands

Theorem 5.1.8 provides a stability result based on the norm $\|\Delta A_{\dots}\|_P$ of the timing-dependent deviations. This section presents a bound for this norm using the general form $M_1(\mathrm{e}^{A\delta} - I)M_2$ of $\Delta A_{\dots}$ shown in Section 5.1.4.

Problem Statement Compute a bound on $\max_{\delta \in [-\bar{\delta}; \bar{\delta}]} \|M_1(\mathrm{e}^{A\delta} - I)M_2\|_P$. The values of $M_1, M_2, A \in \mathbb{R}^{n \times n}$ and $\bar{\delta} \geq 0$ are given in Appendix B.2.4. Note that the timing δ is pessimistically extended to a symmetric interval.

Interpretation This problem is similar to reachability analysis with ellipsoidal sets: The output $y = M_1 x$ of a system $\dot{x} = Ax$ with initial state $x_0 = M_2 z_0$ is $y(t) = M_1 \mathrm{e}^{At} M_2 z_0$. Let $V_P(z_0) \leq 1$ and c be the maximal $V_P(y(t))$ in the time range $0 \leq t \leq \bar{\delta}$. Geometrically, this means that the initial state and reachable output set are bounded by ellipsoids. Without loss of generality, assume $V_P(z_0) = 1$. Then,

$$c = \max_{t \in [0; \bar{\delta}]} \max_{z_0 \text{ with } V_P(z_0) = 1} V_P(M_1 \mathrm{e}^{At} M_2 z_0) \tag{130}$$

$$\overset{(*)}{=} \max_{t \in [0; \bar{\delta}]} \max_{z_0 \neq 0} \frac{V_P(M_1 \mathrm{e}^{At} M_2 z_0)}{V_P(z_0)} = \max_{t \in [0; \bar{\delta}]} \|M_1 \mathrm{e}^{At} M_2\|_P^2. \tag{131}$$

The rewriting $(*)$ is possible because $\| \cdot \|_P$ is an induced norm; see (32) and the proof of Theorem 5.1.14.

The missing term $-I$ and the negative range of δ can be added by appropriate modifications. Hence, the problem is a special reachability problem. Nevertheless, the following analysis does not use existing reachability tools but uses an explicit bound to avoid further software dependencies.

Idea A series form of the matrix exponential is expanded up to order $r \geq 1$, and the remainder X is bounded:

$$e^{A\delta} - I = \sum_{i=0}^{\infty} \frac{A^i \delta^i}{i!} - I = \sum_{i=1}^{r} \frac{A^i \delta^i}{i!} + \underbrace{\sum_{i=r+1}^{\infty} \frac{A^i \delta^i}{i!}}_{X}, \tag{132}$$

where in the following $0^0 := 1$ to allow for $\delta = 0$. A similar approach appears in Taylor-series methods for reachability analysis (Chen et al., 2013) or for the stability analysis of real-time control systems (Hetel, Daafouz, et al., 2006).

Implementation Applying this idea leads to

$$\|M_1(e^{A\delta} - I)M_2\|_P \overset{(132)}{=} \left\| \sum_{i=1}^{r} \frac{M_1 A^i M_2 \delta^i}{i!} + M_1 \sum_{i=r+1}^{\infty} \frac{A^i \delta^i}{i!} M_2 \right\|_P \tag{133}$$

$$\overset{(29b),(29c),(31)}{\leq} \sum_{i=1}^{r} \frac{\|M_1 A^i M_2\|_P |\delta|^i}{i!} + \|M_1\|_P \sum_{i=r+1}^{\infty} \frac{\|A\|_P^i |\delta|^i}{i!} \|M_2\|_P =: h(|\delta|). \tag{134}$$

The function $h(|\delta|)$ is nondecreasing for increasing $|\delta|$ because it is a power series $\sum_{i=1}^{\infty} c_i |\delta|^i$ with nonnegative coefficients $c_i \geq 0$. Consequently, its bounds for $|\delta| \in [0; \bar{\delta}]$ are $h(0) = 0$ and $h(\bar{\delta})$, so

$$0 \leq \|M_1(e^{A\delta} - I)M_2\|_P \leq h(\bar{\delta}) \quad \forall \delta \in [-\bar{\delta}; \bar{\delta}]. \tag{135}$$

For computation, $h(\bar{\delta})$ with $\bar{\delta} \geq 0$ is rewritten as

$$h(\bar{\delta}) \overset{(134)}{=} \sum_{i=1}^{r} \bar{\delta}^i \underbrace{\frac{\|M_1 A^i M_2\|_P - \|M_1\|_P \|A\|_P^i \|M_2\|_P}{i!}}_{=:\gamma_i,\ i \geq 1} + \|M_1\|_P \underbrace{\sum_{i=0}^{\infty} \frac{\|A\|_P^i \bar{\delta}^i}{i!}}_{e^{\|A\|_P \bar{\delta}}} \|M_2\|_P - \|M_1\|_P \|M_2\|_P \tag{136}$$

$$= \sum_{i=1}^{r} \gamma_i \bar{\delta}^i - \|M_1\|_P \|M_2\|_P (e^{\|A\|_P \bar{\delta}} - 1). \tag{137}$$

To include the special case of $\bar{\delta} = 0$, define $0^0 := 1$. As $\lim_{\bar{\delta}\to 0^+} h(\bar{\delta}) = 0$, this bound preserves the property

$$\|\Delta A_{\dots}\|_P \to 0 \text{ for } \Delta t \to 0 \tag{138}$$

from Theorem 5.1.4, and therefore also the feasibility result from Theorem 5.1.9. In the implementation, $r = 10$ is used.

5.1.7 LMI-Based Synthesis of P

This section describes how P is determined to show stability.

5.1.7.1 Stability Test for Given P

First, consider how to check stability if P is given. In simplified notation, Theorem 5.1.8 shows stability if

$$\|A(\Delta t = 0)\|_P + \sum_{\dots} \|\Delta A_{\dots}(\Delta t)\|_P < 1 \quad \forall \Delta t \in \dots. \tag{139}$$

Checking this condition is implemented using upper bounds on each summand, here denoted $\|\cdot\|_P^{\mathrm{ub}}$: Stability holds if

$$\tilde{\rho} := \Bigg(\|A(\Delta t = 0)\|_P^{\mathrm{ub}} + \sum_{i=1}^{m} \|\Delta A_{\mathrm{u},i}\|_P^{\mathrm{ub}} + \sum_{j=1}^{p} \|\Delta A_{\mathrm{y},j}\|_P^{\mathrm{ub}} + \sum_{i=1}^{m}\sum_{j=1}^{p} \|\Delta A_{\mathrm{uy},i,j}\|_P^{\mathrm{ub}} \Bigg) < 1. \tag{140}$$

For all bounds therein, the impact of numerical error is neglected for now. As detailed in Appendix B.3.3, this is considered later by appropriately increasing the bounds. The bound $\|A(\Delta t = 0)\|_P^{\mathrm{ub}} :\geq \|A(\Delta t = 0)\|_P$ can be computed with negligible error by Theorem 5.1.15. The bounds on $\Delta A_{\dots}$, e.g.,

$$\|\Delta A_{\mathrm{u},i}\|_P^{\mathrm{ub}} :\geq \max_{\Delta t_{\mathrm{u}}^{(i)} \in \left[\underline{\Delta t}_{\mathrm{u}}^{(i)};\, \overline{\Delta t}_{\mathrm{u}}^{(i)}\right]} \|\Delta A_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)})\|_P, \tag{141}$$

are determined per Section 5.1.6.

Theorem 5.1.17 guarantees the existence of P with $\|A(\Delta t = 0)\|_P < 1$. If such P is chosen arbitrarily, the resulting bounds $\|\Delta A_{\dots}\|_P^{\text{ub}}$ are often too large to show stability by $\tilde{\rho} < 1$. Therefore, P must be optimized towards small $\tilde{\rho}$. For this, an LMI-based approach is presented in the following.

5.1.7.2 Introduction to Linear Matrix Inequalities

Linear matrix inequalities (LMIs) are a generalization of scalar linear inequalities, e.g., $3x + 5 < 0$, to positive definite matrices, e.g., $x_1 A + x_2 B \prec 0$ with given matrices A, B and unknown scalars x_1, x_2. LMIs are well-suited for numerical optimization and can encode a large variety of problems related to quadratic Lyapunov functions and matrix norms (Boyd et al., 1994).

Definition 5.1.20. An LMI optimization problem is defined as

$$\min_{x \in \mathbb{R}^a} f(x) \quad \text{subject to } M_1(x) \prec 0, \dots, M_b(x) \prec 0 \tag{142}$$

with the decision variable $x \in \mathbb{R}^a$, the linear objective function $f(x) = c^\top x$ with $c \in \mathbb{R}^a$ and the affine matrix-valued functions

$$M_i(x) = A_{i,0} + \sum_{j=1}^{a} x^{(j)} A_{i,j} \text{ with } A_{i,j} \in \mathbb{R}^{n \times n}. \tag{143}$$

◁

It is also possible to use symmetric matrices $X^\top = X$ as decision variables because they can be parameterized by the elements of their lower (or upper) triangle. In general, any affine function of scalar and matrix decision variables can be used as condition M_i. Taking the product of decision variables is not possible within the LMI framework, which will become relevant later.

For a symmetric matrix M, the eigenvalues are real (Bernstein, 2009, Proposition 5.5.20.iii). The minimum or maximum eigenvalue $\lambda_{\{\min,\max\}}$ of M can be formulated as LMI condition (Bernstein, 2009, Lemma 8.4.1) via

$$\lambda_{\min}(M) > c \Leftrightarrow M \succ cI, \tag{144}$$
$$\lambda_{\max}(M) < c \Leftrightarrow M \prec cI. \tag{145}$$

The same is possible for the singular values $\sigma_{\{\min,\max\}}(M) = \lambda^{1/2}_{\{\min,\max\}}(M^\top M)$:

$$\|M\|_2 \overset{\text{Definition 4.1.6}}{=} \sigma_{\max}(M) < c \quad \Leftrightarrow \quad M^\top M \prec c^2 I, \tag{146}$$
$$\sigma_{\min}(M) > c \quad \Leftrightarrow \quad M^\top M \succ c^2 I. \tag{147}$$

A similar result for the P-ellipsoidal norm can be derived from its definition and Definition 5.1.2 (positive definiteness): For $c \geq 0$,

$$\|M\|_P < c \overset{(107)}{\Leftrightarrow} \max_{x \in \mathbb{R}^n \setminus \{0\}} \sqrt{\frac{(Mx)^\top P (Mx)}{x^\top P x}} < c \tag{148}$$

$$\Leftrightarrow \sqrt{(Mx)^\top P (Mx)} < c\sqrt{x^\top P x} \tag{149}$$

$$\Leftrightarrow x^\top M^\top P M x < x^\top c^2 P x \tag{150}$$

$$\Leftrightarrow M^\top P M < c^2 P. \tag{151}$$

To simplify computations or increase numerical robustness, LMI conditions can be equivalently rewritten:

Lemma 5.1.21 (Rewriting of LMI). *Let $A, M \in \mathbb{R}^{n \times n}$ and M be invertible. Then,*

$$A < 0 \quad \Leftrightarrow \quad M^{-\top} A M^{-1} < 0. \tag{152}$$

Proof. Transform the quadratic form $x^\top A x$ to new coordinates $x =: M^{-1} z$:

$$A < 0 \Leftrightarrow x^\top A x < 0 \Leftrightarrow z^\top M^{-\top} A M^{-1} z < 0 \Leftrightarrow M^{-\top} A M^{-1} < 0. \tag{153}$$

□

5.1.7.3 LMI Problem Formulation

To use LMIs, the norm bounds in (140) are expressed using (151) as

$$A(0)^\top P\, A(0) < \bar{\rho}^2 P \quad (\Leftrightarrow \|A(0)\|_P < \bar{\rho}), \tag{154}$$

$$\Delta A_i^\top P\, \Delta A_i < \beta^2 P \quad (\Leftrightarrow \|\Delta A_i\|_P < \beta) \quad \forall \Delta A_i \in \Delta\mathcal{A}, \tag{155}$$

where $A(0) = A(\Delta t = 0)$ is the nominal-case dynamics and, for now, $\Delta\mathcal{A}$ the set of all $\Delta A_{\dots}$ in Theorem 5.1.4 for all possible Δt. As a first approximation, this leads to $\|A(0)\|_P^{\text{ub}} \approx \bar{\rho}$ and $\|\Delta A_{\dots}\|_P^{\text{ub}} \approx \beta$, so

$$\tilde{\rho} \approx \bar{\rho} + \sum_{\dots} \beta \tag{156}$$

can be optimized by

$$\min_{P>0,\, \bar{\rho}>0,\, \beta>0} \left(\bar{\rho} + \sum_{\dots} \beta \right) \quad \text{subject to (154) and (155).} \tag{157}$$

However, this is not an LMI but a *bilinear* matrix inequality because (154) contains a product of the optimization variables P and $\bar{\rho}$. In general, such bilinear problems are more difficult to solve and a conversion is possible only in special cases (Boyd et al., 1994; VanAntwerp and Braatz, 2000). The same problem occurs for β in (155). To obtain an LMI formulation, $\bar{\rho}$ and β are instead set to a fixed value and optimized separately.

Furthermore, a practical implementation must avoid singular or infinite P, which could occur as discussed in the proof of Remark 5.1.18. Therefore,

$$\gamma I \prec P \prec I \quad (\Leftrightarrow \lambda_{\min}(P) > \gamma \ \wedge \ \lambda_{\max}(P) < 1) \tag{158}$$

with $\gamma > 0$ is used as an additional constraint. Note that the upper bound was arbitrarily fixed as $\lambda_{\max}(P) < 1$ because scaling P does not affect $\|\cdot\|_P$.

Optimizing for numerical robustness results in the LMI optimization problem

$$\max_{P>0,\gamma>0} \gamma \quad \text{subject to (154), (155) and (158), with given } \bar{\rho}, \beta, \tag{159}$$

where the constants $\bar{\rho}$ and β are optimized in an outer loop. Numerical robustness is further improved by preconditioning, as detailed in Appendix B.3.2.

While in theory, $\Delta\mathcal{A}$ should be the set of all $\Delta A_{\{\mathrm{u,y,uy}\},\dots}$ for a representative set of timings, this would be prohibitively large for systems with many sensors and actuators. Instead, the set

$$\Delta\mathcal{A} \ = \ \left\{ A(\Delta t) - A(0) \ \middle| \ \Delta t \ \in \ \left(\{\underline{\Delta t}_{\mathrm{u}}, 0, \overline{\Delta t}_{\mathrm{u}}\} \times \{\underline{\Delta t}_{\mathrm{y}}, 0, \overline{\Delta t}_{\mathrm{y}}\} \right) \setminus \{0\} \right\} \tag{160}$$

is used that represents the total deviation $\sum \Delta A_{\dots}$ for eight extreme combinations of Δt_{u} and Δt_{y}. Due to the approximations used in deriving the LMI, the resulting P does not guarantee any norm bound. Instead, the resulting norm bound is computed in interval arithmetic, as discussed in Appendix B.3.3.

5.1.7.4 Choice of $\bar{\rho}$ and β

In the previous LMIs, the parameters $\bar{\rho}$ and β must be given, whereas the actual goal is to minimize the analysis result $\tilde{\rho}$. Mainly, $\bar{\rho}$ and β should be minimized because, by (156), neglecting the approximation of $\Delta\mathcal{A}$,

$$\tilde{\rho} \leq \bar{\rho} + \beta(m + p + mp). \tag{161}$$

On the other hand, experiments show that smaller $\bar{\rho}$ increases $\|\Delta A_i\|_P$, which may require larger β and hence increase $\tilde{\rho}$. To derive a compromise, the range of useful $\bar{\rho}$ is considered: Stability requires $\bar{\rho} < 1$. Because $\bar{\rho} > \|A(0)\|_P > \rho\{A(0)\}$, the range for $\bar{\rho}$ is $\rho\{A(0)\} < \bar{\rho} < 1$. As a compromise between large and small $\bar{\rho}$, the implementation uses a fixed value $\bar{\rho} = 0.8 + 0.2\rho\{A(0)\}$ in this range and a heuristic search to find the value of β that leads to the smallest overall norm bound $\tilde{\rho}$. This search procedure is detailed in Appendix B.3.1. In future work, it could be extended by a variation of $\bar{\rho}$.

5.1.8 Overall Algorithm

For given system parameters, the following computations are performed, using the results of Section 5.1.7:

1. Compute the preconditioning matrix R (Appendix B.3.2).
2. Repeat for multiple β with fixed $\bar{\rho}$, according to Section 5.1.7.4:
   - Optimize P by LMI methods using the given $\bar{\rho}$, β and preconditioning matrix R. (Section 5.1.7.3 and Appendix B.3.2)
   - Determine a fast numerical approximation $\tilde{\rho}_{\text{approx}}$ of the resulting $\tilde{\rho}$. For this, the computation (140) of $\tilde{\rho}$ is approximated in finite-precision arithmetic. The norm bounds $\|\Delta A_{\dots}\|_P^{\text{ub}}$ therein are not implemented using Section 5.1.6 but instead approximated by the maximum $\max_\delta \|\Delta A_{\dots}(\delta)\|_P$ over 100 samples of δ.
   - This approximation $\tilde{\rho}_{\text{approx}} \approx \tilde{\rho}$ is used to update $\bar{\rho}$ and β.
3. From the iterations, choose the P with best $\tilde{\rho}_{\text{approx}}$.
4. Compute the true bound $\tilde{\rho}$ using interval arithmetic and validated norm bounds (Appendix B.3.3 and Section 5.1.6).
5. Return P and the validated bound $\tilde{\rho}$ by (140).

5.1.9 Experiments

The approach was implemented in Python using *CVXPY* (Diamond and Boyd, 2016) and the *CVXOPT* solver (Andersen et al., 2020) for LMIs, together with *mpmath* (F. Johansson et al., 2018) for interval arithmetic. The used example systems are listed in Appendix A. Implementation details are given in Appendix B and source code is referenced in Appendix F.

Results Table 1 shows the results and computation times, both for the verified result computed in interval arithmetic ($\tilde{\rho}$, t) and for the fast numerical approximation ($\tilde{\rho}_{\text{approx}}$, t_{approx}), as detailed in Section 5.1.8.

First, consider the examples C2 and D2. Stability ($\tilde{\rho} < 1$) could be proven for these examples, for which no previous stability result is known and the reachability analysis of Section 4.2.10 did not succeed (column "Reach").

These examples are the one- (C2) and three-axis (D2) angular rate control of a linearized quadrotor helicopter with a period of $T = 10\,\text{ms}$ and a timing uncertainty of $\pm 1\,\%$. Example C2 is a SISO system with $m = p = 1$, $n_{\text{d}} = 2$ and $n_{\text{p}} = 1$, so that its total dimension is $n = 5$. Example D2 is a MIMO system with $m = 4$, $p = 3$ and a total dimension of $n = 16$.

Impact of Numerical Error While the approximation $\tilde{\rho}_{\text{approx}}$ is not guaranteed to be correct, it is significantly faster ($t_{\text{approx}} < t$). The small deviations $|\tilde{\rho}_{\text{approx}} - \tilde{\rho}|$ indicate that interval arithmetic can be omitted in a practical application. Note that the deviation was not evaluated for cases where stability could not be shown, because then there is no need for verification.

Impact of Timing The influence of timing was also considered separately: The norm-based analysis succeeds ($\tilde{\rho} < 1$) for example for A1 but fails for its variant A3 with increased timing uncertainty. The same occurs when the timing uncertainty of example D2 is doubled (D2_{b}). The results suggest that the conservatism of the norm-based analysis increases with timing uncertainty.

Impact of Dimension Reachability analysis succeeds only for the low-dimensional examples A1 and A3. To analyze the influence of dimension on the norm-based analysis, the dimension of D2 was doubled by block-diagonal repetition. By construction, the resulting system D2_{c} of dimension $n = 32$ has the same stability properties as D2. However, the analysis can only show stability for reduced timing uncertainty (D2_{d}, $\overline{\Delta t_{\text{y}}}$ reduced to 1/10th). This takes about five minutes for the approximation and ca. 37 minutes to verify. The conservatism relates to the fact that the summands of Theorem 5.1.4 are norm-bounded individually, while their total effect is generally less severe.

Table 1: Experimental results

System	n	$\tilde{\rho}_{\text{approx}}$	$\lvert\tilde{\rho} - \tilde{\rho}_{\text{approx}}\rvert$	t_{approx}	t	Reach
A1	4	0.915	$9.2 \cdot 10^{-8}$	1.3	1.7	✔
A3	4	1.757	—	1.3	—	✔
C2	5	0.914	$9.2 \cdot 10^{-8}$	1.0	1.6	✗
D2	16	0.926	$9.3 \cdot 10^{-8}$	17.5	98.1	✗
D2_b: $2\Delta t$	16	1.073	—	12.8	—	✗
D2_c: $2n$	32	1.021	—	312.1	—	✗
D2_d: $2n$, $\frac{\overline{\Delta t_\text{y}}}{10}$	32	0.979	$9.8 \cdot 10^{-8}$	308.1	2196.3	✗

n	Total state dimension $n = n_\text{p} + n_\text{d} + m + p$
$\tilde{\rho}$	Upper bound on stability factor with interval arithmetic
$\tilde{\rho}_{\text{approx}}$	Fast approximation of $\tilde{\rho}$ (cf. Section 5.1.8)
t_{approx}, t	Time for computing $\tilde{\rho}_{\text{approx}}$, $\tilde{\rho}$
Reach ✔	Boundedness ($\tilde{\rho} \leq 1$) could be shown by reachability analysis (Section 4.2.10)

Modified system parameters are indicated as $2n$ (dimension doubled by repetition) and $K\Delta t$ (timing variable(s) increased by factor K).
All numbers are rounded up to the last shown digit. Times are wall-times in seconds, using an Intel i7-8750H CPU and 16GB RAM.

Conclusion In summary, the approach was successfully applied to systems with sufficiently small timing deviations and a dimension up to $n = 32$. In comparison with the direct reachability analysis of Section 4.2.10, it can be seen that the LIS approach is advantageous for systems with higher dimension (e.g., C2, D2), while reachability analysis may be better in certain cases with higher timing uncertainty and low dimension (A3).

5.1.10 Discussion

This section presented an approach to the stability verification of linear MIMO control systems with uncertain IO timing for each sensor and actuator. The major challenge is that the system dynamics depends on the combination of all individual timing variables, that is, varying jitter for each sensor and actuator. For existing methods from the literature, this combination leads to a curse of dimensionality. The presented method avoids this issue by exploiting the problem structure of the MIMO case: A decomposition of the discrete-time dynamics leads to summands with at most two timing

variables. Subsequently, these summands are bounded in terms of a norm that corresponds to a common quadratic Lyapunov function. The experimental results show that the approach is feasible for moderately complex systems with small timing uncertainty.

The results highlight two possible areas of improvement: First, the scalability can be improved by a more efficient implementation, considering that the interval computations are currently much slower than regular finite-precision arithmetic. The major factor is the mpmath library, which uses a software implementation of floating-point arithmetic instead of using the hardware floating-point unit. Second, there is relevant pessimism especially for higher dimensions. This could be tackled by an extension from the P-ellipsoidal norm to more general norms, corresponding to a change from quadratic Lyapunov functions to higher-order or piecewise quadratic variants. Finally, a fundamental limitation is the assumption of linearity: It is not clear if and how the decomposition can be extended to nonlinear systems. An alternative approach supporting nonlinearity will be discussed in the next section.

5.2 Continuization of Hybrid Automata

The major difficulty in analyzing real-time control systems is the mixture of discrete-time and continuous-time dynamics. Nevertheless, many such systems behave similar to a purely continuous-time system disturbed by a small discretization error. This approximation leads to the analysis technique of *continuization*, which is explored in this section.

Prior Publications A previous version of this section was published as Gaukler, 2020a. A feasibility study of an early, less strictly formalized approach to first-order continuization was implemented in the Master's thesis of Wu, 2019 under my supervision.

Introduction The motivation for continuization is the inability of other methods to tackle the actual hybrid system: The LIS-based approach of the previous section works but requires a discretization that is not possible for nonlinear systems. In theory, reachability analysis would support nonlinear systems, but the experiment in Section 4.2.10 showed that a standard reachability tool is unsuitable even for many simple real-time control systems. As discussed before, the reason is that frequent discrete transitions lead to large pessimism (Bak and Johnson, 2015; el Hakim and Bekooij, 2018).

One solution is to tailor reachability analysis to the specific structure of real-time control systems. As noted earlier, transitions with exactly known time can be computed without overapproximation (Gruber and Althoff, 2021), which is however only applicable to ideal timing. For the case of uncertain sampling periods, the timing scheme can be explicitly incorporated into the analysis algorithm (Al Khatib et al., 2016). More generally, the evolution of clock, continuous and discrete states can be considered separately to reduce pessimism (el Hakim and Bekooij, 2018). While such specialized approaches can provide good results for a narrow class of systems, they have the drawback that one can no longer draw from the decades of experience condensed in existing reachability analysis tools. Therefore, for example, an extension from linear to nonlinear systems is no longer as easy as enabling a configuration option in an existing tool but may require years of research and implementation.

An alternative approach that avoids this problem is *continuization* (Althoff, Yaldiz, et al., 2011; Bak and Johnson, 2015): If the mixed discrete-time and continuous-time original system behaves "smoothly enough", it can be approximated by a continuous-time system with "smooth" behavior that can be analyzed efficiently with existing tools. Formally, the discontinuous solution of the original hybrid system is approximated by the continuous solution of a differential inclusion, and the approximation error is bounded.

Contents This section extends the work on continuization by Bak and Johnson, 2015 towards new classes of real-time control systems. The contents are structured as follows: First, the problem statement and notation are introduced. Then, Section 5.2.3 develops a specialized formal framework for proving continuization using abstractions of hybrid automata. Section 5.2.4 restates prior work by Bak and Johnson, here termed *zero-order continuization*, in the new formal framework, uncovering a previously unknown limitation: The method is limited to state feedback, e.g., proportional controllers. Hence, it can not handle controllers with internal dynamics, e.g., integral controllers, for which a complementary variant named *first-order continuization* is developed in Section 5.2.5. At first, only the case of ideal timing is considered. Experiments in Section 5.2.6 show that first-order continuization is feasible with some limitations. Next, the case of uncertain timing is discussed in Section 5.2.7. Then, Section 5.2.8 summarizes the results, discusses possible extensions and points out connections to other analysis methods.

5.2.1 Problem Statement

Given A nonlinear real-time control system H as in Section 3.2 is given.

Goal Determine a bound on the reachable set $\mathrm{Reach}_x(H, t)$. The resulting procedure can then be used for the design-time analysis of safety requirements as defined in Section 3.2.1.

Simplifying Assumptions To avoid excessive complexity, the discussion is at first limited to ideal timing, analogous to the model of Section 4.2.8. Later, a generalization to uncertain timing and skipped events is sketched.

5.2.2 Notation

Important notation for the discussion of continuization is summarized in the following. A full list can be found on page xiv.

Time arguments can be omitted if their value is t, i.e., $x(t)$ may be abbreviated as x, but $x(kT)$ is never abbreviated. For a state x, the time derivative is denoted $\dot{x}$, and the successor state of a discrete transition is denoted x'.

The closed ball of radius r and appropriate dimension n is

$$\mathbb{B}_r \coloneqq \{x \in \mathbb{R}^n \mid |x| \leq r\}. \tag{162}$$

For $c \in \mathbb{R}$ and appropriate sets $\mathcal{A}, \mathcal{B}$, the Minkowski sum $\oplus$, set-valued product $\otimes$ and scaling $c\mathcal{A}$ are defined as

$$A \oplus B \coloneqq \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\}, \quad \mathcal{A} \otimes \mathcal{B} \coloneqq \{ab \mid a \in \mathcal{A}, b \in \mathcal{B}\} \tag{163}$$

$$\text{and} \quad c\mathcal{A} \coloneqq \{ca \mid a \in \mathcal{A}\}. \tag{164}$$

The set-valued product should not be confused with the Cartesian product $\times$. The smallest closed multidimensional interval (box) containing the set S is denoted $\square S$.

For a function $f(x)$ with non-scalar argument or result, the partial derivative $\partial f(x)/\partial x$ is generalized as the Jacobian matrix

$$\frac{\partial \left[f^{(1)}(x^{(1)}, x^{(2)}, \dots) \;\; f^{(2)}(x^{(1)}, x^{(2)}, \dots) \;\; \dots\right]^\mathsf{T}}{\partial \left[x^{(1)} \;\; x^{(2)} \;\; \dots\right]^\mathsf{T}} \coloneqq \begin{bmatrix} \frac{\partial f^{(1)}}{\partial x^{(1)}} & \frac{\partial f^{(1)}}{\partial x^{(2)}} & \dots \\ \frac{\partial f^{(2)}}{\partial x^{(1)}} & \frac{\partial f^{(2)}}{\partial x^{(2)}} & \dots \\ \dots & & \end{bmatrix}. \tag{165}$$

5.2.3 Formal Basis

The point of continuization is to avoid the expensive analysis of the original automaton and instead only consider a simplified variant. However, as will be seen later, this simplification requires a bound on the reachable states, which at first leads to a chicken-and-egg problem: For simple analysis, a bound is required, which in turn requires the analysis result. To break this circle, Bak and Johnson, 2015 use a scheme that can be summarized as follows.

1. A complicated automaton H_1 is abstracted, i.e., simplified, under the assumption that its state remains within a certain set $\mathcal{X}$. This assumption can be abbreviated as $\text{Reach}_x(H_1) < \mathcal{X}$. Here, "$<$" denotes an appropriate condition for "smaller than", possibly stricter than $\subseteq$.
2. For the simplified automaton H_2, an outer bound on the reachable set $\text{Reach}_x(H_2)$ is determined.
3. The assumption is validated only by checking if $\text{Reach}_x(H_2) < \mathcal{X}$, without knowledge of the original reachable set $\text{Reach}_x(H_1)$. If the condition is true, then H_2 is an abstraction of H_1 without any restrictions:

   If $\text{Reach}_x(H_2) < \mathcal{X}$, then $\text{Reach}_x(H_1) < \text{Reach}_x(H_2) < \mathcal{X}$.

While this may seem trivially true at first sight, a careful choice of the details is required to derive a valid theorem. For example, choosing "$<$" as $\subseteq$ leads to the following false claim: 1. Assuming $\text{Reach}_x(H_1) \subseteq \mathcal{X}$, derive an abstraction H_2. 2. Determine a bound on $\text{Reach}_x(H_2)$. 3. Claim: If $\text{Reach}_x(H_2) \subseteq \mathcal{X}$, then $\text{Reach}_x(H_1) \subseteq \text{Reach}_x(H_2) \subseteq \mathcal{X}$.

This is false: 1. Under the assumption $\text{Reach}_x(H_1) \subseteq \mathcal{X}$, a trivial abstraction of any H_1 is $H_2 =$ "$x(t) \in \mathcal{X}$". 2. This yields $\text{Reach}_x(H_2) = \mathcal{X}$. 3. As then $\text{Reach}_x(H_2) \subseteq \mathcal{X}$, the above claim would state that $\text{Reach}_x(H_1) \subseteq \mathcal{X}$. This result is clearly wrong, as one could show any bound $\mathcal{X}$ for any automaton H_1. A correct version is given in the following:

Theorem 5.2.1 (Abstraction of HA Within Assumed Bound)**.** *Let H_1, H_2 be given from Figure 30, $\mathcal{X} \subseteq \mathbb{R}^n$ be a set and $\epsilon > 0$ such that*

$$\mathcal{F}_1(x) \subseteq \mathcal{F}_2(x) \quad \forall x \in \mathcal{X} \oplus \mathbb{B}_\epsilon \quad \textit{and} \tag{166a}$$

$$\text{Reach}_x(H_2) \subseteq \mathcal{X}. \tag{166b}$$

Then, $H_1 \subseteq_x H_2$, so $\text{Reach}_x(H_1) \subseteq \text{Reach}_x(H_2)$.

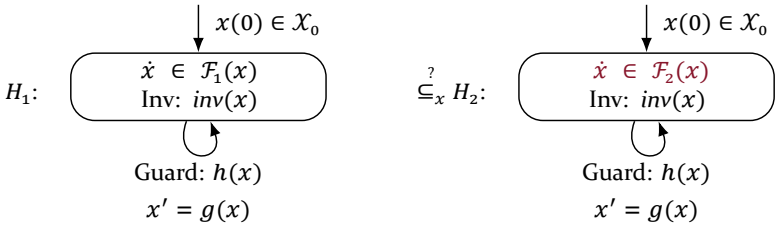


Figure 30: Automata of Theorem 5.2.1

Proof Sketch. The proof uses the continuity of continuous evolution. In graphical terms, x in H_1 can not escape $\mathcal{X}$ without passing through the "buffer zone" of width ϵ around $\mathcal{X}$. As the abstraction is still valid there, this escape would be caught in $\text{Reach}_x(H_2)$. The full proof, a counterexample for $\epsilon = 0$ and further details are given in Appendix C.1.1. □

Usage Theorem 5.2.1 can be used as follows: Given H_1, choose a set (e.g., interval) $\mathcal{X}$, preferably as a rough guess of $\text{Reach}_x(H_1)$ from numerical simulations. Apply the broadening from $\mathcal{F}_1(x)$ to $\mathcal{F}_2(x)$. For example,

$$\mathcal{F}_1(x) = \{-x + \sin(x)\} \tag{167}$$

$$\rightsquigarrow \quad \mathcal{F}_2(x) = \{-x\} \oplus \{\sin(\xi) \mid x_{\min} - \epsilon \leq \xi \leq x_{\max} + \epsilon\} \tag{168}$$

is a valid broadening for $\mathcal{X} = [x_{\min}; x_{\max}]$ and $\epsilon > 0$. Analyze the resulting simplified automaton H_2 to check whether $\text{Reach}_x(H_2) \subseteq \mathcal{X}$. If this condition is true, then $\text{Reach}_x(H_1) \subseteq \text{Reach}_x(H_2) \subseteq \mathcal{X}$. Else, one can try again with $\mathcal{X}$ changed to an enlarged version of $\text{Reach}_x(H_2)$, such as $\mathcal{X} = \text{Reach}_x(H_2) \oplus \mathbb{B}_r$ with some bloating radius $r \geq 0$.

Interval Bound for Continuous Evolution To bound the continuous evolution within finite time, the following lemma is useful:

Lemma 5.2.2. *Let $x \in \mathbb{R}^n$ be the state vector of the differential equation*

$$\dot{x}(t) = f(t), \quad x(t_0) = x_0 \tag{169}$$

whose input is bounded by $f(t) \in \mathcal{F}$ for all $t \in [t_0; t_0 + T]$. Then, x can be bounded by

$$x(t) - x_0 \in [0; T] \otimes \square\mathcal{F} \quad \text{for all } t \in [t_0; t_0 + T]. \tag{170}$$

Proof. This is shown in Appendix C.2 based on Frehse, 2015, Lemma 1. □

Remark 5.2.3. Taking the interval hull $\square\mathcal{F}$ can not be omitted. For a simplified example, consider

$$\mathcal{F} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}, \quad f(t) = \begin{cases} \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right], & 0 \le t < \frac{T}{2}, \\ \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right], & \frac{T}{2} \le t \le T \end{cases} \tag{171}$$

with $x_0 = 0$, $t_0 = 0$, $T = 1$. The discontinuity in $f(t)$ at $t = T/2$ can be ignored for the discussion here because it could equivalently be replaced by a smooth transition of negligibly short duration. The solution

$$x(T) = \int_0^T f(\tau)\,\mathrm{d}\tau = \frac{T}{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{T}{2}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{172}$$

is not contained in (170) if the interval hull $\square\mathcal{F}$ is replaced by $\mathcal{F}$ itself:

$$[0;T] \otimes \mathcal{F} = \left\{ \begin{bmatrix} t \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ t \end{bmatrix} \;\middle|\; 0 \le t \le 1 \right\}. \tag{173}$$

◁

5.2.4 Zero-Order Continuization

The idea of continuization is to abstract a system that admits mixed discrete and continuous behavior by a purely continuous approximation plus bounded error (Althoff, Yaldiz, et al., 2011; Bak and Johnson, 2015). For a controller that is designed in continuous time and then discretized for implementation, continuization approximately inverts this discretization.

A first variant of continuization given by Bak and Johnson, 2015 applies to a control loop with the physical plant

$$\dot{x}_\mathrm{p}(t) = f_\mathrm{p}(x_\mathrm{p}(t), x_\mathrm{d}(t)) \tag{174a}$$

and a *zero-order-hold* state feedback controller

$$x_\mathrm{d}(t) = f_\mathrm{d}(x_\mathrm{p}(kT)), \quad kT \le t < (k+1)T, \quad k \ge 0 \tag{174b}$$

with fixed period T. The hybrid automaton H_0 corresponding to this system model is shown in Figure 31 (top left).

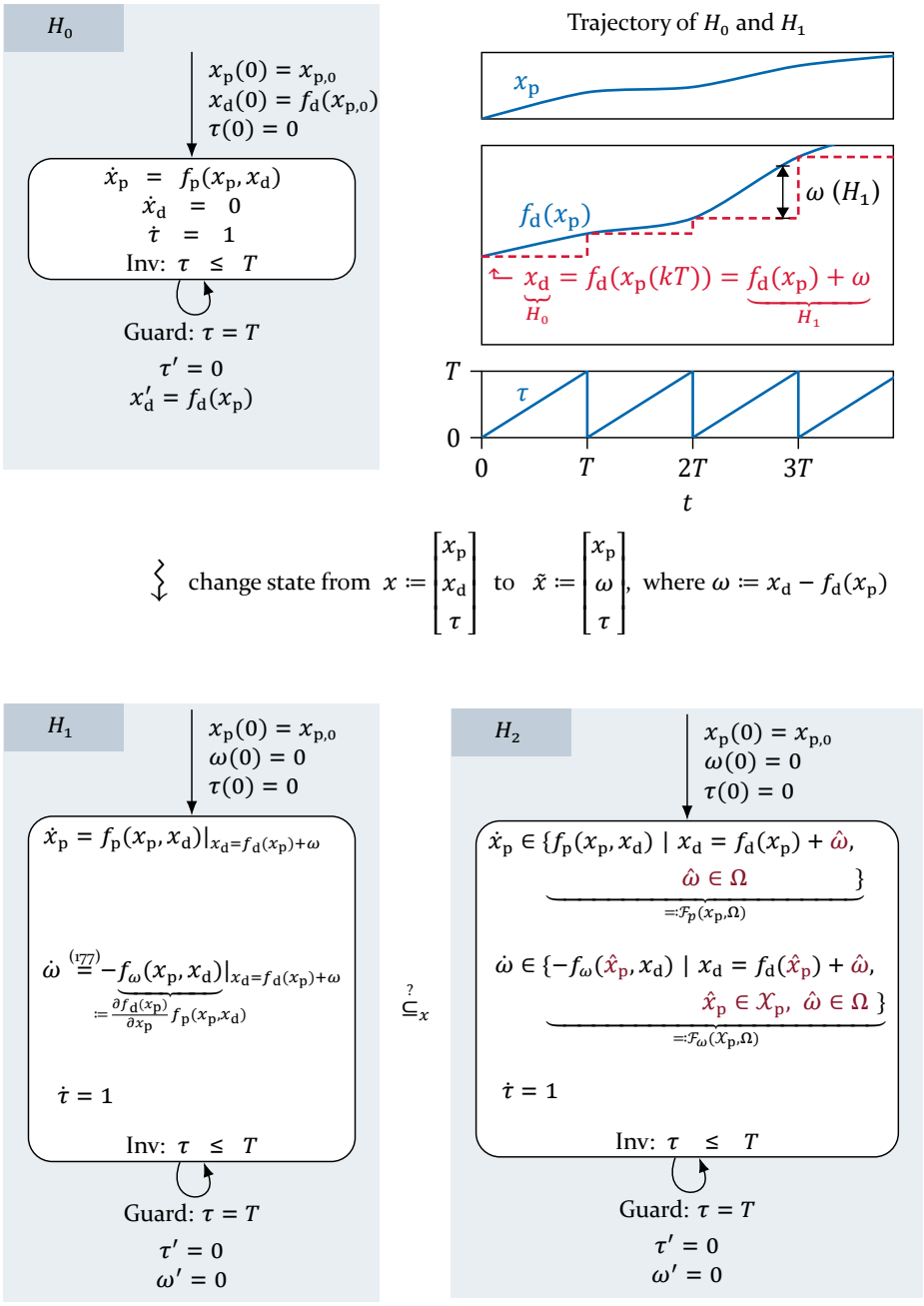


Figure 31: Hybrid automata of control loop and zero-order continuization in Theorem 5.2.4

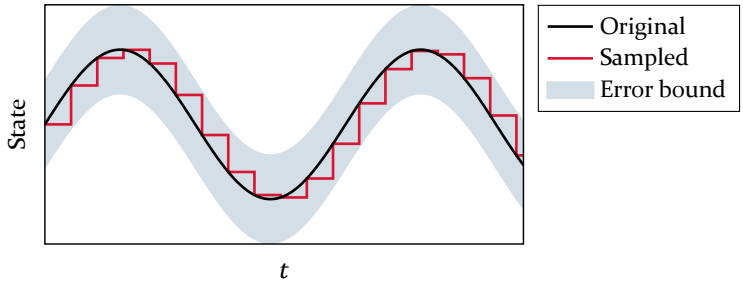


Figure 32: Key idea of zero-order continuization: A sample-and-hold signal can be approximated by its continuous original plus a bounded error.

Compared to the system model considered in this thesis (Section 3.1), the new model lacks a number of features:

- IO timing uncertainty, which is partly present in Bak and Johnson, 2015 but is omitted here for clarity.
- Disturbance is omitted for clarity but can be added trivially.
- The new model does not consider a separate input $u \neq x_\mathrm{d}$ or output $y \neq x_\mathrm{p}$. Without timing uncertainty, these are not required, as the functions g_p and g_d can be merged into f_d and f_p.
- There is no one-step delay from the measurement y_k to the output $u_{k+1} = g_\mathrm{d}(x_{k+1}) = g_\mathrm{d}(f_\mathrm{d}(x_{\mathrm{d},k}, y_k))$. Instead, the measurement is immediately used for the new actuator value.
- There are no internal controller dynamics, i.e., $x_{\mathrm{d},k+1}$ can not depend on $x_{\mathrm{d},k}$. The contrary was claimed in Bak and Johnson, 2015 but disproven in Gaukler, 2020a, Appendix B. However, many practical controllers have internal dynamics, e.g., integrators or state observers.

These differences are set aside for now and reconsidered later.

As illustrated in Figure 32, the key idea of zero-order continuization is that zero-order sample-and-hold with high sampling rate approximates a direct connection plus some small error. Here,

$$x_\mathrm{d}(t) = f_\mathrm{d}(x_\mathrm{p}(kT)) \approx f_\mathrm{d}(x_\mathrm{p}(t)). \tag{175}$$

The approximation error $\omega \coloneqq x_\mathrm{d} - f_\mathrm{d}(x_\mathrm{p})$ can be bounded by, loosely speaking, the product of the period T and the maximum absolute time derivative of $f_\mathrm{d}(x_\mathrm{p}(t))$. For the following explanation, assume that a bound $\omega \in \Omega$ is known. Then, substituting $x_\mathrm{d} = f_\mathrm{d}(x_\mathrm{p}) + \omega$ and $\omega \in \Omega$ in (174a) yields a purely continuous-time system

$$\dot{x}_\mathrm{p}(t) \in \{f_\mathrm{p}(x_\mathrm{p}(t), f_\mathrm{d}(x_\mathrm{p}(t)) + \omega) \mid \omega \in \Omega\} \tag{176}$$

for x_p with bounded disturbance, hence the name *continuization*. Typically, analysis of this new system is easy compared to the original automaton with its mixture of discrete and continuous-time dynamics. With minor modifications, the approach extends to more realistic timing schemes involving uncertain periods or skipped executions (Bak and Johnson, 2015), where analyzing the original automaton would be even more difficult. However, to avoid excessive complexity, this thesis will only consider the periodic case. A second simplifying assumption in the following derivations, however without loss of generality, is that the initial state is exactly known and there is no disturbance, so that the behavior of the control loop is uniquely determined.

Formal Derivation The above idea is now implemented using the formalism of abstractions outlined in Section 5.2.3. Consider the controller given by H_0 in Figure 31 (top left). A state transformation leads to an equivalent automaton H_1: The state is changed from $x = \begin{bmatrix} x_\mathrm{p}^\top & x_\mathrm{d}^\top & \tau \end{bmatrix}^\top$ to $\tilde{x} = \begin{bmatrix} x_\mathrm{p}^\top & \omega^\top & \tau \end{bmatrix}^\top$, which uses the error $\omega \coloneqq x_\mathrm{d} - f_\mathrm{d}(x_\mathrm{p})$ instead of the sampled state $x_\mathrm{d} = f_\mathrm{d}(x_\mathrm{p}(kT))$. The resulting trajectory is shown in Figure 31 (top right). Assuming that the derivative $\partial f_\mathrm{d}(x_\mathrm{p})/\partial x_\mathrm{p}$ exists, the dynamics of ω are

$$\dot{\omega} = \dot{x}_\mathrm{d} - \frac{\partial f_\mathrm{d}(x_\mathrm{p})}{\partial x_\mathrm{p}} \dot{x}_\mathrm{p} = \underbrace{0 - \frac{\partial f_\mathrm{d}(x_\mathrm{p})}{\partial x_\mathrm{p}} f_\mathrm{p}(x_\mathrm{p}, x_\mathrm{d})}_{\coloneqq -f_\omega(x_\mathrm{p}, x_\mathrm{d})} = -f_\omega(x_\mathrm{p}, f_\mathrm{d}(x_\mathrm{p}) + \omega), \tag{177}$$

except for the discrete transitions at $t = kT$. The negative sign of f_ω is chosen to simplify a comparison with the results of Bak and Johnson, 2015. The remaining dynamics and transitions follow immediately from the state transformation. This results in the automaton H_1 (Figure 31, bottom left). By construction, $H_0 =_x H_1$ with

$$x = q(\tilde{x}) \coloneqq \begin{bmatrix} x_\mathrm{p}^\top & (f_\mathrm{d}(x_\mathrm{p}) + \omega)^\top & \tau \end{bmatrix}^\top. \tag{178}$$

Next, Theorem 5.2.1 is applied to H_1, which has

$$\tilde{\mathcal{X}}_0 = \{\begin{bmatrix} x_\mathrm{p}(0)^\top & 0^\top & 0 \end{bmatrix}^\top\}, \tag{179a}$$

$$\mathcal{F}_1(\tilde{x}) = \left\{ \begin{bmatrix} f_\mathrm{p}(x_\mathrm{p}, f_\mathrm{d}(x_\mathrm{p}) + \omega) \\ -f_\omega(x_\mathrm{p}, f_\mathrm{d}(x_\mathrm{p}) + \omega) \\ 1 \end{bmatrix} \right\}, \tag{179b}$$

$$h(\tilde{x}) = (\tau = T), \tag{179c}$$

$$g(\tilde{x}) = \begin{bmatrix} x_\mathrm{p}^\top & 0^\top & 0 \end{bmatrix}^\top, \tag{179d}$$

$$inv(\tilde{x}) = (\tau \leq T). \tag{179e}$$

Choose the sets $\underline{\mathcal{X}}_\mathrm{p}$ and $\underline{\Omega}$ as a (possibly wrong) guess for the bounds of x_p and ω. Define the bloated bounds

$$\mathcal{X}_\mathrm{p} = \underline{\mathcal{X}}_\mathrm{p} \oplus \mathbb{B}_\epsilon, \tag{180}$$

$$\Omega = \underline{\Omega} \oplus \mathbb{B}_\epsilon \tag{181}$$

with arbitrary, small $\epsilon > 0$, and the combined, non-bloated bound

$$\tilde{\mathcal{X}} = \underline{\mathcal{X}}_\mathrm{p} \times \underline{\Omega} \times \mathbb{R}. \tag{182}$$

Now, to get from H_1 to H_2 (Figure 31, bottom right), the dependency of $\dot{x}_\mathrm{p}$ on ω is replaced with the bound $\omega \in \Omega$:

$$\dot{x}_\mathrm{p} \in \left\{ f_\mathrm{p}(x_\mathrm{p}, f_\mathrm{d}(x_\mathrm{p}) + \hat{\omega}) \;\middle|\; \hat{\omega} \in \Omega \right\} =: \mathcal{F}_\mathrm{p}(x_\mathrm{p}, \Omega). \tag{183}$$

Also, all dependencies of $\dot{\omega}$ are also replaced with their bounds:

$$\dot{\omega} \in \left\{ -f_\omega(\hat{x}_\mathrm{p}, f_\mathrm{d}(\hat{x}_\mathrm{p}) + \hat{\omega}) \;\middle|\; \hat{\omega} \in \Omega, \hat{x}_\mathrm{p} \in \mathcal{X}_\mathrm{p} \right\} =: \mathcal{F}_\omega(\mathcal{X}_\mathrm{p}, \Omega). \tag{184}$$

Thereby, $\mathcal{F}_1$ is broadened to

$$\mathcal{F}_2(\tilde{x}) = \mathcal{F}_\mathrm{p}(x_\mathrm{p}, \Omega) \times \mathcal{F}_\omega(\mathcal{X}_\mathrm{p}, \Omega) \times \{1\}. \tag{185}$$

Here, it would also be valid to use an outer approximation of $\mathcal{F}_2$, e.g., as interval. By construction,

$$\mathcal{F}_1(\tilde{x}) \subseteq \mathcal{F}_2(\tilde{x}) \quad \forall \tilde{x} \in \tilde{\mathcal{X}} \oplus \mathbb{B}_\epsilon, \tag{186}$$

so the only missing condition for Theorem 5.2.1 is $\mathrm{Reach}_{\tilde{x}}(H_2) \subseteq \tilde{\mathcal{X}}$. The automaton H_2 (Figure 31, bottom right) consists of three mostly separate subsystems, so its reachable set can be computed as follows:

- The state $x_\mathrm{p}(t)$ is independent of τ and ω and follows the continuous-time differential inclusion

$$\dot{x}_\mathrm{p} \in \mathcal{F}_\mathrm{p}(x_\mathrm{p}, \Omega), \quad x_\mathrm{p}(0) = x_{\mathrm{p},0}. \tag{187}$$

  Therefore it is within

$$\mathcal{X}'_\mathrm{p} \coloneqq \mathrm{Reach}_{x_\mathrm{p}} \begin{pmatrix} \dot{x}_\mathrm{p} \in \mathcal{F}_\mathrm{p}(x_\mathrm{p}, \Omega) \\ x_\mathrm{p}(0) = x_{\mathrm{p},0} \end{pmatrix}. \tag{188}$$

- Every T seconds, $\tau(t) = t \bmod T$ is reset to zero. Therefore, $\tau \in [0; T]$.
- At the same reset, $\omega(t)$ is set to zero. Between these resets it grows by

$$\dot{\omega} \in \mathcal{F}_\omega(\mathcal{X}_\mathrm{p}, \Omega), \tag{189}$$

  so, for $kT < t < (k+1)T$,

$$\omega(t) - \underbrace{\omega(kT)}_{0 \text{ (reset)}} \overset{\text{Lemma 5.2.2}}{\in} [0; T] \otimes \Box\mathcal{F}_\omega(\mathcal{X}_\mathrm{p}, \Omega) =: \Omega'. \tag{190}$$

Hence, $\mathrm{Reach}_{\tilde{x}}(H_2) \subseteq \mathcal{X}'_\mathrm{p} \times \Omega' \times [0; T]$. If $\mathcal{X}'_\mathrm{p} \subseteq \underline{\mathcal{X}}_\mathrm{p}$ and $\Omega' \subseteq \underline{\Omega}$, then, by $[0; T] \subseteq \mathbb{R}$, the condition $\mathrm{Reach}_{\tilde{x}}(H_2) \subseteq \tilde{\mathcal{X}}$ of Theorem 5.2.1 is fulfilled and therefore, $H_1 \subseteq_{\tilde{x}} H_2$. As there is a mapping $x = q(\tilde{x})$, also $H_1 \subseteq_x H_2$.

The result is summarized by the following theorem:

Theorem 5.2.4 (Zero-Order Continuization)**.** *From Figure 31, consider the control loop H_0 with state vector x and its modifications H_1 and H_2 with state vector $\tilde{x}$. Let f_d be a differentiable function. Then, $H_0 =_x H_1$ with*

$$x = q(\tilde{x}) \coloneqq \left[x_\mathrm{p}^\top \;\; (f_\mathrm{d}(x_\mathrm{p}) + \omega)^\top \;\; \tau\right]^\top. \tag{191}$$

Also, $H_1 \subseteq_x H_2$ if $\mathcal{X}'_\mathrm{p} \subseteq \underline{\mathcal{X}}_\mathrm{p}$ and $\Omega' \subseteq \underline{\Omega}$. Herein, $\underline{\mathcal{X}}_\mathrm{p}$ and $\underline{\Omega}$ are arbitrarily chosen and represent a (possibly wrong) guess for the set of x_p and ω. $\mathcal{X}'_\mathrm{p}$ and Ω' are determined by (188) *and* (190)*, and all remaining variables are as given in the above derivation.*

Proof. See the above derivation. □

Remark 5.2.5 (Iterative Analysis). If the condition does not hold, one can retry with an enlarged version of the result as new assumption, e.g., enlarging $\underline{\mathcal{X}}_\mathrm{p}$ to $\mathcal{X}'_\mathrm{p} \oplus \mathbb{B}_r$ and $\underline{\Omega}$ to $\Omega' \oplus \mathbb{B}_r$ with an arbitrary bloating radius $r \geq 0$. ◁

Remark 5.2.6 (Comparison with Prior Work). Theorem 5.2.4 mostly but not exactly matches the results of Bak and Johnson, 2015. In particular, they also permit that f_d depends on x_d. However, these differences can be judged as errors, as shown by the counterexamples in Gaukler, 2020a, Appendix B. ◁

Remark 5.2.7. As $\mathcal{X}'_\mathrm{p}$ can be determined without knowing $\underline{\mathcal{X}}_\mathrm{p}$, it is possible to avoid guessing $\underline{\mathcal{X}}_\mathrm{p}$: First, choose $\underline{\Omega}$ and compute $\mathcal{X}'_\mathrm{p}$. Then, use $\underline{\mathcal{X}}_\mathrm{p} = \mathcal{X}'_\mathrm{p}$ to finally determine Ω'. ◁

In summary, the result of continuization is an abstraction of the original control loop and a bound on its states:

Lemma 5.2.8. *If the conditions of Theorem 5.2.4 hold, the following statements are true for the original system H_0: H_2 is an abstraction of H_0, so*

- (187) *is an abstraction of $x_\mathrm{p}(t)$.*
- *$x_\mathrm{p}(t) \in \mathcal{X}'_\mathrm{p}$ for all t.*
- *$\omega(t) \in \Omega'$ and therefore $x_\mathrm{d}(t) = f_\mathrm{d}(x_\mathrm{p}(t)) + \omega(t) \in \{f_\mathrm{d}(x_\mathrm{p}(t))\} \oplus \Omega'$ for all t.*

Due to the inherent simplification, one can expect that the procedure leads to pessimism and can not show stability in all cases.

Conjecture 5.2.9. *There exist stable real-time control systems (of the form H_0 per Figure 31) whose stability can never be shown by Theorem 5.2.4. This particularly affects real-time control systems whose discrete-time execution differs significantly from quasi-continuous, i.e., arbitrarily fast, execution.*

5.2.5 First-Order Continuization

Motivation The method of zero-order continuization introduced in the previous section is appropriate for static state-feedback controllers, i.e., if the control signal u is determined only from the plant state x_p. Zero-order continuization approximates arbitrarily fast operation of the controller, i.e.,

$T \to 0$ with unchanged f_d. However, this approximation makes no sense if the controller has internal dynamics, e.g., a state observer or an integral part

$$x_{\mathrm{d},k+1} = \underbrace{x_{\mathrm{p},k} + x_{\mathrm{d},k}}_{f_\mathrm{d}(x_{\mathrm{p},k}, x_{\mathrm{d},k})} \approx \frac{1}{T} \int x_\mathrm{p} \,\mathrm{d}t, \tag{192a}$$

$$u_k = K x_{\mathrm{d},k}. \tag{192b}$$

Here, $T \to 0$ leads to an infinite speed of integration, so it is not possible to use zero-order continuization.

System Model Controller dynamics are added to the control loop (174):

$$\dot{x}_\mathrm{p}(t) = f_\mathrm{p}(x_\mathrm{p}(t), x_\mathrm{d}(t)) \tag{193a}$$

$$x_\mathrm{d}(t) = x_{\mathrm{d},k} = f_\mathrm{d}(x_{\mathrm{p},k}, x_{\mathrm{d},k-1}), \quad kT \le t < (k+1)T, \quad k \ge 0, \tag{193b}$$

where in this section the index k denotes the value at $t = kT$. As discussed in the previous section, the system model here deviates from that of Section 3.1 used elsewhere in this thesis. Particularly, there is no one-step delay from the measurement $x_\mathrm{p}(kT)$ to the actuation $x_\mathrm{d}(kT)$. Accordingly, the initial state is parameterized by $x_\mathrm{p}(0) = x_{\mathrm{p},0}$ and $x_{\mathrm{d},-1}$ (note the index -1).

Idea To enable continuization for dynamic controllers, it is extended to the *first-order*-hold approximation sketched in Figure 33: Following the idea of Section 3.1.4, a linear interpolation $\tilde{x}_\mathrm{d}(t)$ between neighboring samples $x_\mathrm{d}(kT)$ and $x_\mathrm{d}((k+1)T)$ is constructed and differentiated: For $kT < t < (k+1)T$,

$$\tilde{x}_\mathrm{d}(t) \coloneqq (1 - \alpha(t))\, x_{\mathrm{d},k} + \alpha(t)\, x_{\mathrm{d},k+1}, \tag{194}$$

$$\alpha(t) \coloneqq \frac{t - kT}{T}, \tag{195}$$

$$\Rightarrow\ \dot{\tilde{x}}_\mathrm{d} = \frac{1}{T}(x_{\mathrm{d},k+1} - x_{\mathrm{d},k}) = \frac{1}{T}(f_\mathrm{d}(x_{\mathrm{p},k}, x_{\mathrm{d},k}) - x_{\mathrm{d},k}). \tag{196}$$

To derive a continuous-time approximation $x_\mathrm{d,cont} \coloneqq\approx \tilde{x}_\mathrm{d}$ of the controller state, it is assumed that x_d and x_p change slowly, i.e., $x_\mathrm{p}(t) \approx x_{\mathrm{p},k}$ and $\tilde{x}_\mathrm{d}(t) \approx x_{\mathrm{d},k}$. Inserting these approximations into (196) yields

$$\dot{x}_\mathrm{d,cont} \coloneqq \frac{1}{T}(f_\mathrm{d}(x_\mathrm{p}, x_\mathrm{d,cont}) - x_\mathrm{d,cont}), \tag{197}$$

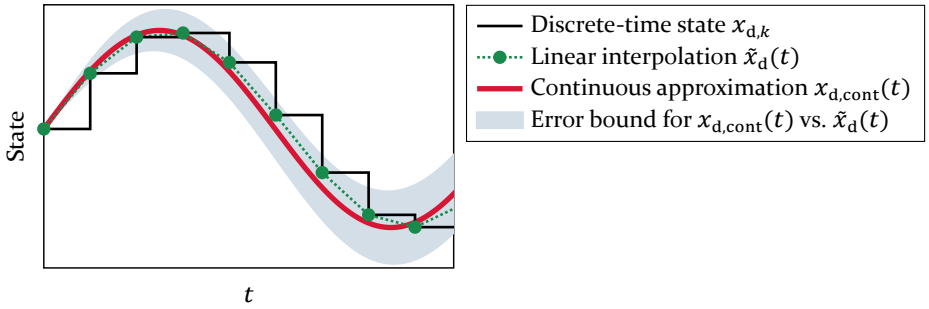


Figure 33: Principle of first-order continuization: A discrete-time system is smoothed by linear interpolation and then approximated as a continuous-time differential equation. The approximation error is modeled as disturbance.

whose solution $x_{\mathrm{d,cont}}$ approximates the linear interpolation $\tilde{x}_\mathrm{d}$ of the discrete-time signal x_d. For the exemplary integral controller (192), this yields

$$\dot{x}_{\mathrm{d,cont}} = \frac{1}{T} x_\mathrm{p}, \tag{198}$$

which recovers the desired integrator dynamics: The discrete-time integrator has been continuized. For a formal application of this idea, the difference between $\dot{x}_{\mathrm{d,cont}}$ and $\dot{\tilde{x}}_\mathrm{d}$ is considered as a bounded disturbance.

Approach Building on this idea, a general approach to first-order continuization can be derived. It is shown in Figure 34, which will be gradually explained in the following. The building blocks are the same as for zero-order continuization: first, a state transformation from H_0 to H_1 such that $H_0 =_x H_1$ with $x = q(\tilde{x})$ and second, an abstraction to $H_2 \supseteq_x H_1$.

H_0 implements the control loop (193), including controller dynamics (highlighted in red). The trajectories of H_0 and H_1 are sketched in Figure 34 (top right). H_0 uses the plant and controller state x_p and x_d. H_1 uses a larger number of states to achieve a similar structure as for zero-order continuization: The plant state x_p and interpolated controller state $\tilde{x}_\mathrm{d}$ have mostly smooth dynamics. Discrete-time effects are represented using sampling errors δ_p and δ_d, which are periodically reset to zero. Lastly, τ remains unchanged.

The states of H_1 are constructed such that they represent the states of H_0 and additional sample-and-hold versions thereof (Figure 34, top right): For the plant state x_p, a sampled version $x_\mathrm{p}^- := x_\mathrm{p} + \delta_\mathrm{p}$ is introduced, using the sampling error δ_p with dynamics of $\dot{\delta}_\mathrm{p} = -\dot{x}_\mathrm{p}$. The new controller state $\tilde{x}_\mathrm{d}$ is a delayed and interpolated version of x_d. Furthermore, $x_\mathrm{d}^- := \tilde{x}_\mathrm{d} + \delta_\mathrm{d}$ is

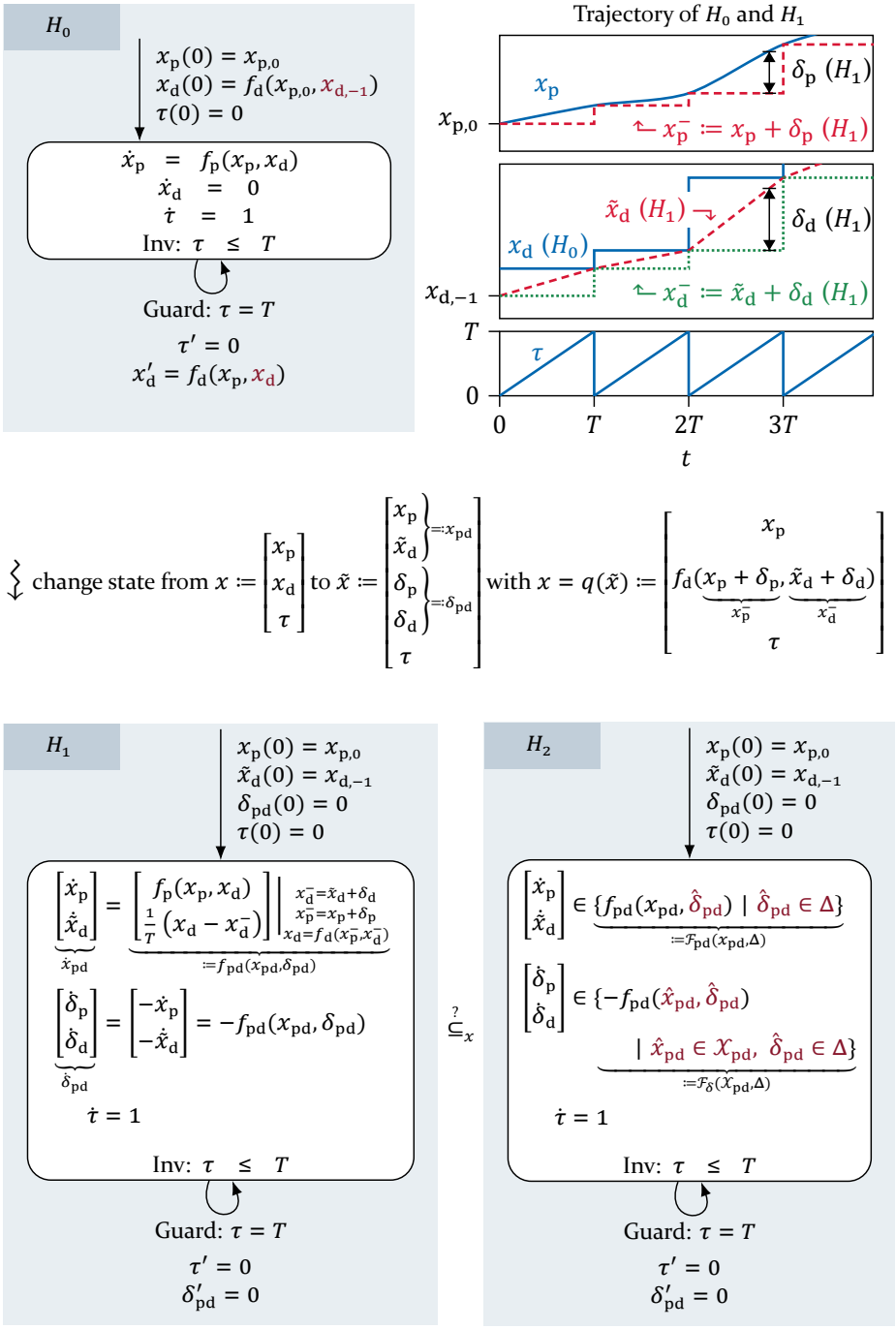


Figure 34: Hybrid automata of control loop and first-order continuization in Theorem 5.2.10

a delayed version of x_d, implemented using a state variable δ_d analogous to δ_p. The original controller state x_d can be reconstructed from these delayed versions by $f_\mathrm{d}(x_\mathrm{p}^-, x_\mathrm{d}^-)$ This results in the automaton H_1 in Figure 34 (bottom left).

Finally, an abstraction H_2 is constructed to avoid the discrete-time dynamics (highlighted in red), i.e., the dynamics of $\delta_{\dots}$ and the dependency of x_p and $\tilde{x}_\mathrm{d}$ on the sampling errors $\delta_{\dots}$. The behavior of $\tilde{x}_\mathrm{d}$ in H_2 can be interpreted as the continuous approximation with error bound from Figure 33. The correctness of the preceding informal derivation is stated by the following theorem:

Theorem 5.2.10 (First-Order Continuization)**.** *From Figure 34, consider the control loop H_0 with state vector x and its modifications H_1 and H_2 with state vector $\tilde{x}$. Assume that all functions are sufficiently well-behaved, as will be detailed later in the proof.*

Then, $H_0 =_x H_1$ with $x = q(\tilde{x})$ as given in the figure. Additionally, $H_1 \subseteq_x H_2$ if $\mathcal{X}'_\mathrm{pd} \subseteq \underline{\mathcal{X}}_\mathrm{pd}$ and $\Delta' \subseteq \underline{\Delta}$. Herein, $\underline{\Delta}$ and $\underline{\mathcal{X}}_\mathrm{pd}$ are arbitrarily chosen and represent a (possibly wrong) guess for the set of $\delta_\mathrm{pd} \coloneqq \begin{bmatrix}\delta_\mathrm{p}^\top & \delta_\mathrm{d}^\top\end{bmatrix}^\top$ and $x_\mathrm{pd} \coloneqq \begin{bmatrix}x_\mathrm{p}^\top & \tilde{x}_\mathrm{d}^\top\end{bmatrix}^\top$. $\mathcal{X}'_\mathrm{pd}$ and Δ' are defined as

$$\mathrm{Reach}_{x_\mathrm{pd}}(H_2) = \mathrm{Reach}_{x_\mathrm{pd}}\begin{pmatrix} \dot{x}_\mathrm{pd} \in \mathcal{F}_\mathrm{pd}(x_\mathrm{pd}, \Delta) \\ x_\mathrm{p}(0) = x_{\mathrm{p},0} \\ \tilde{x}_\mathrm{d}(0) = x_{\mathrm{d},-1} \end{pmatrix} =: \mathcal{X}'_\mathrm{pd}, \tag{199}$$

$$\mathrm{Reach}_{\delta_\mathrm{pd}}(H_2) = \mathrm{Reach}_{\delta_\mathrm{pd}}\left(\begin{pmatrix} \dot{\delta}_\mathrm{pd} \in \mathcal{F}_\delta(\mathcal{X}_\mathrm{pd}, \Delta) \\ \delta_\mathrm{pd}(0) = 0 \end{pmatrix}, [0;T]\right) \tag{200}$$

$$\subseteq [0;T] \otimes \square(\mathcal{F}_\delta(\mathcal{X}_\mathrm{pd}, \Delta)) =: \Delta', \tag{201}$$

where $\mathcal{X}_\mathrm{pd} \coloneqq \underline{\mathcal{X}}_\mathrm{pd} \oplus \mathbb{B}_\epsilon$, $\Delta \coloneqq \underline{\Delta} \oplus \mathbb{B}_\epsilon$, and $\mathcal{F}_\mathrm{pd}$, F_δ are given in H_2 in Figure 34.

Proof. The proof for $H_0 =_x H_1$ is detached to Appendix C.3. For $H_1 \subseteq_x H_2$, the proof is analogous to Theorem 5.2.4 and therefore omitted. □

5.2.6 Experiments

This section evaluates an implementation of first-order continuization using the same tools as the initial reachability experiments in Section 4.2.10, namely SpaceEx and HyST, together with mpmath for interval computations. The following experiments consider the linear case with ideal timing, which can be summarized as

$$\dot{x}_\mathrm{p} = f_\mathrm{p}(x_\mathrm{p}, x_\mathrm{d}) = A_\mathrm{p} x_\mathrm{p} + B_\mathrm{pd} x_\mathrm{d}, \tag{202a}$$
$$x'_\mathrm{d} = f_\mathrm{d}(x_\mathrm{p}, x_\mathrm{d}) = B_\mathrm{dp} x_\mathrm{p} + A_\mathrm{d} x_\mathrm{d} \quad \text{at } t = kT. \tag{202b}$$

Zero-order continuization is not evaluated, as this was already done in Bak and Johnson, 2015. The implementation mostly follows Section 5.2.5; further details are given next. The source code is referenced in Appendix F.

5.2.6.1 Implementation Details

This section gives details on the implementation of the experiments. From H_1 in Figure 34, the first-order continuization of the linear case used here is

$$\dot{x}_\mathrm{pd} \overset{H_1,(202a)}{=} \begin{bmatrix} A_\mathrm{p} x_\mathrm{p} + B_\mathrm{pd} f_\mathrm{d}(x_\mathrm{p} + \delta_\mathrm{p}, \tilde{x}_\mathrm{d} + \delta_\mathrm{d}) \\ \frac{1}{T}(f_\mathrm{d}(x_\mathrm{p} + \delta_\mathrm{p}, \tilde{x}_\mathrm{d} + \delta_\mathrm{d}) - \tilde{x}_\mathrm{d} - \delta_\mathrm{d}) \end{bmatrix} \tag{203}$$

$$\overset{(202b)}{=} \begin{bmatrix} A_\mathrm{p} x_\mathrm{p} + B_\mathrm{pd} \left(B_\mathrm{dp}(x_\mathrm{p} + \delta_\mathrm{p}) + A_\mathrm{d}(\tilde{x}_\mathrm{d} + \delta_\mathrm{d}) \right) \\ \frac{1}{T}(B_\mathrm{dp}(x_\mathrm{p} + \delta_\mathrm{p}) + (A_\mathrm{d} - I)(\tilde{x}_\mathrm{d} + \delta_\mathrm{d})) \end{bmatrix} \tag{204}$$

$$= \begin{bmatrix} A_\mathrm{p} + B_\mathrm{pd} B_\mathrm{dp} & B_\mathrm{pd} A_\mathrm{d} & B_\mathrm{pd} B_\mathrm{dp} & B_\mathrm{pd} A_\mathrm{d} \\ \frac{1}{T} B_\mathrm{dp} & \frac{1}{T}(A_\mathrm{d} - I) & \frac{1}{T} B_\mathrm{dp} & \frac{1}{T}(A_\mathrm{d} - I) \end{bmatrix} \begin{bmatrix} x_\mathrm{p} \\ \tilde{x}_\mathrm{d} \\ \delta_\mathrm{p} \\ \delta_\mathrm{d} \end{bmatrix} \tag{205}$$

$$\overset{H_1}{=} f_\mathrm{pd}(x_\mathrm{pd}, \delta_\mathrm{pd}). \tag{206}$$

Intervals are used for the sets $\mathcal{X}_\mathrm{pd}$ and Δ. Analogous to Remark 5.2.7, $\underline{\mathcal{X}}_\mathrm{pd} = \square \mathcal{X}'_\mathrm{pd}$ is used. The bloating ϵ in Theorem 5.2.10 is chosen as $\epsilon = 10^{-10}$.

The analysis is executed as follows:

1. The iteration starts with the empty set $\underline{\Delta} = \{\}$ as initial guess.
2. Repeat up to five times:

- If the condition of Theorem 5.2.10 is not satisfied, enlarge $\underline{\Delta}$ by $\underline{\Delta} = 2\square\Delta'$.
- If it is satisfied, the analysis was successful. To reduce excess growth, shrink $\underline{\Delta}$ to $\underline{\Delta} = \Delta'$ and re-run the analysis per Theorem 5.2.10, which will succeed by construction. Return the resulting bounds and stop.

3. If the maximum number of iterations was reached, return "continuization failed".

The settings of SpaceEx are the same as in Section 4.2.10.

5.2.6.2 Results

The following examples are based on Example C1 (Appendix A), which is a one-axis angular rate control for a quadrotor helicopter using a PI-controller approximating $u = -K_\mathrm{p} x_\mathrm{p} - K_I \int x_\mathrm{p}\,\mathrm{d}t$. In this example, the period $T = 0.01$ was chosen such that the behavior differs from the continuous equivalent by about 20 percent (Figure 45 on page 206).

The discretized implementation incurs a one-period processing delay from x_p to u. However, the formulation used here in Section 5.2.5 does not directly admit this delay, as there is no extra sample-and-hold mechanism between $x_\mathrm{d}(t)$ and the plant input $u(t)$. To avoid extra delay states, the example is simplified by dropping the delay:

Example C3 (PI discrete). Choosing $x_\mathrm{d}^{(1)} \approx \int x_\mathrm{p}\,\mathrm{d}t$ and $x_\mathrm{d}^{(2)} \approx x_\mathrm{p}$ leads to

$$A_\mathrm{p} = 0, \quad B_\mathrm{pd} = \frac{1}{J_x}\begin{bmatrix} -K_I & -K_P \end{bmatrix}, \quad B_\mathrm{dp} = \begin{bmatrix} T \\ 1 \end{bmatrix}, \quad A_\mathrm{d} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{207}$$

Continuization of this system fails: No valid error bound can be determined, as the iteration does not converge even for the significantly smaller period $T = 0.001$. Presumably this is due to the zero eigenvalue of A_d. Such discrete-time dead-beat behavior is problematic because has no proper continuous equivalent. It is, however, required here to model the feedthrough from x_p to u, which must be implemented via x_d. This variant is therefore excluded from further analysis.

Example C4 (P continuous, I discrete). As a simple academic example, the proportional part of the controller is changed to continuous time and merged with the plant. The remaining controller is a discrete-time integrator.

$$A_\mathrm{p} = \frac{-K_P}{J_x}, \quad B_\mathrm{pd} = \frac{-K_I}{J_x}, \quad B_\mathrm{dp} = T, \quad A_\mathrm{d} = 1, \quad T = 0.003 \tag{208}$$

For $T = 0.003$, continuization shows reasonable bounds, as detailed later. For larger periods, the bounds grow until continuization fails near $T = 0.01$.

Example C5 (P as discrete lowpass, I discrete). As a more realistic variant, the proportional feedback is lowpass-filtered in discrete time with an equivalent time constant of $T_L = 0.1$:

$$A_\mathrm{p} = 0, \qquad B_\mathrm{pd} = \frac{1}{J_x}\begin{bmatrix} -K_I & -K_P \end{bmatrix}, \tag{209}$$

$$B_\mathrm{dp} = \begin{bmatrix} T \\ 1 - \mathrm{e}^{-T/T_L} \end{bmatrix}, \qquad A_\mathrm{d} = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-T/T_L} \end{bmatrix}, \qquad T = 0.001 \tag{210}$$

For $T_L \to 0$, this would be equivalent to Example C3. With the smaller period of $T = 0.001$, results are similarly positive as in the previous example.

Graphical Results To illustrate the results, the behavior of the original and continuized systems for C4 and C5 is shown in Figure 35. The left column shows the outer approximation of the reachable set $\mathrm{Reach}_{x_\mathrm{p}}(H_i, t)$ of the plant state $x_\mathrm{p}(t)$ determined using SpaceEx. The remaining columns show randomized simulations of x_p (middle) and the first controller state $x_\mathrm{d}^{(1)}$ (right) using PySim. It can be seen that the original systems are stable but ill-suited for reachability analysis: The approximated reachable set diverges (C4) or can not be computed within ten hours (C5), although the simulations indicate stability. It should be noted that because this experiment is restricted to strictly periodic controllers, i.e., zero timing deviation, it should in principle be possible to solve these systems without continuization, as discussed in Section 4.2.10 and in the introduction of Section 5.2.

After continuization, reachability analysis is easy: SpaceEx returns a useful result with little pessimism compared to the simulations. The only downside is the pessimism incurred by continuization itself. Particularly, some steady-state uncertainty remains even if the original system converges to zero. This results from abstracting the sampling error by a constant set.

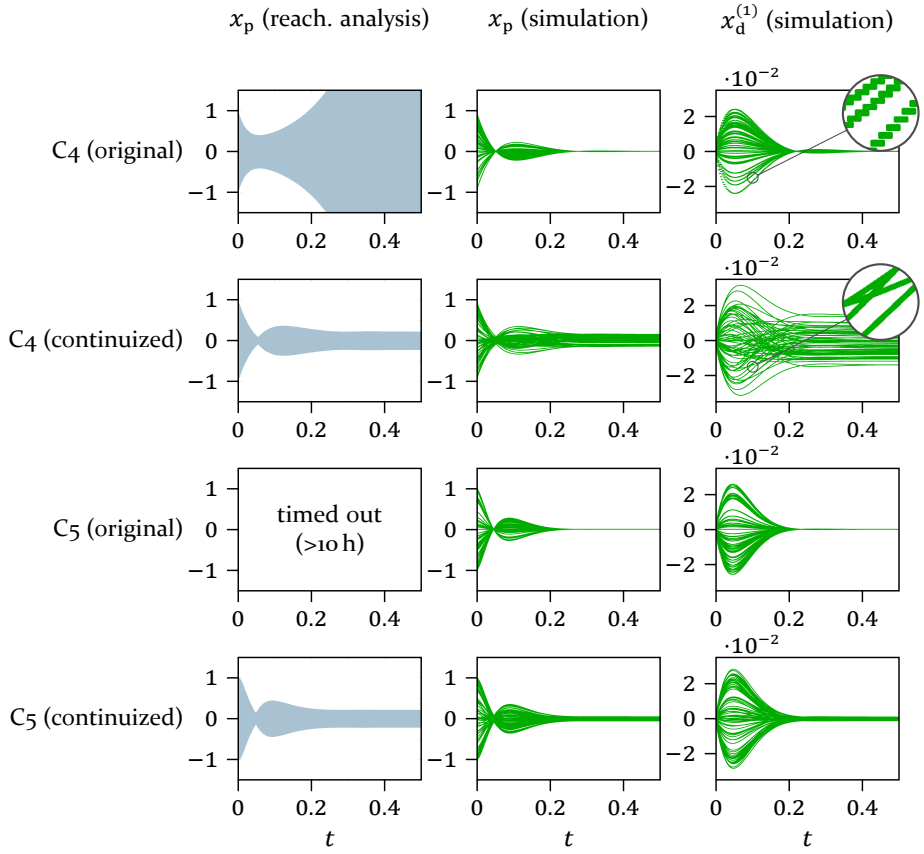


Figure 35: Reachability analysis and randomized simulations for the example systems

5.2.7 Extension to Uncertain Input/Output Timing

An early feasibility study for continuization with IO time windows was undertaken in the master thesis of Wu, 2019 that was supervised during the research on this doctoral thesis.

Theorem 5.2.11 (First-Order Continuization for Uncertain IO Timing)**.** *Consider a linear quasi-SISO variant of the system model from Chapter 3:*

$$\dot{x}_p = A_p x_p + B_p u, \qquad x_p(0) = x_{p,0}, \tag{211a}$$

$$x_{d,k+1} = A_d x_{d,k} + B_d y_k, \tag{211b}$$

$$u(t) = C_d x_{d,k}, \qquad kT + \Delta t_{u,k} \le t < (k+1)T + \Delta t_{u,k+1}, \tag{211c}$$

$$y_k = C_p x_p(kT + \Delta t_{y,k}). \tag{211d}$$

According to Wu, 2019, this system can be continuized as

$$\dot{x}_\mathrm{p} = A_\mathrm{p} x_\mathrm{p} + B_\mathrm{p} u, \qquad x_\mathrm{p}(0) = x_{\mathrm{p},0}, \tag{212a}$$

$$\dot{\tilde{x}}_\mathrm{d} = \frac{A_\mathrm{d} - I}{T} \tilde{x}_\mathrm{d} + \frac{B_\mathrm{d} C_\mathrm{p}}{T} x_\mathrm{p}, \quad \tilde{x}_\mathrm{d}(0) = x_{\mathrm{d},0}, \tag{212b}$$

$$u = C_\mathrm{d} \tilde{x}_\mathrm{d} + \delta_\mathrm{u}, \tag{212c}$$

$$y = C_\mathrm{p} x_\mathrm{p} + \delta_\mathrm{y}, \tag{212d}$$

$$|\delta_\mathrm{u}| < T_\mathrm{u} \|C_\mathrm{d}\|_2 \max_{t>0} |\dot{x}_\mathrm{p}(t)|, \tag{212e}$$

$$|\delta_\mathrm{y}| < T_\mathrm{y} \|C_\mathrm{p}\|_2 \max_{t>0} |\dot{\tilde{x}}_\mathrm{d}(t)|. \tag{212f}$$

Herein, the factors T_u and T_y depend on the timing bounds of u and y:

- *For varying timing within $(-T/2; T/2)$, they are $T_\mathrm{u} = T_\mathrm{y} = 3T/2$.*
- *For constant timing $\Delta t_{\mathrm{u},k} \equiv \Delta t_\mathrm{u} = \text{const}$ and $\Delta t_{\mathrm{y},k} \equiv \Delta t_\mathrm{y} = \text{const}$, all within $(-T/2; T/2)$, they are $T_\mathrm{u} = T + \Delta t_\mathrm{u}$ and $T_\mathrm{y} = T - \Delta t_\mathrm{y}$. (Note the different signs.)*

T_u and T_y can be loosely interpreted as the "maximum data age": the bounds for $|\delta_\mathrm{u}|$ and $|\delta_\mathrm{y}|$ decrease with earlier actuation and later measurement.

Proof. The derivation is given in Wu, 2019. It does not use hybrid automata but an explicit notation for the discrete events and a less rigorous formalism. □

For a numerical implementation, one must first resolve the circular dependency between the bounds for $|\delta_\mathrm{u}|$, $|\delta_\mathrm{y}|$ and the derivatives $\dot{x}_\mathrm{p}$, $\dot{\tilde{x}}_\mathrm{d}$. This is possible using Theorem 5.2.1. With the formal framework of this thesis, it should be possible to put the above result onto a rigid formal basis. This is sketched in the following, while the details are left for future work. The suggested steps follow the same scheme as before in Theorems 5.2.4 and 5.2.10:

- Introduce multiple sampling-deviation variables $\delta_\ldots$.
- Transform to an extended state vector $\tilde{x}$ so that, as far as possible, all transitions become resets of the form $\delta'_\ldots = 0$. These periodic resets ensure that the sampling deviations can later be bounded by Lemma 5.2.2.
- Determine the mapping $x = q(\tilde{x})$ so that u, x_d and y_d are expressed by new quasi-continuous states and the sampling-deviation variables $\delta_\ldots$.

- Prove x-equivalence of the transformed automaton as in Appendix C.3. Some of the simplifying assumptions of the existing proof will need to be lifted. For example, the existing proof assumes an uninterrupted continuous transition within one period, which is no longer true because the sampling event occurs at an unknown time in between.
- Abstract the automaton to remove the dynamics of the sampling deviation. Prove the x-abstraction analogous to Theorem 5.2.4.

5.2.8 Discussion

This section presented continuization, a method for the analysis of real-time control systems whose behavior is similar to a continuous-time differential equation. Two variants were derived: *zero-order continuization* is equivalent to previous work by Bak and Johnson. However, this method is limited to static controllers, i.e., controllers with only feedthrough and no dynamics. As a complement, the novel method of *first-order continuization* is presented. For the first time, this method allows the continuization of controllers with internal dynamics, e.g., an integral part or lowpass filter. Experiments show it can be successfully applied to dynamic controllers *without feedthrough* if the period is small enough to approximate continuous-time behavior. However, controllers with dynamics *and* feedthrough can not be solved yet.

In summary, first-order continuization and the underlying formal framework are a promising step towards the verification of real-time control systems with uncertain timing. The results point to a number of potential extensions to reduce pessimism and support more general timing scenarios:

5.2.8.1 Timing

This chapter mainly considered strictly periodic execution. An extension to relaxed timing is of high practical importance, e.g., to uncertain periods or uncertain IO timing. The former was already solved by Bak and Johnson, 2015 for zero-order continuization, and the latter was sketched in Section 5.2.7. Both cases should be solvable in the formal framework presented here by adding additional sample-and-hold states analogous to δ_p and δ_d. With some pessimism, it should also be possible to reuse the results for skipped events because skips are related to a delay of one period.

MIMO An extension to the MIMO case of uncertain timing requires a technique that works for an arbitrary number of sensors and actuators. For the linear case, it may be possible to split the system behavior into a sum, analogous to Theorem 5.1.4. A more general approach is to compose the automaton from multiple sub-automata, to which the transformation and abstraction steps are applied individually. A hurdle to overcome is that x-equivalence and x-abstraction are not necessarily preserved under composition: As x-equivalence only considers the specified subset of the continuous state variables and neglects the transition labels, $H_a =_x H'_a$ and $H_b =_x H'_b$ does not imply $H_a \parallel H_b =_x H'_a \parallel H'_b$. A solution to this compositional verification would also open the door for verifying hierarchical or multi-rate control loops and similar high-complexity systems.

Varying Sample Rate To model uncertainty in the sampling period T, the guard condition $\tau = T$ of $H_{0,1,2}$ (Theorem 5.2.4 or Theorem 5.2.10) can be replaced with an inequality $\underline{T} < \tau < \bar{T}$. The factor $1/T$ in the transformed dynamics H_1 becomes an uncertain variable, changing them to a differential inclusion. The abstraction can be derived as before.

Skipped Executions (M, K)-weak controller execution can be modeled by an automaton for the skips, as discussed on page 92. This leads to multiple locations in the original automaton H_0. A straightforward approach is to transform it into an automaton H_1 with the same number of locations. The proof and its assumptions will need to be adapted accordingly. By Conjecture C.3 of Appendix C, having multiple locations is no problem for the abstraction theorem. While the resulting automaton is no longer purely continuous, analysis may still be feasible if the ratio of skips to regular executions is low.

If the skips only affect the sensing and actuation but not the controller update, this may also be considered as a longer maximum delay: Skipping the sensor or actuator update corresponds to skipping the respective transition $\delta_{\dots} = 0$, which leads to a higher maximum time in the bound of Lemma 5.2.2.

Generalization From an abstract mathematical point of view, it is interesting if the transformation and abstraction process can be generalized to arbitrary timed automata. Then, it could be performed automatically for a broad range of timing scenarios.

5.2.8.2 Dynamics

Regarding the class of systems, the presented formalism should allow for generalization with little effort: The theory already covers nonlinear dynamics, and the results should hold unchanged for the case of bounded disturbance.

Sample-and-Hold and Quantization The derivations assume that the output uses zero-order hold and that the input is sampled instantaneously. However, the abstraction to $\delta_{\mathrm{pd}} \in \underline{\Delta}$ potentially includes a broader set of possible sample-and-hold mechanisms. In particular, first-order continuization is closely related to first-order hold at the output. Furthermore, the effects of quantization could be easily included as an additional bounded uncertainty.

Hybrid Controllers The importance of hybrid systems is growing with more complex functionality that switches between multiple modes. For example, adaptive cruise control of cars behaves differently depending on the presence of a car in front, and verification must ensure that the resulting mode transitions do not lead to unsafe behavior. In this case, the controller itself would be described as a hybrid automaton, and the continuization must be extended appropriately (cf. Conjecture C.3 of Appendix C). For these hybrid systems, reachability analysis as employed here is particularly suitable.

5.2.8.3 Accuracy

Steady-State Uncertainty In the experiments, the reachable set over time does not converge to zero because the assumed disturbance bound is constant over time. This could be mitigated by "restarting" with a lower disturbance bound after some time has passed (Bak and Johnson, 2015, Lemma 2). A further possibility is a *destabilizing* state transformation from $x(t)$ to $\mathrm{e}^{\lambda t}x(t)$, where $\lambda > 0$. If the destabilized system is proven stable, then the original system is exponentially stable with a decay of $\mathrm{e}^{-\lambda t}$ or better. This was successfully used in Wu, 2019 for linear systems without disturbance.

Continuization Order To analyze controllers with internal dynamics *and* feedthrough, e.g., the common PI controller, one could combine zero- and first-order continuization. Another possible direction of research could be higher-order continuization, analogous to higher-order (e.g., Runge-Kutta) numerical integration methods. This may reduce the pessimism that restricted the experiments to small sample times.

5.2.8.4 Related Approaches

Continuization is closely related to multiple approaches commonly used outside the field of hybrid automata. Using these connections is an interesting subject for future research.

An early application of continuization by Althoff, Yaldiz, et al., 2011 shows a relation to the stability of arbitrarily switched continuous-time systems: Consider the switched system $\dot{x} = f_{\sigma(t)}(x)$, where $\sigma(t) \in \{0, 1\}$ is a switching variable depending on some timing mechanism. To simplify the discussion, assume that the timing is truly arbitrary, so that $\sigma(t)$ is an unrestricted switching variable. The system is rewritten as $\dot{x} = \sigma f_1(x) + (1 - \sigma) f_0(x)$ and then abstracted to $\sigma \in [0; 1]$. The resulting uncertain continuous-time dynamics can be rewritten as the convex hull $\dot{x} \in \text{conv}\{f_0(x), f_1(x)\}$, similar to the interval hull in Lemma 5.2.2 and its proof. For linear systems with arbitrary switching, this convex hull is equivalent with regard to stability (Hetel, 2007, p. 9), analogous to the discrete-time case (Section 2.5.2.1).

A major similarity exists with the input/output approach to stability analysis as discussed in Section 2.5.2.2: The system is rewritten to make the "sampling error" explicit and then replace the complicated dynamics of this error by a simple abstraction. The difference lies in the further analysis methods and the supported system model: In this thesis, the framework is geared towards general hybrid automata and uses bounds on the state sets. For the input/output approach, literature is instead focused on operator-gain bounds and corresponding Lyapunov methods, which may admit more precise analysis but can not be directly applied to hybrid systems.

5.3 Concluding Remarks

Periodic IO time windows are a practical alternative to the traditional requirement of ideal IO timing. In practice, time windows are widely used because they result if the ideal timing is only implemented up to some tolerance. In the real-time operating system, they can be readily implemented using periodic tasks with release times and deadlines, removing the need for specialized hardware synchronization of IO operations. This chapter presented two solutions to analyze the stability of real-time control systems under this new timing.

The key challenge to verifying the stability of real-time control systems with uncertain timing are the hybrid dynamics, which merge continuous and discrete behavior. The presented solutions solve the problem by avoiding one half of it: Effectively, continuization of HA removes the discrete transitions, while discretization of LIS removes the continuous behavior.

The discretization-based LIS analysis exploits the structure of the system to simplify the dependencies on timing. With some conservatism, the method was successfully applied to an example with uncertain timing for multiple inputs and outputs. Theoretically, the method is guaranteed to prove stability for sufficiently small timing deviation, which matches the practical experience that small timing uncertainty can be neglected. The main limitation, which may be impossible to overcome, is that only linear systems are supported.

On the other hand, continuization of HA supports nonlinear systems. Unlike the previous method, it requires fast sampling so that a good continuous-time approximation exists. In its current form, continuization was successfully applied to simplified examples. Extensions towards the full system model of Section 3.2 were sketched and are a promising candidate for future research.

While both solutions employ different formal frameworks, they admit a common interpretation in terms of state sets: Reachability analysis immediately yields the set of states over time. For the LIS-based stability analysis, the connection is less obvious: A CQLF $V_P(x_k)$ with an upper bound $V_P(x_k) \leq r_k^2$ is equivalent to an ellipsoidal set $\{x_k \mid V_P(x_k) \leq r_k^2\} = \{x_k \mid x_k^\top P x_k \leq r_k^2\}$, where P defines the fixed shape and r_k the time-varying size. Let a bound $V_P(x_0) \leq r_0^2$ on the initial state be given. Then, in every time step the size r_k decays at least according to the P-ellipsoidal norm: $r_{k+1} = \sqrt{V_P(x_{k+1})} \overset{(108)}{\leq} \|A_k\|_P \sqrt{V_P(x_k)}$. Hence, the presented LIS stability analysis is related to set-valued reachability analysis with sets of fixed ellipsoidal shape.

The results of the current chapter, particularly the norm-based stability analysis, were limited to an execution in which every period has the same timing bounds. However, this excludes skips because in practice one cannot allow skipping for all periods. If, for example, skipping is only allowed every second period, then the methods must be modified. To handle this case, the following chapter will explore the connection between stability analysis and state sets: The size r_k of these sets can be considered as a state that evolves over time, e.g., $r_{k+1} = \|A_k\|_P \, r_k$. Then, stability is shown if this state converges.

6 Convergence Rate Abstractions

The previous chapter made a first step away from the classical requirement that the IO timing of real-time control systems must be strictly periodic: Stability verification was considered for an IO time window, i.e., a fixed tolerance for the times of sampling and actuation. However, requiring the same timing bounds in every period is still stricter than necessary: Due to inertia, the physical effect of timing is "smoothed out" over multiple periods. Therefore, rare short-term violations of the "timing stability limit" are acceptable as long as the "average" execution is good enough.

As outlined in Section 3.2, the vision of this thesis is to use this timing flexibility as far as possible: A complex control system with multiple sensors and actuators should employ a simple algorithm at run-time to adapt the timings and skip ratios to the current need while guaranteeing safety. However, designing such an algorithm is nontrivial due to a dilemma between complexity and pessimism: The computational cost of the algorithm at run-time must not exceed the benefits of higher timing flexibility. On the other hand, simplifying computations too much will lead to pessimistic estimates and therefore unnecessarily restricted timing.

Figure 36 illustrates the dilemma by a system with state $x = \left[x^{(1)}\ x^{(2)}\ x^{(3)}\right]^\mathsf{T}$ that is disturbed by a known but varying skipping of actuations (e.g., due to deadline misses). It is assumed that the system is stable for a 1-of-10 skipping with rare short-term episodes of 4-of-10.

The original, full-dimensional system model (Figure 36, left) allows for exact results: the system remains stable and the initial disturbance decays despite the short-term "instability". However, safety analysis and even more the decision making at run-time are burdened by complexity. For the considered timing model, the individual delays and skips for each sensor and actuator lead to a large space of timing sequences. As detailed in Section 2.5.4, a direct extension of existing techniques seems infeasible given their complexity for much simpler decisions, such as quasi-SISO (M, K)-skips.

Contrary, analysis for a fixed timing specification (Figure 36, right) is done at design-time and therefore free of run-time overheads. However, it is inherently pessimistic as it must always assume the worst timing pattern. In this example, the system state is unstable (exponentially growing) for the assumed maximum 4-of-10 skips, rendering design-time 4-of-10 guarantees *useless*. Therefore,

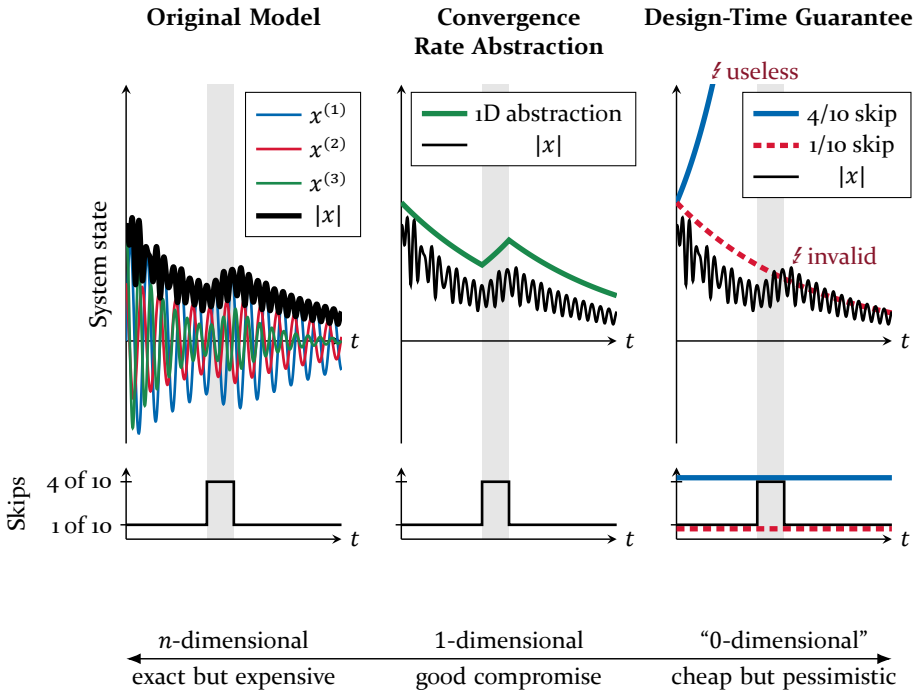


Figure 36: This illustration depicts that dynamic convergence rate abstractions (center) aim to hit a sweet spot for stability analysis between the high complexity of the original system (left) and the pessimism resulting from design-time guarantees (right). Note that the illustration is an artistic depiction and does not correspond to an actual system.

the considered scenario can not be shown stable using design-time guarantees within a "n-of-10" framework. At the same time, analysis for 1-of-10 skips (dashed line) is quite close to reality, yet its result becomes *invalid* in 4-of-10 conditions due to the violated assumption of 1-of-10 skips.

This calls for an appropriate *abstraction*, i.e., a simplification of the system behavior, that allows for decisions at run-time with reasonable complexity and pessimism. As a solution, this chapter introduces the concept of one-dimensional convergence rate abstractions that aim at a compromise between complexity and pessimism (Figure 36, center): In simple terms, the abstraction computes a scalar *damage counter* variable that increases or decreases over time depending on whether the current execution is bad or good. This variable is called the *abstract state* and represents a bound on the system

state, e.g., on $|x|$. Thereby, a bound can be estimated dynamically without considering the whole complexity of the original system. Because they are stateful, convergence rate abstractions can exploit the inertia of the system and temporarily go beyond the limits of design-time analysis.

"Good abstractions turn a nearly impossible task into two manageable ones" (Tanenbaum, 2009). Here, the task of dynamic resource management is split into two feasible parts: First, a simplified abstract system is derived at design time. Second, this abstraction is used in a run-time algorithm, so that safety can be guaranteed by construction.

Such abstractions appear in existing literature, although often implicitly. For example, Huang et al., 2019 prove the stability of a nonlinear real-time control system with (M, K)-weak execution with *fixed* parameters by considering an upper bound for $|x|$ similar to the graph in Figure 36 (center). As will be detailed later, a key contribution of this chapter is to consider the abstraction as a dynamical system of its own.

This chapter is structured as follows: First, the formalism is defined and compared with other concepts of this thesis (Sections 6.2 and 6.3). At its core, the convergence rate abstraction is a time-varying dynamic bound, which is implemented in Sections 6.4 and 6.5 for nominal and then for weak execution. In Chapter 7, this will form the theoretical basis for dynamic resource management. Here, it is first specialized to (M, K)-weak execution in Section 6.6. Finally, Sections 6.7 and 6.8 discuss related work and summarize the results.

Earlier versions of this chapter were published in Gaukler, Rheinfels, et al., 2019 and as an internal technical report (Gaukler, 2020b).

6.1 Notation

For brevity, the following conventions are used throughout this chapter:

- If not stated differently, k refers to all $k \in \mathbb{N}_0$, and x to all $x \in \mathbb{R}^n$.
- The abbreviation $a \equiv 0$ denotes $a_0 = a_1 = \cdots = 0$.

6.2 Problem Statement

Basically, the goal is to find a one-dimensional dynamic system with state v_k that provides an upper bound such as $|x_k| \leq v_k$ for the state x_k of a real-time control system with varying timing. Analysis can then be performed using only the simplified v-system.

Given A real-time control system with uncertain execution $\sigma_k \in \Sigma$ is given by the general discrete-time dynamics

$$x_{k+1} = f_{\sigma_k}(x_k) + g_{\sigma_k}(d_k) \overset{\text{lin}}{=} A(\sigma_k)x_k + G(\sigma_k)d_k, \quad |d_k| \leq |d_k|_{\max} \tag{213a}$$

with disturbance d_k, safety-relevant output

$$s_k = \bar{C}_\text{s} x_k, \quad \bar{C}_\text{s} \neq 0 \tag{213b}$$

and initialization

$$x_0 = C_0 \tilde{x}_0, \quad |\tilde{x}_0| \leq |\tilde{x}_0|_{\max}, \quad C_0 \neq 0. \tag{213c}$$

The set Σ of possible executions contains the nominal execution $\sigma = 0$ and possibly other values, i.e., $\Sigma \supseteq \{0\}$. Furthermore, $\tilde{x}_0 \in \mathbb{R}^{n_0}$ is a normalized initial state and $C_0 \in \mathbb{R}^{n \times n_0}$ parametrizes the ellipsoidal initial set.

In some aspects, this model deviates from or goes beyond the discretized linear real-time control system from Section 4.1.4.6:

- Nonlinear dynamics are supported. However, the discussion in this chapter will be mainly limited to the linear case.
- The execution influence σ_k is not restricted to the definition of Section 3.1: In general, Σ can be any set containing a zero element $\sigma_k = 0$ that represents a "safety mode" whose timing is good enough to ensure safety for worst-case disturbance. For example, $\Sigma = \{0, 1\}$ could model the case where the controller is either completely skipped ($\sigma_k = 1$) or executed normally ($\sigma_k = 0$).
- The initial set is an ellipsoid. According to (48), the controller states are initialized as zero, so C_0 must be zero except for an upper part $C_{\text{p},0}$:

$$x_0 = C_0 \tilde{x}_0 = \begin{bmatrix} x_{\text{p},0} \\ 0 \end{bmatrix} =: \begin{bmatrix} C_{\text{p},0} \\ 0 \end{bmatrix} \tilde{x}_0. \tag{214}$$

  Then, $x_{\text{p},0} \in \mathcal{X}_{\text{p},0} = C_{\text{p},0} \mathbb{B}_{|\tilde{x}_0|_{\max}}$.
- The disturbance set is time-varying, but its shape is restricted to a ball. An ellipsoid shape can nevertheless be represented by using the input matrices G_p and H_p as shape matrices.

For feasibility, it is assumed that the control system is stable for ideal timing.

Assumption 6.2.1 (Nominal Exponential Stability)**.** *For ideal timing $\sigma \equiv 0$ and zero disturbance $d \equiv 0$, the dynamics are DGES(ρ, α). This means that there exist an overshoot factor $\alpha \geq 1$ and a growth rate $\rho \in (0; 1)$ such that*

$$(\forall k \in \mathbb{N}_0 : \ \sigma_k = 0 \wedge d_k = 0) \ \Rightarrow \ \left(\forall x_0 \in \mathbb{R}^n, k \in \mathbb{N}_0 : \ |x_k| \leq \alpha \rho^k |x_0|\right). \tag{215}$$

Goal The goal is to reduce the original n-dimensional system (213) with state x_k to a one-dimensional *abstract system* with scalar state $v_k \geq 0$. Instead of bounding $|x_k|$, only the safety-relevant output s_k is considered to provide a more precise bound $|s_k| \leq v_k$.

Definition 6.2.2. The abstract system

$$v_0 = \alpha |\tilde{x}_0|_{\max}, \tag{216a}$$
$$v_{k+1} = \rho_{\sigma_k} v_k + \beta_{\sigma_k} |d_k|_{\max} \tag{216b}$$

is an *output convergence rate abstraction* of the original system (213) if and only if the abstraction guarantee $|s_k| \leq v_k$ holds for all inputs, i.e.,

$$\begin{aligned} |s_k| \leq v_k \quad & \forall \tilde{x}_0 \text{ with } |\tilde{x}_0| \leq x_{0,\max} \\ & \forall d_{0,1,\dots,k-1} \text{ with } |d_i| \leq |d_i|_{\max} \\ & \forall \sigma_{0,1,\dots,k-1} \in \Sigma \\ & \forall k. \end{aligned} \tag{217}$$

◁

The parameters $\alpha > 0, \beta_\sigma > 0$ and $\rho_\sigma > 0$ for all $\sigma \in \Sigma$ should be determined such that the abstraction is a good bound for the desired stability and quality (cf. Figure 36, center, on p. 148).

Application As the abstraction is a safe overapproximation, it can be used to verify safety:

Theorem 6.2.3. *The condition*

$$v_k \leq |s|_{\max} \quad \forall k \tag{218}$$

for the abstraction is sufficient for the physical safety specification (26).

Proof. This immediately follows from (217): $|s_k| \overset{(217)}{\leq} v_k \overset{(218)}{\leq} |s|_{\max}$. □

An analogous result holds for exponential decay. This also translates to DGES of the system state, as long as $\bar{C}_s = I$ or an appropriate observability condition holds for the safety output s_k.

6.3 Conceptual Discussion

The idea of convergence rate abstraction picks up aspects of other concepts previously introduced in this thesis:

Connection to Abstractions If the abstraction guarantee is interpreted as a nondeterministic output equation

$$0 \leq \tilde{s}_k \leq v_k, \tag{219}$$

this results in a nondeterministic system S_{abstract} whose output $\tilde{s}_k$ contains all trajectories of $|s_k|$ between zero and the upper bound v_k. In this sense, the convergence rate abstraction is an abstraction (per Definition 4.2.9) of the original control system S: $S_{\text{abstract}} \supseteq_{\tilde{s}_k} S$ with the transformation $\tilde{s}_k = |s_k|$.

Connection to Reachability Analysis Denote the closed ball of radius v by $\mathbb{B}_v := \{x \in \mathbb{R}^n | \, |x| \leq v\}$. To simplify the discussion, let $s_k = x_k$, so there is no distinction between output and state. Then, the state v_k of the abstraction is a parametrization of an outer bound of the reachable output set $s_k \in \mathbb{B}_{v_k}$. This is roughly comparable to set-valued reachability analysis, where some data vector is used to represent the reachable set. However, there are two important distinctions: The data vector is typically of high dimension to accurately represent the set, while the dimension of the abstract state is one. Also, there is usually a one-to-one correspondence between the data vector and the approximated state set, which is not the case here:

As the abstraction returns a circular bound on the state, one could believe that it is the same as reachability analysis using circular bounds, loosely

$$x_{k+1} = f(x_k) \quad \rightsquigarrow \quad \mathcal{X}_{k+1} = \text{Ball}(f(\mathcal{X}_k)), \tag{220}$$

where $\text{Ball}(\mathcal{S})$ denotes the smallest enclosing ball $\mathbb{B}_r \supseteq \mathcal{S}$ of a set $\mathcal{S}$.

However, this reachability analysis would fail for many systems with "ellipsoidal" trajectories, although it will be shown later that stable abstractions always exist for nominal-case stability. Hence, v_k can not be directly interpreted as reachable set but merely as an outer bound. In some cases, the shape of the state set represented by v_k can be explicitly shown, which will be discussed later (Section 6.5.2.3).

6.4 Abstraction for Ideal Timing

For this section, consider the case of ideal timing ($\sigma \equiv 0$) and a disturbance that immediately acts on the state ($G(\sigma) \equiv I$). In this case, a convergence rate abstraction is mainly the reinterpretation of exponential stability in terms of a dynamical system: The bound $\alpha\rho^k|x_0|$ arising in DGES can be generated by the dynamical system $v_{k+1} = \rho v_k$ with $v_0 = \alpha|x_0|$.

To formalize this idea including the effect of disturbance, the stability definition is extended:

Definition 6.4.1 (Exponential Input-to-State Stability (EISS))**.** The system

$$x_{k+1} = f(x_k) + g(d_k) \tag{221}$$

is exponentially input-to-state stable, denoted EISS(ρ, α, β), if and only if there are constants $\rho \in (0; 1), \alpha \geq 1, \beta > 0$ such that

$$|x_k| \leq \alpha\rho^k|x_0| + \beta \sum_{i=0}^{k-1} \rho^{k-1-i}|d_i|. \tag{222}$$

for all $k \geq 0$, for all initial states x_0 and for all disturbance input $d_0, \ldots, d_{k-1}$. For $k = 0$, note that "backwards" sums such as $\sum_{i=0}^{-1}(\ldots)$ are defined as zero throughout this thesis. ◁

The bound resulting from this definition can be considered as the solution of a one-dimensional "abstract" dynamic system:

Theorem 6.4.2. *A system is EISS(ρ, α, β) if and only if the abstract system*

$$v_{k+1} = \rho v_k + \beta|d_k|, \quad v_0 = \alpha|x_0| \tag{223}$$

obeys the abstraction guarantee

$$|x_k| \leq v_k \quad \forall x_0,\ d_0, \ldots, d_{k-1} \quad \forall k. \tag{224}$$

This is a simple example of a one-dimensional *convergence rate abstraction*: The one-dimensional v-system summarizes the relevant information about stability of the n-dimensional x-system.

Proof. The explicit solution of v_k matches the right-hand side of (222), here denoted r_k: Start with $v_0 = r_0$, which is trivially true. For a proof by induction, assume $v_k = r_k$ to see that for $k \geq 0$

$$v_{k+1} \overset{(223)}{=} \rho \underbrace{v_k}_{r_k} + \beta |d_k| \overset{(222)}{=} \alpha \rho^{k+1} |x_0| + \beta \sum_{i=0}^{k-1} \rho^{k-i} |d_i| + \beta \rho^0 |d_k| \overset{(222)}{=} r_{k+1}. \tag{225}$$

Hence, $v_k = r_k$ for all k. □

The requirement of EISS is reasonable, as it is equivalent to the assumption of DGES for linear systems. With the assumption of Lipschitz continuity, similar but more pessimistic results for the nonlinear case are sketched in Gaukler, Rheinfels, et al., 2019, Theorem 3.4 and implicitly used in Huang et al., 2019.

Theorem 6.4.3. *A linear system*

$$x_{k+1} = Ax_k + Gd_k \tag{226}$$

is EISS(ρ, α, β) for some β if and only if it is is DGES(ρ, α) for $d \equiv 0$.

The proof is straightforward but nevertheless given in Appendix D.

In summary, the results of this section show that for nominal execution, there is an abstraction that accurately captures the stability result of exponential input-to-state stability and, assuming linear dynamics, equivalently of DGES.

6.5 Abstraction for Weak Execution

In the previous section, nominal execution $\sigma \equiv 0$ was assumed and abstractions were mainly a reinterpretation of stability that did not yet lead to new results. The actual benefit of abstractions appears for weak execution, i.e., if $\sigma \neq 0$ is permitted.

This section considers the linear case of the system model with disturbance given in Section 6.2,

$$x_{k+1} = A(\sigma_k)x_k + G(\sigma_k)d_k, \quad x_0 \in \mathbb{R}^n. \tag{227a}$$

Analogous results hold for nonlinear systems, as detailed in Gaukler, Rheinfels, et al., 2019, Section 6.

Two methods for determining the abstraction parameters $\rho_\sigma, \alpha, \beta_\sigma$ are presented in the following. The first considers the deviation from the nominal dynamics as disturbance analogous to Huang et al., 2019 and reuses the EISS results of Section 6.4. The second approach is based on Lyapunov functions.

Note that an implementation does not need to compute the parameters or the abstraction state v_k exactly. Instead, any upper approximation is also possible because it preserves the abstraction guarantee (224).

6.5.1 Simple Robustness-Based Abstraction

A simple but pessimistic abstraction can be derived by considering the deviation from the nominal dynamics as disturbance:

Theorem 6.5.1 (Robustness-Based Abstraction)**.** *Let $G(\sigma) = I$, so that the disturbance d_k acts directly on the state. If ρ_0, α, β_0 are parameters of an abstraction for the nominal case $\sigma \equiv 0$, e.g., by Section 6.4, then the same $\alpha, \beta_{\sigma_k} = \beta_0$ and*

$$\rho_{\sigma_k} \coloneqq \rho_0 + \beta_0 \gamma_{\sigma_k}, \tag{228}$$

are valid parameters of a new general abstraction for any σ_k, where

$$\gamma_\sigma \coloneqq \|A(\sigma) - A(0)\|_2 . \tag{229}$$

Proof Sketch. Denote the original abstraction by v_k. By assumption, $G(\sigma) = I$. The deviation of x_{k+1} resulting from $\sigma \neq 0$ is interpreted as additional disturbance, leading to the new disturbance $\tilde{d}_k$ for a system with nominal execution $\tilde{\sigma}_k \equiv 0$:

$$x_{k+1} = \underbrace{A(\sigma_k)x_k + d_k}_{\text{before: } \sigma_k,\, d_k} = \underbrace{A(0)x_k}_{\text{now: } \tilde{\sigma}_k = 0,} + \underbrace{A(\sigma_k)x_k - A(0)x_k + d_k}_{\text{disturbance } \tilde{d}_k}. \tag{230}$$

Therefore, σ_k and d_k in the original system can be equivalently replaced by $\tilde{\sigma}_k = 0$ and $\tilde{d}_k$. Inserting this into the old abstraction leads to

$$v_{k+1} \leq \rho_0 v_k + \beta_0 |\tilde{d}_k|. \tag{231}$$

Herein,

$$|\tilde{d}_k| \overset{(230),(29c),(33)}{\leq} \underbrace{\|A(\sigma_k) - A(0)\|_2}_{\gamma_{\sigma_k}} \underbrace{|x_k|}_{\leq v_k} + |d_k|. \tag{232}$$

Inserting this into the previous equation yields the upper bound

$$v_{k+1} \overset{(231),(232)}{\leq} \underbrace{(\rho_0 + \beta_0 \gamma_{\sigma_k})}_{\rho_{\sigma_k}} v_k + \beta_0 |d_k| \tag{233}$$

that can be interpreted as a new abstraction. □

6.5.2 Abstraction From Lyapunov Functions

The previous abstraction treats any deviation from the nominal case as disturbance, which may cause pessimism. As an extreme example, consider the case of $A(0) \neq 0, A(1) = 0$, i.e., the state jumps to zero immediately for the non-nominal execution $\sigma_k = 1$. Then, as detailed in the previous derivation, this behavior is abstracted as the worst behavior possible from any disturbance with magnitude $|d_k| \leq |A(1)x_k - A(0)x_k|$, i.e., as possibly increasing the state instead of actually zeroing it.

An improved abstraction which represents the non-nominal case with better accuracy is possible by considering a quadratically bounded Lyapunov function $V(x)$ for the nominal case and abstracting the state space by level sets $V(x) \leq \text{const}$. Here, the idea is specialized to the linear case and a quadratic Lyapunov function $V_P(x) = x^\top P x$.

It is important to note that $V_P(x)$ is only required to be decreasing in the nominal case without disturbance. As will be detailed later, it is no longer a Lyapunov function in general; typically it will increase at skipped events because these render the system "temporarily unstable".

6.5.2.1 Preliminaries

First, a number of bounds related to $V_P(x)$ are derived.

Definition 6.5.2 (Tight Bound)**.** The symbol $\trianglelefteq$ is used to mark an inequality as a tight bound, e.g., $|Ax| \trianglelefteq \|A\|_2 |x|$, where "tight" is defined as: For all values of the non-vector variables (scalars and matrices), there is a nontrivial choice of vector-valued variables such that the bound is met by equality. Here, nontrivial means that at least one of the vector variables is nonzero. Therefore, the previous example means $\forall A \in \mathbb{R}^{n\times n}\ \exists x \in \mathbb{R}^n \setminus \{0\} : |Ax| = \|A\|_2 |x|$. ◁

Lemma 6.5.3 (Spectral Norm as Tight Bound)**.** *For all $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$,*

$$|Ax| \trianglelefteq \|A\|_2 |x|. \tag{234}$$

Proof. This follows from the definition of the spectral norm (Definition 4.1.6). An explicit result for the value x that meets the bound by equality is given in the appendix in Lemma E.3. □

Lemma 6.5.4. *Analogous to the previous lemma,*

$$\sqrt{V_P(Ax)} \trianglelefteq \|A\|_P \sqrt{V_P(x)} \qquad \forall x \in \mathbb{R}^n \quad \forall A \in \mathbb{R}^{n\times n}. \tag{235}$$

Proof. Analogous the proof of Theorem 5.1.15, the P-ellipsoidal norm can be seen as a transformed version of the spectral norm, using a coordinate transformation to z with $x = (P^{1/2})^{-\mathsf{T}} z$, where $P^{1/2}$ denotes the Cholesky Decomposition $P = P^{1/2}(P^{1/2})^{\mathsf{T}}$ according to Theorem 5.1.3. Therefore,

$$\sqrt{V_P(Ax)} \overset{(121)}{=} |(P^{1/2})^{\mathsf{T}} Ax| = |(P^{1/2})^{\mathsf{T}} A (P^{1/2})^{-\mathsf{T}} z| \tag{236}$$

$$\overset{(234)}{\trianglelefteq} \|(P^{1/2})^{\mathsf{T}} A (P^{1/2})^{-\mathsf{T}}\|_2 |z| \tag{237}$$

$$\overset{\text{Theorem 5.1.15}}{=} \|A\|_P |z| = \|A\|_P |(P^{1/2})^{\mathsf{T}} x| \overset{(121)}{=} \|A\|_P \sqrt{V_P(x)}. \tag{238}$$

□

Lemma 6.5.5 (Subadditivity)**.**

$$\sqrt{V_P(x+y)} \trianglelefteq \sqrt{V_P(x)} + \sqrt{V_P(y)} \qquad \forall x, y \in \mathbb{R}^n. \tag{239}$$

Proof. As noted in the proof of Theorem 5.1.14, $\sqrt{V_P(x)}$ is a vector norm. Hence, the triangle inequality (29c) holds. To see that the bound is tight, set $y = x$ and use the homogeneity of norms (29b). □

Lemma 6.5.6 (Bounding $|x|$ by $\sqrt{V_P(x)}$)**.** *For all $x \in \mathbb{R}^n$ and $M \in \mathbb{R}^{a\times n}$, $M \neq 0$:*

$$\text{For } a = n: \quad \underbrace{\|(P^{1/2})^\top M\|_2^{-1}}_{=:c_1(P,M)} \sqrt{V_P(Mx)} \leq |x|. \tag{240}$$

$$\text{For any } a: \quad |Mx| \leq \underbrace{\|M(P^{1/2})^{-\top}\|_2}_{=:c_2(P,M)} \sqrt{V_P(x)}. \tag{241}$$

In particular, for $M = I$,

$$c_1(P,I)\sqrt{V_P(x)} \leq |x| \leq c_2(P,I)\sqrt{V_P(x)}. \tag{242}$$

Proof. This follows from rewriting and then applying the tight bound of Lemma 6.5.3:

$$\sqrt{V_P(Mx)} \overset{(121)}{=} |(P^{1/2})^\top Mx| \overset{(234)}{\leq} \|(P^{1/2})^\top M\|_2 \, |x| \quad \text{and} \tag{243}$$

$$|Mx| = |M(P^{1/2})^{-\top}(P^{1/2})^\top x| \overset{(234)}{\leq} \|M(P^{1/2})^{-\top}\|_2 \underbrace{|(P^{1/2})^\top x|}_{\overset{(121)}{=}\sqrt{V_P(x)}}. \tag{244}$$

The reciprocal $\|(P^{1/2})^\top M\|_2^{-1}$ in c_1 always exists: $M \neq 0$ is a condition of this lemma and $P^{1/2}$ is invertible due to the Cholesky Decomposition, so $(P^{1/2})^\top M \neq 0$. □

Remark 6.5.7 (Graphical Interpretation). Figure 37 shows an illustration of the above lemma for square M: Geometrically, $\sqrt{V_P(x)} = 1$ is an ellipsoid surface $\mathcal{E}$ and $|x| = r$ is a sphere $\mathcal{S}_r$ of radius r. For $M = I$ and $\sqrt{V_P(x)} = 1$, (242) implies $c_1(P,I) \leq |x| \leq c_2(P,I)$. These bounds are tight because the original bound in Lemma 6.5.6 is tight and the restriction to $\sqrt{V_P(x)} = 1$ only fixes the length but not the direction of x. Therefore, c_1 and c_2 are the radii of the inscribed and circumscribed spheres of the ellipsoid $\mathcal{E}$ (Figure 37, left).

For $M \neq I$, an additional transformation step is included: For $|x| = c_1(P,M)$, (240) shows that $\sqrt{V_P(Mx)} \leq 1$, so $c_1(P,M)$ is the radius of a sphere $\mathcal{S}_{c_1}$ such that $M\mathcal{S}_{c_1} = \{Mx \mid |x| = c_1\}$ is inscribed in $\mathcal{E}$ (Figure 37, center). For $\sqrt{V_P(x)} = 1$, (241) leads to $|Mx| \leq c_2(P,M)$, so c_2 is the radius of the circumscribed sphere of the transformed ellipsoid $M\mathcal{E} = \{Mx \mid x \in \mathcal{E}\}$ (Figure 37, right). ◁

6.5.2.2 Result

This leads to the main result:

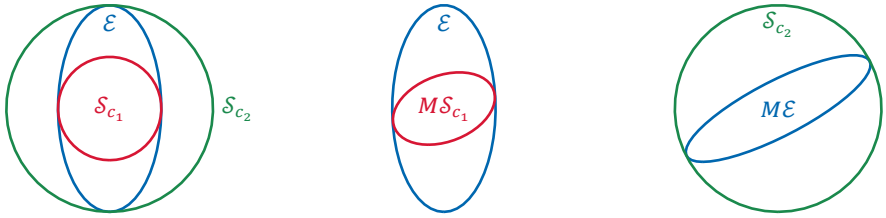


Figure 37: As detailed in Remark 6.5.7, Lemma 6.5.6 can be interpreted geometrically as inscribing or circumscribing ellipsoids $\mathcal{E} = \{x \mid \sqrt{V_P(x)} = 1\}$ and spheres $\mathcal{S}_r = \{x \mid |x| = r\}$ subject to a linear transformation M.

Theorem 6.5.8 (Computing the Abstraction)**.** *Let $P \succ 0$. If*

$$\rho_\sigma \geq \|A(\sigma)\|_P, \quad \alpha \geq c_2(P, \bar{C}_s)/c_1(P, C_0), \quad \beta_\sigma \geq c_2(P, \bar{C}_s)/c_1(P, G(\sigma)) \quad \forall \sigma \in \Sigma, \quad (245)$$

then the abstract system (216) *is a valid output convergence rate abstraction of the linear original system* (213)*. The terms $\|\cdot\|_P$, c_1 and c_2 are computed by Theorem 5.1.15 and Lemma 6.5.6.*

It is best to match the above "$\geq$" conditions by equality. However, as noted earlier, an upper approximation may be used instead. This is particularly useful if σ is not exactly known, for example if the timing deviation Δt_k can not be measured exactly.

Proof. Consider the evolution of the magnitude of the safety output:

$$|s_{k+1}| \overset{(213b)}{=} |\bar{C}_s x_{k+1}| \quad (246)$$
$$\overset{(241)}{\leq} c_2(P, \bar{C}_s)\sqrt{V_P(x_{k+1})} \quad (247)$$
$$\overset{(213a)}{=} c_2(P, \bar{C}_s)\sqrt{V_P(A(\sigma_k)x_k + G(\sigma_k)d_k)} \quad (248)$$
$$\overset{(239)}{\leq} c_2(P, \bar{C}_s)\sqrt{V_P(A(\sigma_k)x_k)} + c_2(P, \bar{C}_s)\sqrt{V_P(G(\sigma_k)d_k)} \quad (249)$$
$$\overset{(235)}{\leq} c_2(P, \bar{C}_s)\|A(\sigma_k)\|_P\sqrt{V_P(x_k)} + c_2(P, \bar{C}_s)\sqrt{V_P(G(\sigma_k)d_k)} \quad (250)$$
$$\overset{(240)}{\leq} c_2(P, \bar{C}_s)\|A(\sigma_k)\|_P\sqrt{V_P(x_k)} + c_2(P, \bar{C}_s)\, c_1^{-1}(P, G(\sigma_k))|d_k| \quad (251)$$

The abstract state v_k tracks an upper bound of $c_2(P, \bar{C}_s)\sqrt{V_P(x_k)}$ arising from this equation, which is shown by induction in the following:

Induction Assumption IA(k)

$$\mathrm{IA}(k) \quad :\Leftrightarrow \quad c_2(P,\bar{C}_\mathrm{s})\sqrt{V_P(x_k)} \leq v_k \tag{252}$$

Start of Induction

$$v_0 \overset{(216a)}{=} \alpha|\tilde{x}_0|_{\max} \overset{(213c)}{\geq} \alpha|\tilde{x}_0| \overset{(240)}{\geq} \alpha c_1(P,C_0)\sqrt{V_P(\underbrace{C_0\tilde{x}_0}_{\overset{(213c)}{=} x_0})} \tag{253}$$

$$\overset{(245)}{\geq} c_2(P,\bar{C}_\mathrm{s})\sqrt{V_P(x_0)} \Rightarrow \mathrm{IA}(0). \tag{254}$$

Induction Step Assume IA(k).

$$|\bar{C}_\mathrm{s}x_{k+1}| \overset{(251)}{\leq} c_2(P,\bar{C}_\mathrm{s})\|A(\sigma_k)\|_P\sqrt{V_P(x_k)} + c_2(P,\bar{C}_\mathrm{s})c_1^{-1}(P,G(\sigma_k))|d_k| \tag{255}$$

$$\overset{\mathrm{IA}(k)}{\leq} \underbrace{\|A(\sigma_k)\|_P}_{\overset{(245)}{\leq}\rho_{\sigma_k}} v_k + \underbrace{c_2(P,\bar{C}_\mathrm{s})c_1^{-1}(P,G(\sigma_k))}_{\overset{(245)}{\leq}\beta_{\sigma_k}} \underbrace{|d_k|}_{\overset{(213a)}{\leq}|d_k|_{\max}} \tag{256}$$

$$\leq \rho_{\sigma_k}v_k + \beta_{\sigma_k}|d_k|_{\max} \overset{(216b)}{=} v_{k+1} \tag{257}$$

$$\Rightarrow \mathrm{IA}(k+1) \tag{258}$$

Conclusion By induction, IA(k) holds for all $k \geq 0$. This proves the desired abstraction guarantee (217) since

$$|s_k| \overset{(213b)}{=} |\bar{C}_\mathrm{s}x_k| \overset{(241)}{\leq} c_2(P,\bar{C}_\mathrm{s})\sqrt{V_P(x_k)} \overset{\mathrm{IA}(k)}{\leq} v_k \quad \forall k \geq 0. \tag{259}$$

□

6.5.2.3 Discussion

The starting point for convergence rate abstractions given in Section 5.3 was a connection between quadratic Lyapunov functions and reachability analysis. The relation to these concepts is discussed in the following.

Geometric Interpretation Following the idea of Remark 6.5.7 shows that the output convergence rate abstraction is equivalent to set-valued reachability analysis using ellipsoid sets of a fixed shape given by P:

By IA(k), the abstract state v_k translates to an ellipsoid set

$$x_k \in \mathcal{E}_{v_k} \coloneqq \{x | \sqrt{V_P(x)} \leq v_k/c_2(P,\bar{C}_s)\}. \tag{260}$$

Let Ellipsoid($\mathcal{S}$) denote the smallest ellipsoid of this shape containing a set $\mathcal{S}$, i.e., Ellipsoid($\mathcal{S}$) $\coloneqq \mathcal{E}_c \supseteq \mathcal{S}$ with minimal c. The Minkowski sum of such ellipsoid sets is $\mathcal{E}_a \oplus \mathcal{E}_b = \mathcal{E}_{a+b}$, i.e., it is again an ellipsoid set of the same shape. This can be shown using the support function representation (Guernic and Girard, 2010); the proof is omitted here for the sake of brevity.

This shows that the sums in the above proof correspond to Minkowski sums of ellipsoid sets. Therefore, (247) to (251) can be interpreted as the set-valued upper approximation

$$\mathcal{X}_{k+1} = \mathcal{E}_{v_{k+1}} = \text{Ellipsoid}(A(\sigma_k)\mathcal{X}_k) \oplus \text{Ellipsoid}(G(\sigma_k)\mathcal{D}_k) \tag{261}$$

of the state dynamics $x_{k+1} = A(\sigma_k)x_k + G(\sigma_k)d_k$, where $\mathcal{X}_k$ and $\mathcal{D}_k$ are the sets of state and disturbance. In other words, this abstraction is an *ellipsoid abstract domain* as described by Roux et al., 2012.

Connection to Lyapunov Functions Consider an abstraction resulting from Theorem 6.5.8. For the stability of this abstraction in the nominal case, it is required that $\|A(0)\|_P < 1$. This equivalently means that $V_P(x)$ is a discrete-time Lyapunov function for the *nominal case* ($\sigma \equiv 0, d \equiv 0$): V_P is positive definite and decreases per

$$V_P(x_{k+1}) = V_P(A(0)x_k) \overset{(235)}{\leq} \underbrace{\|A(0)\|_P^2}_{<1} V_P(x_k) \prec V_P(x_k). \tag{262}$$

However, the same does not necessarily hold for weakly-hard execution, e.g., skipped executions $\sigma_k = 1$: If $\|A(1)\|_P > 1$, then $V_P(x)$ may increase on $\sigma_k = 1$, so it is no longer a Lyapunov function for the *weakly-hard* system. Nevertheless, as will be discussed in the next sections, stability can still be shown if the abstraction (or equivalently, $V_P(x)$) is decreasing on average, i.e., in the long term. Then, $V_P(x)$ is a *Lyapunov-like function* for the weakly-hard system, which decreases in the long term but may temporarily increase, similar to the definition in Ye et al., 1996, Theorem 4.2.

6.6 Design-Time Stability Analysis

To illustrate a first use of abstractions, this section considers the stability (DGES) analysis of systems with weakly-hard execution, assuming zero disturbance. For simplicity, $s_k = x_k$ is assumed. Note that by Theorem 4.1.13, DGES is equivalent to CGES for a real-time control system as given in Section 4.1.4.6, and analogous arguments hold in the general nonlinear case under appropriate technical conditions.

System Model This section considers a disturbance-free but possibly nonlinear system model

$$x_{k+1} = f_{\sigma_k}(x_k), \quad x_0 \in \mathbb{R}^n. \tag{263a}$$

with a corresponding abstraction

$$v_{k+1} = \rho_{\sigma_k} v_k, \quad v_0 = \alpha|x_0|. \tag{263b}$$

that guarantees

$$|x_k| \leq v_k \tag{263c}$$

for any initial state x_0 and execution sequence $(\sigma_0, \sigma_1, \dots)$.

General Case Stability can be shown if the execution is good enough "on average":

Theorem 6.6.1 (Abstracted Stability Criterion for Weak Execution)**.** *The system* (263a) *with abstraction parameters ρ_σ and α is DGES($\tilde{\rho}, \tilde{\alpha}\alpha$) if the execution sequence σ_k satisfies*

$$\forall k \geq 0 \quad \kappa_k \leq \tilde{\alpha}\tilde{\rho}^k \tag{264}$$

with $\tilde{\rho} \in (0; 1)$ and $\tilde{\alpha} \geq 1$, where

$$\kappa_k \coloneqq \frac{v_k}{v_0} = \prod_{i=0}^{k-1} \rho_{\sigma_i}. \tag{265}$$

Proof. Assume $\tilde{\rho} \in [0; 1)$ and $d \equiv 0$.

$$(264) \Rightarrow \left(\forall k \geq 0, x_0 \in \mathbb{R}^n : \quad |x_k| \overset{(263c)}{\leq} v_k \overset{(265)}{=} \kappa_k v_0 \overset{(264),\,(263b)}{\leq} \tilde{\alpha}\tilde{\rho}^k \alpha |x_0| \right)$$
$$\Rightarrow (263a) \text{ is DGES}(\tilde{\rho}, \tilde{\alpha}\alpha) \text{ by Definition 4.1.12.} \tag{266}$$

□

(M, K)-Weak Execution Now, the general idea of abstractions is applied to the case of (M, K)-weak execution of the whole control system, which is considered in most work on weakly-hard control systems suffering from packet loss or deadline misses. There are only two execution modes ($\Sigma = \{0, 1\}$): Either, everything is executed ideally ($\sigma_k = 0$) or all tasks are skipped, here abbreviated as $\sigma_k = 1$. (In the original system model of Section 4.1.4.6, this skipping was denoted by $\Delta t_k = 0, \sigma_{\text{skip},k} = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^\top$.)

To recall the definition of (M, K)-weak execution (Definition 3.1.1): In any K consecutive control periods, at least M controller executions are executed normally ($\sigma_i = 0$), while the remaining up to $\overline{m} \coloneqq K - M$ controller executions are skipped ($\sigma_i = 1$). Formally,

$$\sigma_i \in \{0, 1\}, \tag{267a}$$

$$\forall k \geq 0 : \quad \sum_{i=k}^{k+K-1} \sigma_i \leq \overline{m} \coloneqq K - M. \tag{267b}$$

Theorem 6.6.2 (Exponential Stability Criterion for (M, K)-weak Execution)**.** *The system* (263) *is DGES($\tilde{\rho}$, $\tilde{\alpha}\alpha$) under* (M, K)*-execution* (267) *if* $0 < \rho_0 \leq \rho_1$ *and* $\tilde{\rho} < 1$, *where*

$$\tilde{\rho} \coloneqq \rho_0^{\frac{M}{K}} \rho_1^{\frac{K-M}{K}}, \quad \tilde{\alpha} \coloneqq \left(\frac{\rho_1}{\rho_0} \right)^{K-M}. \tag{268}$$

Proof. The proof is detached to Appendix D. The key idea is to derive a bound on κ_k from Theorem 6.6.1 using the maximum ratio of skips, $(K - M)/K$, and the maximum number of consecutive skips, $K - M$. □

Discussion Theorem 6.6.2 exemplifies the benefit of convergence rate abstractions: Stability can be shown without the intricate computation of an explicit stability certificate for the *weakly-hard* system, such as a Lyapunov function or reachable set. Instead, the criterion only requires the execution parameters (M, K) and a simple abstraction summarizing the stability and

robustness of the *nominal* system. The detailed system dynamics are not required, since they are abstracted by three scalar parameters ρ_0, α and ρ_1, which model exponential decay, initial overshoot and the sensitivity to skipping the controller execution. Determining these is possible merely from the exponential stability of the nominal case (Section 6.5.1) or, optionally, via Lyapunov functions for the *nominal* case (Section 6.5.2). Both methods are considerably easier than directly analyzing the weakly-hard case.

Note that the stability condition of the criterion only depends on the skip ratio $(K - M)/K$ but not on the absolute number of skips K: Increasing K and M proportionally increases the overshoot $\tilde{\alpha}$, but not the growth rate $\tilde{\rho}$. Therefore, the criterion shows stability for (M, K)-weak execution if and only if it shows stability for (cM, cK)-weak execution, where $c \in \mathbb{N}_1$ is an arbitrary integer. This will be reconsidered later.

The generality of the proposed convergence rate abstractions can be seen by the fact that they contain existing results as a special case. For example, in Horssen et al., 2016, Theorem 2, an LMI equivalent to the quadratic Lyapunov-like function $V(x) = x^\top P x$ is used to show stability if, in the terms used here[1], $\rho_1^{K-M}\rho_0^M < 1$, which is equivalent to $\tilde{\rho}^K < 1$ and to $\tilde{\rho} < 1$. In that context, the criterion is shown to be conservative, which leads to the following question:

Existence of Stable Abstractions To investigate if the approach is pessimistic, it is important to know if there is a converse variant of Theorem 6.6.2: If a system is stable under weak execution, does this imply the existence of a stable abstraction? For a one-dimensional linear abstraction as considered in this work, the answer is no, as shown in the following. Therefore, this abstraction carries some pessimism.

False Conjecture 6.6.3 (Converse Stability Criterion for (M, K)-weak Execution)**.** *Consider a linear system* (263a) *without disturbance. One could expect that if the system is exponentially stable (DGES) under* (M, K)*-weak execution, then there is a valid abstraction* (263b) – (263c) *that proves exponential stability by Theorem 6.6.2, i.e., the abstraction constants* ρ_0, ρ_1 *satisfy*

$$\tilde{\rho} = \rho_0^{\frac{M}{K}} \rho_1^{\frac{K-M}{K}} < 1. \tag{269}$$

[1] Note that here, $\sigma = 0$ means nominal execution and $\sigma = 1$ means skipping. In Horssen et al., 2016, it is the opposite.

This would be helpful as it would imply that Theorem 6.6.2 is sufficient and necessary, so that stability under (M, K)-execution is *always* equivalent to stability under (cM, cK)-execution for $c \in \mathbb{N}_1$. However, the following academic counterexample will show that this is not generally true, which also matches the conservatism stated in Horssen et al., 2016, Theorem 2.

Therefore, at least this simple variant of an abstraction is conservative. Possible improvements to reduce this pessimism will be discussed later.

Counterexample to False Conjecture 6.6.3. The system

$$A(0) = \begin{bmatrix} a & 0 & a \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad A(1) = \begin{bmatrix} a & 0 & 0 \\ c & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad a = \frac{1}{2}, \quad c = 1000 \tag{270}$$

is stable under (M, K)-execution for $M = 1, K = 2$, but not for $M = 2, K = 4$: As detailed in Gaukler, Rheinfels, et al., 2019, pp. 24 ff., this follows from $\|A(0)A(1)\|_2 < 1$, $\|A(0)\|_2 < 1$ and $\rho\{A(1)^2 A(0)^2\} > 1$. □

6.7 Related Work and Contribution

The idea of abstracting a system to an upper bound of its state radius is implicitly contained in the notion of exponential stability, so it has existed in hidden form for centuries. In the following, select examples from the literature are pointed out in which the idea appeared more explicitly.

Radial Growth Bounds As discussed earlier, a one-dimensional inequality for the state radius is used in Huang et al., 2019 to analyze stability of a nonlinear real-time control system under weak execution. This reduces the stability test to a system of linear inequalities. The test does not require explicit system dynamics, but only requires bounds on the nominal stability and Lipschitz constant.

A one-dimensional growth and perturbation bound for the open-loop plant is termed *incremental forward completeness* in Zamani et al., 2012. There, it is used to guarantee the correctness of a discrete-state controller determined from a state-space discretization of the plant. In contrast to the abstractions in this chapter, the convergence rate of Zamani et al., 2012 only abstracts the open-loop plant, so that an unstable plant will result in an unstable convergence

rate even if the closed loop is stable. However, it should be possible to transfer the results given in terms of Input-to-State Lyapunov functions to closed-loop analysis. A similar but full-dimensional convergence rate is used in Reissig et al., 2017, Section VIII.C.

Generalized Growth Bounds In Bund, 2017, Chapters 6.1 and 6.2.2, networked control systems with packet loss are modeled in an abstract *interval domain* by means of a magnitude impulse response. To some extent, this corresponds to the impulse response of a convergence rate abstraction. However, Bund, 2017 employs an explicit formulation of the linear dynamics to generate this impulse response, resulting in a non-exponentially decaying response, which corresponds to an abstract system of order higher than one. The concept is extended to *signal densities* that describe the temporal distribution of signals, e.g., short spikes vs. persistent disturbance.

The idea of abstracting the state space to certain sets, especially ellipsoids, is widely used for the formal verification of computer programs and dynamical systems (Roux et al., 2012). In some sense, a convergence rate abstraction results in a radial abstraction, as it yields spherical sets $|x_k| \leq \text{const}$. If the Lyapunov function is a quadratic form, as in Section 6.5.2, then the convergence rate abstraction resembles an ellipsoid abstraction (cf. Section 6.5.2.3).

A similar ellipsoid abstraction appears in some variants of tube model predictive control (Cannon et al., 2011, Section III-B) to describe the tube of possible disturbed trajectories $x(t)$ around the disturbance-free nominal trajectory $\tilde{x}(t)$. This tube

$$(x(t) - \tilde{x}(t))^\top Q\,(x(t) - \tilde{x}(t)) \leq r^2(t) \tag{271}$$

has ellipsoid cross section with fixed shape matrix $Q \in \mathbb{R}^{n\times n}$ but varying size $r(t) \geq 0$. Therefore, the dynamics of $r(t)$ are a one-dimensional abstraction of the influence of disturbance. Raković et al., 2016 present a similar approach using polytopes as the tube shape, although then the dynamics of the tube can only be modeled implicitly as a set of inequalities.

For the case of uncertain measurements, *set-valued observation*, such as interval observers, provides a guaranteed set-valued bound on the state. While this approach also models the evolution of the state set, the description is high-dimensional and typically implicit (Blanchini, 2008, Chapter 10).

Comparison Theory More generally, the concept of stability analysis by reduction to a simple dynamical system is formalized by *comparison theory* (Michel et al., 2001). Convergence rate abstractions (Definition 6.2.2) can be interpreted as an application of that theory (e.g., Michel et al., 2001, Theorem 3.4.1 and Proposition 4.1.3): Consider the system

$$x_{k+1} = f(x_k, d_k), \quad k \geq 0 \tag{272}$$

and a mapping $V(x)$ from $\mathbb{R}^n$ to $\mathbb{R}_{\geq 0}$, similar to a Lyapunov function. Assume there is a bound c such that

$$0 \leq |x| \leq V(x) \leq c|x| \qquad \forall x \tag{273}$$

and a function $\tilde{f}$ that bounds the evolution of the Lyapunov function per

$$0 \leq V(f(x_k, d_k)) \leq \tilde{f}(V(x_k), d_k) \qquad \forall x_k, d_k. \tag{274}$$

Then, a one-dimensional *comparison system* is given by the inequality

$$0 \leq v_{k+1} \leq \tilde{f}(v_k, d_k), \quad v_0 = V(x_0). \tag{275}$$

This nondeterministic v_k-system generates all trajectories for $V(x_k)$ that are possible according to the bound (274). Consequently, every trajectory for $V(x_k)$ resulting from the actual x_k-dynamics are contained in the set of trajectories of v_k. Therefore, if the v_k-system is stable (respectively, bounded), then $V(x_k)$ converges (is bounded) and by (273) the same holds for x_k.

In summary, stability (respectively, boundedness) of the comparison system (275) implies stability (boundedness) of the original system (272). The converse is not necessarily true because v_k may grow faster than $V(x_k)$ whenever the upper bound of (274) is pessimistic.

In general, the stability of a comparison system is sufficient but not necessary for the stability of the original system. In the special case in which both are equivalent, $V(x)$ is a *stability-preserving mapping* (Michel et al., 2001). While Michel et al., 2001 provides theoretical constructions for such mappings, these remain impractical because the state-space realization of the comparison system would in general have the same complexity as the original system.

Discussion For most existing techniques, the method for analyzing the dynamics (e.g., based on robust stability or Lyapunov function synthesis) is hard-coded in the weakly-hard stability analysis (e.g., exponential stability under (M, K)-execution, state bound under disturbance, dynamic resource

management). While, as in computer programming, hard-coding may allow for some benefit, it hurts reuse and understanding. For example, a LMI-based approach to dynamic resource management for linear systems may be impossible to adapt to nonlinear systems because LMI methods are typically restricted to linear systems. In contrast, abstractions provide a clean interface that facilitates reuse and understanding, though with the drawback of some pessimism. Therefore, they can be applied to a variety of scenarios, such as skipped executions or varying timing, as will be exemplified in the next chapter.

6.8 Concluding Remarks

The stability analysis of weakly-hard real-time control systems is significantly more complex than for the classical hard real-time case. In this chapter, *convergence rate abstractions* were introduced as a method for reducing said complexity: The discretized real-time control system is characterized by a one-dimensional model of the worst-case behavior. This time-varying bound aims to be a useful compromise between the pessimism inherent to stability analysis for fixed timing bounds and the complexity associated with analyzing the original model for weakly-hard execution.

The key distinction between convergence rate abstractions and many related concepts from prior work (cf. Section 6.7) is that the simplified model is considered as an independent dynamical system. As a result of this independence, stability analysis of weakly-hard execution does no longer depend on the original system and general results can be developed for any system, both linear and nonlinear, that admits a convergence rate abstraction. The existence of this abstraction was proven and implemented for linear dynamics. The nonlinear case was not discussed here but can be obtained from an appropriate generalization. Once the abstraction has been determined, it can be used as a general interface from which various analysis results are derived:

A first application was shown for the common case of (M, K)-weak execution for a monolithic controller, i.e., either all or no control tasks are executed in one period. Here, stability can be verified at design time. Because computing the one-dimensional abstract dynamics is computationally cheap, the idea can also be applied at run-time for provably safe dynamic resource management. In the next chapter, this will be explored and illustrated by experiments.

Limitations The theoretical discussion indicates that the simplicity of the abstractions comes with the drawback of pessimism. This will be confirmed by experiments in the next chapter. A practical application could profit from abstractions that are more accurate, even if their complexity is slightly increased. Possibilities for such improvements are discussed in the following.

Improved Abstractions One path to improvement is to reuse techniques for the stability analysis of switched discrete-time systems (cf. Section 2.5.2.1). Geometrically, the technique of Section 6.5.2 based on a quadratic Lyapunov function is conservative because it uses state sets of a fixed, ellipsoid shape. A possible improvement is to use a *piecewise* ellipsoid set, equivalent to a piecewise quadratic Lyapunov function. Alternatively, one can vary the shape of the ellipsoid depending on the skip sequence, corresponding to a parameter-dependent quadratic Lyapunov function as in Horssen et al., 2016, Section V.B. The framework of path-complete Lyapunov functions (Philippe et al., 2019) provides connections between both variants. Further research could translate these techniques to convergence rate abstractions, e.g., by having ρ_{σ_k} also depend on σ_{k-1}.

Multi-Dimensional Generalization To improve the precision of abstractions, the abstract state v_k could be higher-dimensional, analogous to the fact that set-valued reachability analysis typically uses a set representation with a large number of parameters. This would require a general relation $s_k \in \mathcal{S}(v_k) \subseteq \mathbb{R}^n$ instead of $|s_k| \leq v_k$. One possible approach is that the m components of v_k represent the m slowest eigenmovements of the system. This leads to the question of model order reduction with guaranteed error bounds, for which the technique of Tran et al., 2017 could be extended to varying timing. Furthermore, the formalism for comparison systems by Michel et al., 2001 is a promising candidate to generalize the abstraction-based stability criteria to the higher-dimensional case.

Extensions Another relevant question is an abstraction of nonlinear systems, which in principle is possible by the same Lyapunov methods. This is sketched in Gaukler, Rheinfels, et al., 2019, Section 6. From a practical perspective, one could also sidestep the theoretic part and determine the abstraction parameters by black-box optimization, as will be sketched in an experiment in the next chapter (page 186). However, this typically only allows for stochastic guarantees.

7 Dynamic Resource Management

In this chapter, the development towards flexible timing is continued further. Chapter 5 dismissed the traditional requirement of perfect IO timing and provided stability analysis for time windows. This came with the restriction that the timing bounds are equal in every period, which was resolved in Chapter 6 by convergence rate abstractions, a generic framework for safety analysis under time-varying execution. The framework was first applied to design-time stability analysis with fixed (M, K) skip rates in a simplified setting. In this chapter, convergence rate abstractions will be used to implement dynamic resource management: As envisioned in Section 3.2.4, a run-time mechanism *dynamically* relaxes the input/output timing requirements of a controller as far as possible while guaranteeing a physical safety specification.

As discussed at the start of Chapter 6, the baseline for comparison is design-time stability analysis. Design-time analysis is restricted to fixed timing parameters, e.g., constant time windows or (M, K) skip rates. Therefore, it can not go beyond the stability limit, i.e., the worst parameter value that may occur *permanently*. On the other hand, it has negligible run-time overhead.

To be beneficial, dynamic resource management must allow for more timing flexibility than this baseline and still have little computational overhead. The desired timing flexibility is enabled by two factors: First, the inertia of the system allows to temporarily exceed the stability limit, as long as the system is stable on average (cf. Theorem 6.6.1). Second, the actual timing within the timing bound will rarely be the worst case, assuming a typical distribution of execution times (cf. Figure 4 on page 21). Therefore, it is possible to permit larger time windows most of the time.

The approach presented in this chapter uses convergence rate abstractions to represent these factors: First, the abstraction dynamics correspond to the system inertia. Second, the actual timing can be incorporated by computing the abstraction state at run-time. In contrast to other techniques, the approach supports control systems with multiple sensors and actuators that have individual time windows or can be skipped individually. Despite the large number of timing variables in such systems, it is possible to derive simple but safe decision criteria and therefore keep the run-time overhead low. However, it will be seen that the simplifications may incur some pessimism.

The contents of this chapter are the following: Sections 7.1 and 7.2 present the notation and a formal problem statement. Then, a simple scheme for dynamic resource management with strict safety guarantees is introduced in Section 7.3 and formally proven in Section 7.4. Simulative experiments in Section 7.5 demonstrate that real-time control applications may be run with relaxed timing requirements without giving up safety. Section 7.6 summarizes the results and discusses open issues.

A preliminary version of this chapter has appeared as an internal technical report (Gaukler, 2020b). Furthermore, the high-level idea of a dynamic model of the "quality of control" was sketched earlier in Ulbrich and Gaukler, 2019.

7.1 Notation

If not mentioned differently, i, j, k, N are integer variables, where $\forall k$ abbreviates $\forall k \in \mathbb{N}_0$. All uppercase variables are matrices of appropriate dimension, with the exception of the counter N and the scalar sampling period T.

7.2 Problem Statement

This chapter addresses dynamic resource management (Section 3.2.4) for a real-time control system affected by disturbance, IO timing deviations and skipped executions of controller, sensors and actuators (Section 3.1).

Goal The goal is to allow as wide timing deviations and as many skipped executions as possible while satisfying a safety specification $|s_k| \leq |s|_{\max}$. For optimal decisions, the permissible execution should be determined at run-time but with little computational overhead. This is detailed in the following.

Assumptions To obtain a discrete-time model and abstraction, this section inherits the assumptions of Section 4.1.4.6 and Chapter 6. Beyond, the disturbance bound is assumed as constant, i.e.,

$$|d_k|_{\max} \equiv |d|_{\max}. \tag{276}$$

Furthermore, modeling all scheduling and run-time aspects is beyond the scope of this work. Therefore, the scheduler is mainly treated as a black box. For the simulative experiments, a stochastic description of the scheduler is used: If an event is allowed to be skipped, the skip will occur with a given

probability. If the skip does not occur, the timing is randomly drawn from a probability distribution and then limited to the maximum range that is currently permitted by the resource management policy. Further details will later be given in the description of the experiments.

7.3 Approach

In this section, the key concepts of the approach are introduced.

7.3.1 Output Convergence Rate Abstraction

First, an abstraction is determined by the following steps:

1. Use the discretized system model of Section 4.1.4.6.
2. Compute a Lyapunov matrix P for small timing deviations by Section 5.1.
3. By Section 6.5.2, determine an output convergence rate abstraction (Definition 6.2.2) from this Lyapunov function. The implementation of this step is detailed later (Section 7.5).

A conceptual advantage of abstractions is that the design of the real-time system must no longer consider the original system model but only the abstraction. It is therefore possible to treat the result of the above computation steps as a black box, in particular for readers whose focus is on the real-time system aspects. For the same reason, future work can extend or improve the methods for determining the abstraction parameters but reuse the rest of this chapter.

As will be shown in the next sections, verification is significantly easier than for the original system, because the one-dimensional and positive dynamics of the abstraction exclude most problematic cases occurring in the high-dimensional original system model, such as overshoot. In particular, it turns out that looking forward only one time step is sufficient for safety, while in general a larger prediction horizon must be considered for the original system to account for overshoot, as discussed in Section 3.2.4.

7.3.2 Resource Management Policy

The simple abstract system description can now be used to provide both safety and timing flexibility by dynamically adapting the timing bounds. It turns out that a simple condition is sufficient for safety: The scheduling parameter σ can be chosen arbitrarily as long as the abstraction fulfills the

abstract specification $v_k \leq |s|_{\max}$. By Theorem 6.2.3, this condition implies the desired safety specification $|s|_k \leq |s|_{\max}$. It is not necessary to make further restrictions, e.g., to prescribe a specific implementation of scheduling or resource management. In general notation, this results in the following scheme run at the beginning $t = kT + T/2$ of every period k:

1. Start with known abstraction state v_k (or an upper bound). Determine the range $\hat{\Sigma}_k \subseteq \Sigma$ of permitted execution σ_k in the upcoming period k such that $v_{k+1} \leq |s|_{\max}$ for any $\sigma_k \in \hat{\Sigma}_k$. As will be proven later, it is sufficient to check only v_{k+1} and not look further ahead, e.g., to v_{k+2}.
2. Schedule and execute the period k. This can be implemented arbitrarily as long as the limitation $\sigma_k \in \hat{\Sigma}_k$ is satisfied.
3. Update the abstract state v_{k+1} according to the actual execution σ_k of the last period k. Typically, the execution will be better than the permitted worst case, particularly if the worst-case execution time occurs rarely.
4. Increment k and repeat.

To simplify formal analysis, this can be equivalently restated as an arbitrary choice of σ_k that can be overridden by a safety mechanism:

Definition 7.3.1 (Safety-Net Policy)**.** A method for choosing σ_k is a *safety-net policy* if and only if it matches the following scheme:

1. A desired execution $\hat{\sigma}_k$ is chosen. There is no restriction on this choice.
2. If this choice would immediately violate the abstract safety condition $v_{k+1} \leq |s|_{\max}$, the safety mode is used instead ($\sigma_k = 0$). Otherwise, the chosen execution is used ($\sigma_k = \hat{\sigma}_k$):

$$\sigma_k = \begin{cases} \hat{\sigma}_k, & \hat{v}_{k+1}(\hat{\sigma}_k) \leq |s|_{\max}, \\ 0, & \text{else}, \end{cases} \tag{277a}$$

   where

$$\hat{v}_{k+1}(\hat{\sigma}_k) \coloneqq \rho_{\hat{\sigma}_k} v_k + \beta_{\hat{\sigma}_k} |d|_{\max} \tag{277b}$$

   is the abstract state that would result from the chosen $\hat{\sigma}_k$. Herein, v_k is the current abstract state computed from the abstract dynamics

$$v_0 = \alpha |\tilde{x}_0|_{\max}, \tag{277c}$$
$$v_{k+1} = \rho_{\sigma_k} v_k + \beta_{\sigma_k} |d|_{\max} \tag{277d}$$

   or an upper bound thereof. ◁

First, a number of variants are discussed. A formal safety proof is given later.

7.3.3 Notes on the Resource Management Policy

In its simplest variant, the scheme of Definition 7.3.1 is used for switching between two modes, a safety mode $\sigma = 0$ with deterministic execution and an optimistic mode $\sigma \neq 0$ with large timing bounds. This implements the idea of a "safety net" presented by Ulbrich and Gaukler, 2019, where a system normally runs under weakened timing bounds and is switched to strict timing bounds only if required to ensure safety guarantees.

This safety net is inspired by the Simplex architecture of Seto, Krogh, et al., 1998, whose idea is explained by the following analogy: Consider a learner driver supervised by a driving instructor. As long as the action $\hat{\sigma}_k$ intended by the learner is safe, he has control of the vehicle, i.e., $\sigma_k = \hat{\sigma}_k$. If the decision would lead to a crash, the instructor intervenes immediately, i.e., $\sigma_k \neq \hat{\sigma}_k$.

In the safety-net policy (277), it is enough to intervene just one time step before a possible violation $\hat{v}_{k+1} > |s|_{\max}$, while in reality an instructor must make predictions over a longer time horizon to have enough braking distance. This included prediction is a favorable property of the proposed convergence rate abstractions: Loosely interpreted, they do not simply measure the distance to violating the specification (distance to neighboring cars) but the distance to the "point of no return" at which a future violation would become unavoidable.

Uncertain Execution The formal notation suggests that the desired execution decision $\hat{\sigma}_k$ must be *deterministically* known before the start of the period, which may be restrictive in practice. However, as noted before, it is equivalently possible to limit the execution decision $\hat{\sigma}_k$ (marking tasks as optional or widening deadlines) to a *set* of decisions satisfying $\hat{v}_{k+1}(\hat{\sigma}_k) \leq |s|_{\max}$. Due to the simplicity of the abstraction dynamics, it is also possible to move this check to design-time and derive planning rules, e.g., "sensor 3 is optional if $v_k < 4$" or "extend all IO deadlines by 1 ms if $v_k < 10$".

Uncertain Execution in the Safe Mode The safe mode $\sigma = 0$ need not be exactly scheduled. The implementation of $\sigma = 0$ may be defined arbitrarily as long as the theoretical requirements are fulfilled: If $\sigma = 0$ is defined as a "set" of possible execution results and ρ_0, β_0 are defined as the worst ρ and β occurring in this set, the theoretical results hold unchanged. In practice, however, this set should be small to avoid too large values of ρ_0. As will be shown later, increased ρ_0 increases the best $|s|_{\max}$ that can be guaranteed.

Varying Disturbance The abstraction dynamics always assume the worst disturbance. Future work should consider safely detecting the actual disturbance amplitude to reduce the pessimism in times of low disturbance.

7.4 Safety Proof

Now, safety of the proposed scheme is proven. The following derivation holds for any safety-net policy (Definition 7.3.1), i.e., any scheme that falls back to a safety mode $\sigma = 0$ if the desired execution for the next period would not meet the safety requirement $v_{k+1} \leq |s|_{\max}$. Therefore the proofs are valid for a large class of dynamic resource management schemes.

7.4.1 Feasibility Assumptions

Two intuitive assumptions on the abstract system are required for the following derivations:

Assumption 7.4.1. *The safety mode $\sigma \equiv 0$ is stable:*

$$0 < \rho_0 < 1. \tag{278}$$

Assumption 7.4.2 (Feasibility)**.** *It is safe to always use the safety mode:*

$$\sigma_k \equiv 0 \quad \Rightarrow \quad v_k \leq |s|_{\max} \quad \forall k. \tag{279}$$

The advantage of one-dimensional abstractions is that, given the parameters of the abstraction, these assumptions are easy to test:

Lemma 7.4.3. *Under Assumption 7.4.1, Assumption 7.4.2 is equivalent to*

$$|s|_{\max} \geq \max\left\{v_0, \frac{\beta_0 |d|_{\max}}{1-\rho_0}\right\}. \tag{280}$$

Proof. Assume $\sigma_k \equiv 0$ and that Assumption 7.4.1 holds. Then, the abstraction (216b) becomes the one-dimensional linear time-invariant system

$$v_{k+1} = \rho_0 v_k + \beta_0 |d|_{\max} \tag{281}$$

with known initial state v_0 and constant input $\beta_0|d|_{\max}$. The remainder of this proof follows from basic linear systems theory.

As $|\rho_0| < 1$, the system is stable and converges to the steady state v_∞ that can be determined by setting $v_k = v_{k+1} = v_\infty$:

$$\begin{aligned} (281)\big|_{v_k = v_{k+1} = v_\infty} \quad &\Rightarrow \quad v_\infty = \rho_0 v_\infty + \beta_0 |d|_{\max} \\ &\Rightarrow \quad v_\infty = \frac{\beta_0 |d|_{\max}}{1 - \rho_0}. \end{aligned} \tag{282}$$

The difference between v_k and the steady state decays exponentially:

$$v_{k+1} - v_\infty \overset{(281)}{=} \rho_0 v_k + \beta_0 |d|_{\max} - v_\infty \tag{283}$$

$$= \rho_0 (v_k - v_\infty) + \rho_0 v_\infty + \beta_0 |d|_{\max} - v_\infty \tag{284}$$

$$\overset{(282)}{=} \rho_0 (v_k - v_\infty) + \underbrace{\rho_0 \frac{\beta_0 |d|_{\max}}{1 - \rho_0} + \beta_0 |d|_{\max} - \frac{\beta_0 |d|_{\max}}{1 - \rho_0}}_{0} \tag{285}$$

$$\Rightarrow \quad v_k - v_\infty = \rho_0^k (v_0 - v_\infty). \tag{286}$$

By Assumption 7.4.1, $0 < \rho_0 < 1$, so $v_k - v_\infty$ monotonically converges to zero. Therefore, v_k is contained within v_0 and v_∞:

$$\Rightarrow \quad v_k - v_\infty \in \begin{cases} [0;\ v_0 - v_\infty], & v_0 > v_\infty, \\ [v_0 - v_\infty;\ 0], & \text{else} \end{cases} \tag{287}$$

$$\Rightarrow \quad v_k \in \begin{cases} [v_\infty;\ v_0], & v_0 > v_\infty, \\ [v_0;\ v_\infty], & \text{else} \end{cases} \tag{288}$$

Both of these bounds are tight in the sense that $v_k = v_0$ is reached for $k = 0$ and $v_k = v_\infty$ is reached asymptotically for $k \to \infty$. Consequently,

$$\sup_k v_k = \max\{v_0, v_\infty\}, \tag{289}$$

which proves the original claim:

$$v_k \leq |s|_{\max} \ \forall k \quad \Leftrightarrow \quad |s|_{\max} \geq \sup_k v_k \overset{(289)}{\Leftrightarrow} |s|_{\max} \geq \max\{v_0, v_\infty\}. \tag{290}$$

□

Lemma 7.4.4 (Recursive Feasibility)**.** *Let Assumptions 7.4.1 and 7.4.2 hold. If* $v_k \leq |s|_{\max}$ *holds at the current time* k*, then it is a safe strategy to switch to the safety mode and stay in it forever:*

$$\Big(v_k \leq |s|_{\max} \quad \Rightarrow \quad \Big(v_{k+i}\big|_{\sigma_{k,k+1,\dots}=0} \leq |s|_{\max} \quad \forall i\Big)\Big) \quad \forall \sigma_{0,1,\dots,k-1} \in \Sigma \quad \forall k. \tag{291}$$

This result can be interpreted as a *recursive feasibility condition* for model predictive control with a horizon of only one step.

Proof of Lemma 7.4.4. In terms of system theory, v_k is the *state* of the abstraction: The future behavior $v_{k+1,k+2,\dots}$ is uniquely determined by v_k and the future inputs $\sigma_{k,k+1,\dots}$. Therefore, given the bound for v_k, the past inputs $\sigma_{0,1,\dots,k-1}$ are irrelevant for the proof. The requirement "$\forall \sigma_{0,1,\dots,k-1}$" can therefore be dropped without loss of generality, as shown by the next step:

In the following, consider a fixed value of k. To remove the dependency on k, the time axis is shifted: Let $\sigma_{k,k+1,\dots} = 0$. Then, starting at v_k, the abstraction is a linear time-invariant system as in the previous proof. Due to time-invariance, one can shift the time axis so that k becomes 0: For all i,

$$v_{k+i}\big|_{\sigma_{k,k+1,\dots}=0,\ v_0=v_0} = \underbrace{v_i\big|_{\sigma_{0,1,\dots}=\tilde{\sigma}_{0,1,\dots},\ v_0=\tilde{v}_0}}_{=:\tilde{v}_i}, \text{ where } \tilde{v}_0 := v_k,\ \tilde{\sigma}_i := 0. \tag{292}$$

The right-hand side $\tilde{v}_i$ denotes the solution of the original abstraction dynamics (216b) for a different initial state $\tilde{v}_0$ and ideal execution $\tilde{\sigma}_i = 0$. In this formulation, previous results for ideal execution can be reused:

$$(291) \overset{(292)}{\Leftrightarrow} \Big(\tilde{v}_0 \leq |s|_{\max} \quad \Rightarrow \quad \underbrace{\Big(v_i\big|_{\sigma_{0,1,\dots}=0,\ v_0=\tilde{v}_0} \leq |s|_{\max} \quad \forall i\Big)}_{\Leftrightarrow \text{ Assumption 7.4.2 with } v_0 \text{ replaced by } \tilde{v}_0.}\Big). \tag{293}$$

Applying Lemma 7.4.3, this is equivalent to

$$(291) \quad \Leftrightarrow \quad \Big(\tilde{v}_0 \leq |s|_{\max} \quad \Rightarrow \quad \underbrace{\Big(|s|_{\max} \geq \max\Big\{\tilde{v}_0, \frac{\beta_0 |d|_{\max}}{1-\rho_0}\Big\}\Big)}_{(280) \text{ with } v_0 \text{ replaced by } \tilde{v}_0}\Big). \tag{294}$$

By Assumption 7.4.2 and Lemma 7.4.3, $|s|_{\max} \geq \frac{\beta_0 |d|_{\max}}{1-\rho_0}$ is already satisfied. The previous statement then reduces to truth, which concludes the proof:

$$(291) \quad \Leftrightarrow \quad (\tilde{v}_0 \leq |s|_{\max} \Rightarrow |s|_{\max} \geq \tilde{v}_0) \quad \Leftrightarrow \quad \text{true}. \tag{295}$$

□

7.4.2 Safety-Net Policy

With the previous results, a simple proof suffices to show that the safety-net policy achieves the goal:

Theorem 7.4.5 (Safety)**.** *Under Assumptions 7.4.1 and 7.4.2, any safety-net policy (Definition 7.3.1) guarantees $v_k \leq |s|_{\max} \forall k$ and therefore guarantees the physical safety specification $|s_k| \leq |s|_{\max}$ over all time.*

Proof. The proof follows by induction.

Induction Assumption IA(k): $v_k \leq |s|_{\max}$.

Start of Induction IA(0) is $v_0 \leq |s|_{\max}$, which holds by Assumption 7.4.2 and Lemma 7.4.3.

Induction Step Assume IA(k).

1. Consider the case $\sigma_k = 0$. By Lemma 7.4.4, $v_{k+1} \leq |s|_{\max}$ if $\sigma_{k,k+1,\ldots} = 0$. By the structure of Equation (216b), v_{k+1} does not depend on $\sigma_{k+1,\ldots}$, so the condition $\sigma_{k,k+1,\ldots} = 0$ can be relaxed to $\sigma_k = 0$ without loss of generality. Therefore, IA($k+1$) holds.
2. Consider the remaining case $\sigma_k \neq 0$. By the policy (277), this case only occurs if $\hat{v}_{k+1}(\hat{\sigma}_k) \leq |s|_{\max}$, and then $\sigma_k = \hat{\sigma}_k$. Hence, $v_{k+1} = \hat{v}_{k+1} \leq |s|_{\max}$, so IA($k+1$) holds.

Conclusion IA(k) holds for all k, so safety is guaranteed by Theorem 6.2.3. □

7.5 Experiments

The results are now applied to the three-axis angular rate control subsystem of a quadrotor helicopter (Example D2 from Appendix A) and evaluated in numerical experiments. For the corresponding source code, see Appendix F.

In addition to the assumptions stated in Section 7.2, the following experiments assume an exact initial state ($x_0 = 0$), negligible measurement noise ($H_\mathrm{p} = 0$) and bounded disturbance acting on the rotor speed ($G_\mathrm{p} = B_\mathrm{p}, |d|_\mathrm{max} = 10^{-3}$). The safety specification of the control system is given as $C_\mathrm{s} = I, |s|_\mathrm{max} = 10$. This implies that the states $x_\mathrm{p}^{(i)}$ are bounded. Here, these states are the angular velocity around each axis of rotation. As the initial state is $x_0 = 0$, this implies $|\tilde{x}_0|_\mathrm{max} = 0$, $C_0 = I$. The value of $|d|_\mathrm{max}$ is chosen so that for ideal execution, the best known bound on the worst case is approximately $|s| \leq 1$. Loosely interpreted, the specification therefore states that the angular rate error under modified execution may be ten times worse than under ideal execution. Note that as the system model is linear, the absolute values of disturbance and safety specification are not relevant, only their ratio $|s|_\mathrm{max}/|d|_\mathrm{max}$ matters.

Ideally, an abstraction should achieve an exact bound $v_k = |s_k|$ on the safety output. As the abstraction used here does not consider any measurements of the disturbance or initial state, at best it can match the worst-case of s_k, i.e.,

$$v_k = |s_k|_\mathrm{worst} \coloneqq \max_{\substack{x_0, d_0, \ldots, d_{k-1} \\ |d_i| \leq |d|_\mathrm{max}, |\tilde{x}_0| \leq |\tilde{x}_0|_\mathrm{max}}} |s_k|. \tag{296}$$

Formally, $|s_k|_\mathrm{worst}$ is related to the worst-case gain of a time-varying linear discrete-time MIMO system. As detailed in Appendix E.1, no exact result is known, so upper and lower bounds for $|s_k|_\mathrm{worst}$ are determined instead.

For comparison, a simulation with random disturbance is shown in each experiment. The simulation considers the same execution and timing as the rest of the experiment, which will be explained later for each scenario. The random disturbance d_k in each time step is computed by scaling a vector of n_dist uniformly random numbers within $[-1; +1]$ to maximum amplitude $|d_k| = |d|_\mathrm{max}$. Compared to uniformly drawing from $|d_k| \leq |d|_\mathrm{max}$, using the maximal disturbance amplitude typically increases the probability of reaching large output amplitudes. This claim is supported by numerical experiments and motivated by the fact that for a more restricted system class (SISO, time-invariant), the worst-case disturbance only takes values $d_i = \pm|d|_\mathrm{max}$ (Balakrishnan and Boyd, 1992).

7.5.1 Basic Case

An example with ideal timing is shown in Figure 38. The graph shows a significant margin between the lower and upper bound on the unknown actual worst-case $|s_k|_\mathrm{worst}$. The random simulation sometimes exceeds the lower but never the upper worst-case bound. The abstraction is a safe upper

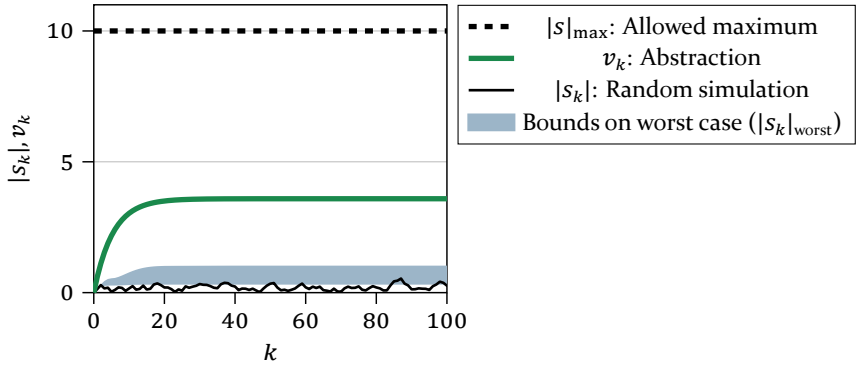


Figure 38: Basic case of the simulation with ideal timing and execution.

bound but pessimistic by a factor of about 4 compared to the theoretical upper bound. Still, there remains a relevant margin between the abstraction value $v_k \approx 4$ for ideal timing and the allowed maximum $v_k \leq |s|_{\max} = 10$ that was specified in the experiment. This margin can be traded off for relaxed timing, as will be illustrated by two scenarios.

7.5.2 Skipped Events

In the first scenario, timing is neglected ($\Delta t \equiv 0$) but skips are permitted. The combination of timing and skips is not considered because there is no known abstraction or other analysis method that supports both skips and timing delays for multiple inputs and outputs. However, this restriction may be lifted in future work (cf. Conjecture 5.1.12).

Six categories of execution are considered:

- o: Normal execution
- E: Everything was skipped, including the controller.
- S: At least one of the three sensor tasks was skipped. According to the system model, old sensor data is used in these cases for updating the controller. The actuator tasks and controller are executed normally.
- SA3 (SA4): One to three (all four) actuator tasks and an arbitrary number of sensor tasks are skipped. The controller is executed normally. As before, the controller is run without knowledge of these skips, possibly using old data values.

- X: Other combinations (skip the controller but not all sensors and actuators).

Abstraction Formally, these categories partition Σ into subsets Σ_0, Σ_E et cetera. For this summarized execution choice $i \in \{0, E, S, SA3, SA4, X\}$, the abstraction parameters are determined by $\rho_i = \max_{\sigma \in \Sigma_i} \|A(\sigma)\|_P$ (cf. Theorem 6.5.8) as $\rho_0 = 0.83$, $\rho_S = 1.53$, $\rho_E = 1.77$, $\rho_{SA3} = 917$, $\rho_{SA4} = 1.64$ and $\rho_X = 277\,000$. Also, $\beta_i = 610$ for any i, and the value of α is irrelevant because $x_0 = 0$. These values were rounded upwards to three significant digits. With slight abuse of notation, these six categories are in the following considered as abstract execution choices, e.g., $\sigma_k = E$ abbreviates $\sigma_k \in \Sigma_E$.

The results show that the modes $SA3$ and X are not reasonably usable with the current method. They are therefore excluded by merging all actuators into one task and by always executing the controller if any sensor or actuator was used. This leaves $\Sigma = \{0, S, SA4, E\}$.

The problematic values of ρ for skipping only a subset of the actuators are related to the physical structure of the system: The three directions of rotation are each controlled by the sum or difference of multiple rotors. Delaying the thrust command of one rotor therefore causes an unintended cross-coupling between multiple axes and consequently a large deviation from the original system dynamics. Also, the controller does not react to skips but always uses the same dynamics. To allow for even more flexibility, future work should consider controllers that depend on which tasks were executed in the previous period and thereby partially compensate the effects of skipping.

Policy The resource management used in this experiment follows simple rules that are easy to implement at run-time:

1. Skipping everything is allowed if $\hat{v}_{k+1}(E) \leq |s|_{\max}$.
2. Skipping all four actuators and any number of sensors is allowed if $\hat{v}_{k+1}(SA4) \leq |s|_{\max}$.
3. Skipping any number of sensors is allowed if $\hat{v}_{k+1}(S) \leq |s|_{\max}$.
4. Skipping nothing is always allowed, as this is the safety mode.

The implementation must ensure that the combination of skipped tasks matches the rules but is not further restricted. It should be noted that if a rule is enabled, then the rules below it will also be enabled: As $\rho_0 \leq \rho_S \leq \rho_{SA4} \leq \rho_E$ and all β_i are equal, $\hat{v}_{k+1}(0) \leq \hat{v}_{k+1}(S) \leq \hat{v}_{k+1}(SA4) \leq \hat{v}_{k+1}(E)$.

Proving the safety of this scheme is easy: By construction, the policy is a valid safety policy matching Definition 7.3.1, as it includes a fallback to the safety mode $\sigma = 0$. The safety mode is valid because Assumptions 7.4.1 and 7.4.2 hold, where the latter can be checked via Lemma 7.4.3, which requires $|s|_{\max} \geq 3.589$. Consequently, Theorem 7.4.5 guarantees safety.

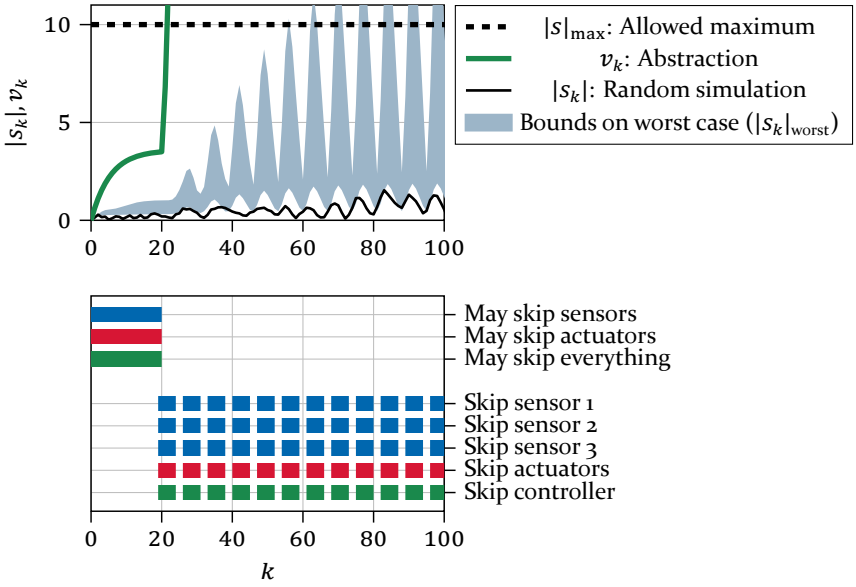
Simulated Scheduler The scheduling is simulated by a stochastic choice following the above rules. A practical motivation for this scenario is a real-time system that misses its deadlines with a low probability p_{skip} and then recovers from this fault by canceling or skipping some tasks (Pazzaglia et al., 2019). This kind of soft deadline enables higher average utilization and is used whenever the policy allows skipping. Otherwise, deadlines are considered hard, which means that some non-control tasks must be omitted to guarantee that the control tasks finish. At worst, when no skips are currently allowed, the execution is the same as in a classical hard real-time system.

In the following, $R_{\dots}$ denotes an independent (Bernoulli-distributed) random boolean variable that is true with probability p_{skip}.

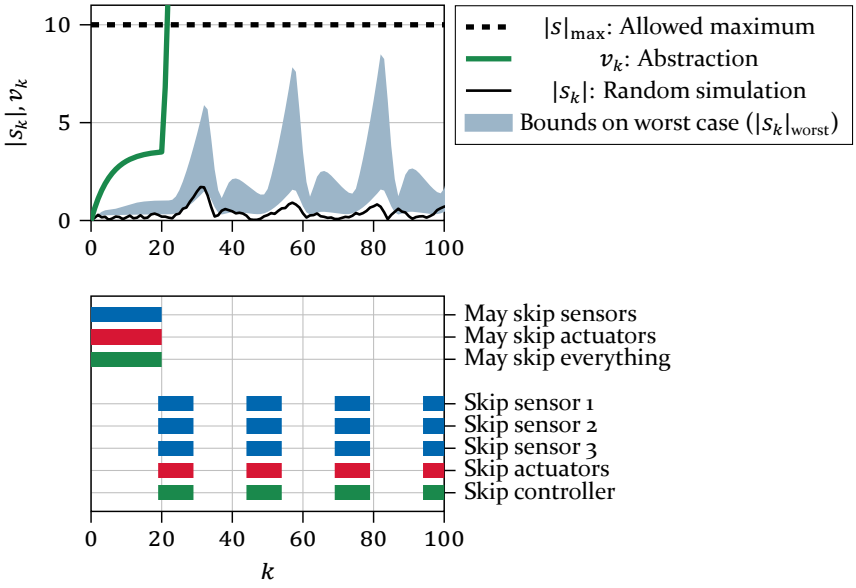
1. Skip everything if $\hat{v}_{k+1}(E) \leq |s|_{\max}$ and $R_{E,k}$.
2. Skip all four actuators if $\hat{v}_{k+1}(SA4) \leq |s|_{\max}$ and $R_{A,k}$.
3. For each sensor j, skip it if $\hat{v}_{k+1}(S) \leq |s|_{\max}$ and $R_{S,j,k}$.
4. If a task was not skipped by any rule, run it.

Experimental Results To illustrate the susceptibility of the system to skips, two scenarios slightly beyond the stability limit are shown in Figure 39. In the experiment, either everything or nothing is skipped, according to a fixed pattern. The figure shows the execution pattern "skip 5, execute 2" in the top and "skip 10, execute 15" in the bottom. For slightly lower skip ratios, which are not depicted here, the system is stable and respects the safety specification. The abstraction returns a significantly more pessimistic view and would not permit such a large skip ratio.

Next, the skipping is determined according to the abstraction-based policy. The result for a small probability of skips ($p_{\text{skip}} = 0.02$) is shown in Figure 40. It can be seen that the value of the abstraction increases whenever a control task is skipped and decays to a steady state of $v \approx 4$ otherwise. Most of the time, all three rules allow skips, while the behavior becomes more restrictive if the abstract state grows due to a recent skip.



(a) Skip 5, then execute 2



(b) Skip 10, then execute 15

Figure 39: System response to skipped executions. The skip patterns are fixed without regard to the policy.

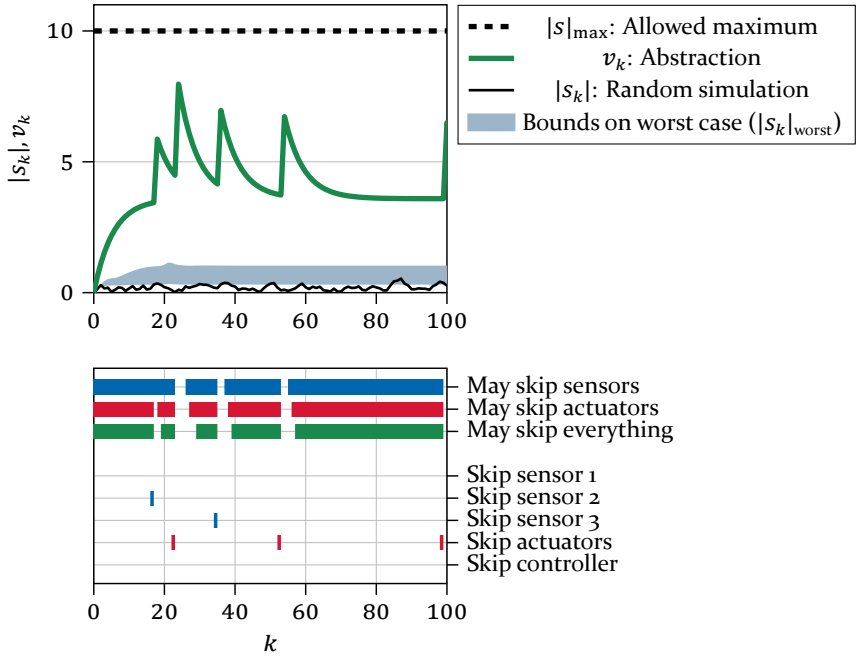


Figure 40: Abstraction-based resource management for low-probability skipping of tasks ($p_{\text{skip}} = 0.02$)

Figure 41a shows the behavior for a high skip probability of $p_{\text{skip}} = 0.7$: The system converges to an approximately periodic pattern, where skips occur until the abstract state is near the safety limit and then, skipping is denied until the abstract state has decayed slightly.

Unlike before, skipping is permitted only for a small fraction of time, which can be expected from the scenario: In the analogy of a driving school, a worse learner driver will necessarily suffer from more interference by the driving instructor. Comparing the abstraction with the computed worst-case reveals that the abstraction-based resource management is significantly pessimistic.

Discussion The results show that the abstractions are considerably pessimistic and only applicable to scenarios with a low probability of skips. Matching the theoretical observations of Section 6.6, the pessimism at skips is higher than for the nominal case.

Nevertheless, the presented technique constitutes an improvement since none of the reviewed works on dynamic resource management can handle multiple variants of skipping and nevertheless provide safety guarantees. Abstractions are more general but also more pessimistic than specialized results for restricted scenarios such as a single binary decision (e.g., skip all or nothing). Overall, there is a need for future research on improving the accuracy of abstractions. This discussion will be continued later.

Heuristic Variant If the requirement of mathematically proven safety is set aside, the parameters of the abstraction can be heuristically chosen to define the skip pattern: Increasing $|s|_{\max}$ or decreasing β allows for a higher number of consecutive skips. The maximum ratio between skipping and normal execution is mainly adjusted by ρ_σ/ρ_0. The relative badness of two execution modes σ_1 and σ_2 is parameterized by $\rho_{\sigma_1}/\rho_{\sigma_2}$.

In this interpretation, the abstraction parameters can be used as tuning factors, analogous to the heuristic choice of task priorities or requirements such as "skip at most 1 out of 10 executions". Here, abstractions are more versatile since they support multiple execution modes and, if available, can use information about varying external influence $|d|_{\max}$. Besides manual adjustment, black-box optimization using simulations is a possible route.

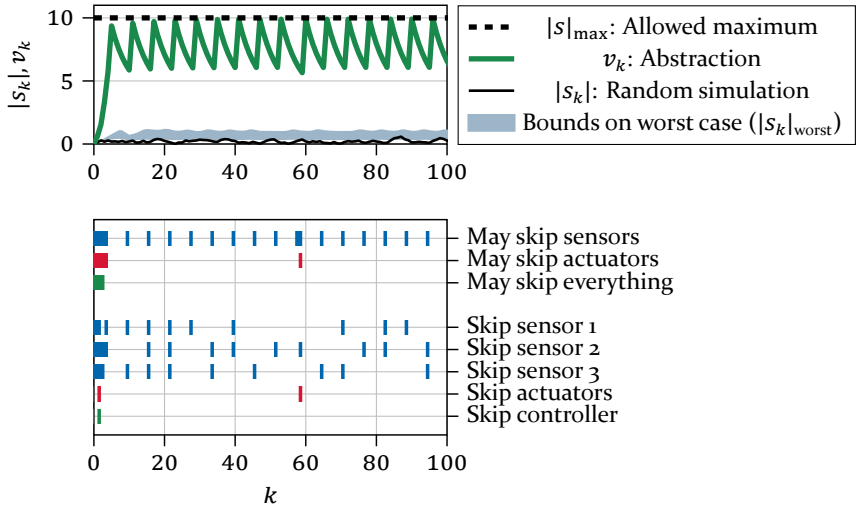
For illustration, the parameters of the previous simulation are adjusted to permit significantly more skips by changing ρ_i to $\rho_i^{1/5}$ for $i \neq 0$, ρ_0 to 0.7, and β to 0.15β. Figure 41 compares simulations for the original, validated abstraction (top) and for the new heuristic variant (bottom). A high skip probability of $p_{\text{skip}} = 0.7$ was assumed. The plots show that the heuristic "abstraction" permits significantly more skips but *may* still be valid: For the simulation run, the abstraction is above the simulation and above the theoretical lower bound on the worst case. While the result is not proven, it hints towards the possible existence of less pessimistic abstractions.

In summary, the abstraction approach works even for complex skip combinations, however, its pessimism limits the use to very low skip probabilities.

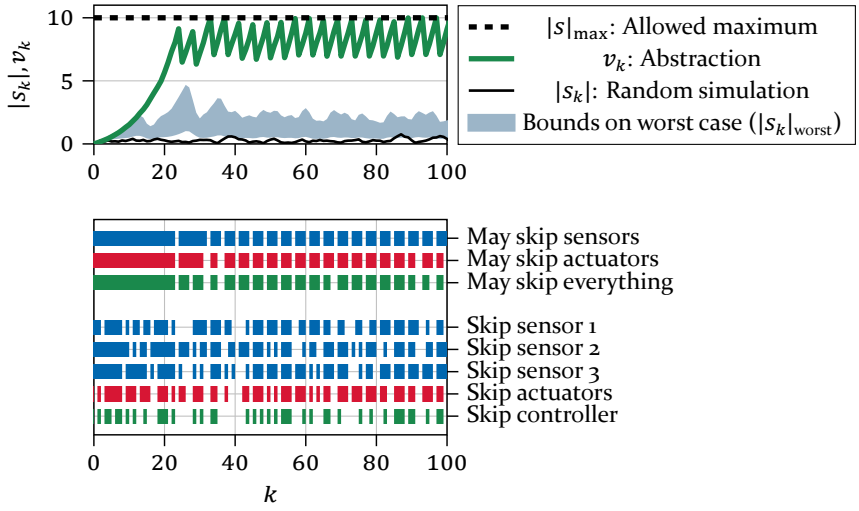
7.5.3 Input/Output Timing

The second scenario focuses on the effects of varying IO timing. No tasks are skipped. The normalized actual timing is defined as

$$\sigma_k \coloneqq \frac{1}{\Delta t_{\text{nominal}}} \max\{|\Delta t_{\text{u},k}^{(1)}|, \dots, |\Delta t_{\text{u},k}^{(m)}|, |\Delta t_{\text{y},k}^{(1)}|, \dots, |\Delta t_{\text{y},k}^{(p)}|\}. \tag{297}$$



(a) Original abstraction with proven safety guarantee



(b) Heuristic variant: More skips permitted but no mathematical safety proof

Figure 41: Dynamic resource management for a high probability of skipping, comparing the original abstraction (top) with a heuristic, not guaranteed variant (bottom)

As in the previous example, this summarized parameter is denoted σ_k with some abuse of notation. $\Delta t_{\text{nominal}} = 0.01T$ is a normalization factor, chosen such that $\sigma = 1$ represents the maximum timing uncertainty of Example D2 (Appendix A.3). The analysis method of Section 5.1 can show stability for a permanent timing uncertainty within $\sigma \leq 1$ but not for $\sigma \leq 2$.

The dynamic resource management can specify the maximum permitted timing of the next period, i.e., the range of σ_{k+1}, by a variable $\hat{\sigma}_{k+1}$. Again, denoting this as $\hat{\sigma}$ is with some abuse of notation.

The timing $\Delta t_{u,k}^{(i)}$ and $\Delta t_{y,k}^{(i)}$ is modeled as independent identically distributed random values which are mostly close to zero but with some extreme outliers up to the limit of $\pm\hat{\sigma}_k \Delta t_{\text{nominal}}$, so that $0 \leq \sigma_k \leq \hat{\sigma}_k$. The random model, which is detailed in the following, is motivated by the real-world timing distribution shown in Figure 4 on page 21.

Details on Randomness The deliberately simple mathematical model for the random timing is

$$\Delta t_{\mathrm{u},k}^{(i)} = \mathrm{sat}(\Delta t_0 R_{\mathrm{u},i,k}, [-\hat{\sigma}_k \Delta t_{\text{nominal}}; \hat{\sigma}_k \Delta t_{\text{nominal}}]), \tag{298}$$

where

$$\mathrm{sat}(x, [a; b]) \coloneqq \begin{cases} x, & a \leq x \leq b, \\ b, & x > b, \\ a, & x < a \end{cases} \tag{299}$$

is the saturation function, $R_{\dots}$ is a random variable, Δt_0 is a scaling factor that influences the mean square timing deviation and $\hat{\sigma}_k$ is the timing limit given by the dynamic resource management.

In (298), $R_{\mathrm{u},i,k} = R_{\gamma,\mathrm{u},i,k} R_{\mathrm{sign},\mathrm{u},i,k}$ is the product of two random variables, where $R_{\gamma,\mathrm{u},i,k}$ is drawn from the gamma distribution with mean $1/3$ and variance 1, and $R_{\mathrm{sign},\mathrm{u},i,k}$ is a random sign uniformly drawn from $\{-1, +1\}$. All random variables are independent. The resulting cumulative distribution function is shown in Figure 42. The same distribution is used for $\Delta t_{\mathrm{y},i,k}$.

Abstraction Using the simplified definition of σ_k and $\hat{\sigma}_k$ introduced before, an abstraction is determined from P. As in the previous scenario, $\beta = 610$. By the stability analysis of Section 5.1 and an approximation discussed in Appendix E.2, ρ is determined as

$$\rho_\sigma = c_3 + c_4 \sigma \quad \text{for } 0 \leq \sigma \leq 10, \quad c_3 = 0.860, c_4 = 0.0996. \tag{300}$$

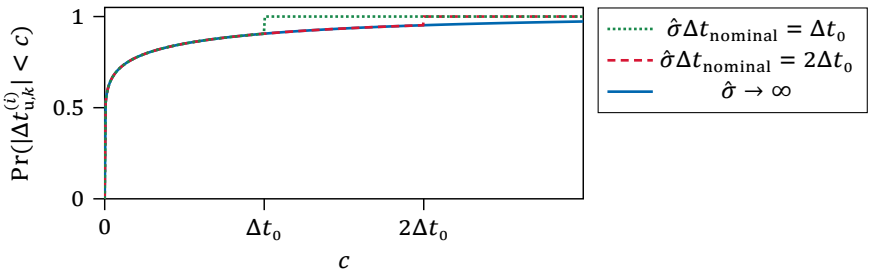


Figure 42: Cumulative distribution function of random timing deviation

The resulting limit for guaranteed stability is $\sigma = 1.4$, as this value results in $\rho_\sigma = 1$. As will be shown soon, the largest permissible *static* timing bound is slightly below the stability limit since disturbance rejection is a stronger requirement than stability. *Dynamic* resource management can temporarily exceed the static and even the stability limit if the recent timing was good.

Policy As a resource management policy, the largest timing range satisfying the abstract safety requirement is used, limited to $\hat{\sigma}_k \leq 10$:

$$\hat{\sigma}_k = \operatorname{sat}\left(\max\{\hat{\sigma}_k \mid \overbrace{\rho_{\hat{\sigma}_k} v_k + \beta |d|_{\max}}^{\hat{v}_{k+1}(\hat{\sigma}_k)} \leq |s|_{\max}\},\ [0; 10]\right) \tag{301}$$

$$\overset{(300)}{=} \operatorname{sat}\left(\frac{1}{c_4}\left(\frac{|s|_{\max} - \beta |d|_{\max}}{v_k} - c_3\right),\ [0; 10]\right), \tag{302}$$

where division by zero is defined as positive infinity. Safety of this policy can be shown in the same way as for the previous example.

Inserting $v_k = |s|_{\max}$ into the above formula yields the largest permissible *static* timing policy as $\hat{\sigma}_k = \sigma_{\text{static}} \approx 0.793$. For this value, the safety specification is barely guaranteed, as the abstraction converges to $v_\infty \approx |s|_{\max}$.

Experimental Results Figure 43 illustrates the sensitivity of the system to large timing deviation. The simulation starts with ideal timing ($\sigma_k = 0$) until $k = 50$. Then, the timing deviation is set to $\pm 0.49T$, where the sign is chosen randomly for every sensor and actuator in every period ($\sigma_k = 49, \Delta t_0 \to \infty$). The minor response to increased timing indicates that the system would tolerate permanent timing deviations much larger than $\sigma_k = 1$. Here, dynamic resource management is limited by the underlying stability analysis of Chapter 5, which does not succeed for $\sigma_k = 2$.

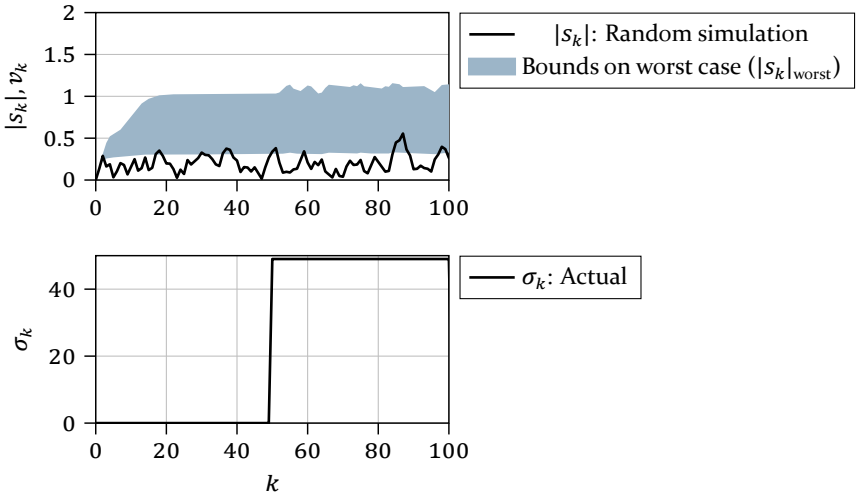
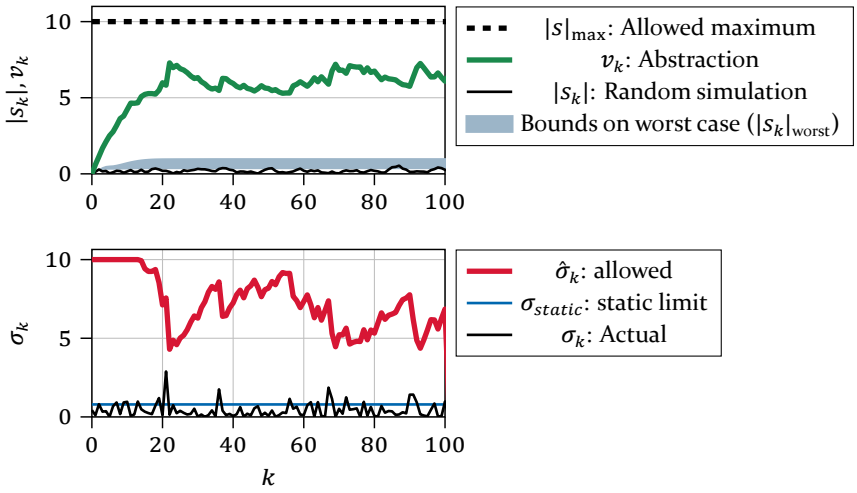


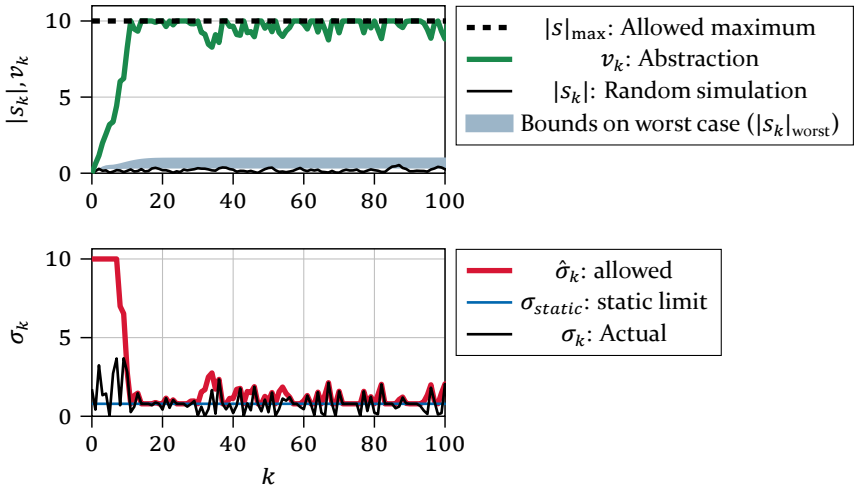
Figure 43: System response to varying timing. The timing pattern is pre-determined without regard to the policy.

Experimental results for $\Delta t_0 = \Delta t_{\text{nominal}}/4$, i.e., for a low probability of timing deviation, are shown in Figure 44a. Throughout the experiment, the dynamic abstraction-based policy permits large timing deviations $\sigma > \sigma_{\text{static}}$ beyond the limit σ_{static} of a static policy. The dynamic nature of the abstractions allows temporarily violating this limit while respecting safety: It can be seen that the abstraction state is lower and the permitted timing deviation is higher whenever the recent timing was good.

As illustrated in the previous experiments on skipped executions, a key to success is that the actual timing is typically much better than the permitted maximum. Figure 44b shows a scenario in which this does not hold and therefore dynamic resource management has little benefit. Therein, the previous experiment is repeated with a larger probability of timing deviations ($\Delta t_0 = \Delta t_{\text{nominal}}$). Then, the abstraction is often near the safety bound, so the permitted timing deviation is close to the static limit. This illustrates an important property of dynamic resource management in general: In the worst case, there is nothing to win compared to a static timing bound. Dynamic adaptation is only useful if there is a significant difference between average- and worst-case timing.



(a) Low probability of timing deviation: $\Delta t_0 = \Delta t_{\text{nominal}}/4$



(b) Higher probability of timing deviation: $\Delta t_0 = \Delta t_{\text{nominal}}$

Figure 44: Resource management with guaranteed abstractions and uncertain timing.

Discussion The experiments on varying timing paint a similar picture as the previous experiments on skipped executions. Because the complex timing scenario far exceeds the capabilities of all reviewed existing work, the only baseline is the design-time analysis of Section 5.1. If the probability of large timing deviations is sufficiently low, the presented approach to dynamic resource management is beneficial. For higher probabilities, the pessimism of the abstractions dominates and design-time analysis is better.

Obtaining more accurate abstractions is therefore the key to broader applicability. The simulations of Figure 11 on page 51 indicate that increased timing uncertainty changes the system dynamics significantly but has little effect on stability and disturbance rejection. However, the norm bounds of the underlying stability analysis (Section 5.1) treat this benign deviation the same as a malign, destabilizing influence and are therefore significantly pessimistic.

7.6 Concluding Remarks

The classical design of safety-critical real-time control systems suffers from unnecessarily strict timing requirements. This chapter presented a solution to dynamically relax the timing requirements while still providing strict physical safety guarantees. The key element of the solution is an *output convergence rate abstraction* of the control system. At run-time, this abstraction provides a lightweight dynamic model that predicts the "health" of the system based on the "damage" inflicted by previous execution conditions and the gradual "healing" under normal execution. Based on this model and simple decision rules, a safe estimate of the currently permissible timing requirements can be determined. Executing the abstraction and decision rules is cheap and boils down to a few scalar arithmetic and comparison operations, even for control systems with multiple sensors and actuators and therefore a significant number of timing variables. Hence, the technique supports more complex timing models than all known work on dynamic resource management.

Conservatism As the experiments show, the presented solution can trade off larger physical bounds for more relaxed timing requirements. In their current implementation, the abstractions are quite conservative, which means there is significant room for further relaxation of timing by determining tighter abstractions. Theoretical ideas on how to improve the abstractions were discussed previously in Section 6.8.

Furthermore, there are possible improvements specific to this chapter: As the Lyapunov matrix P was not optimized towards this case but merely reused from the stability analysis of Section 5.1, it can be assumed that an improvement is possible by optimizing P. A further aspect of conservatism is that the maximum disturbance amplitude is assumed as constant. In practice, this is pessimistic: For example, a UAV is designed for worst-case wind disturbance but often used in good weather, where the execution could be relaxed even further. Measurements could be used to correct the pessimistic abstract state to a more realistic value, analogous to a set-valued observer.

Implementation in Real-Time Operating Systems The general scheme derived in this chapter supports many forms of weakly-hard execution. In particular, the current implementation supports either timing jitter or skipping tasks, and a combination could be achieved by extending the underlying stability analysis (cf. Conjecture 5.1.12). This allows for more flexible timing, which in principle allows for more efficient resource usage. However, current real-time operating systems are tailored to fixed timing, so new variants must be developed. For this, the framework of abstractions offers wide freedom of implementation, as long as two simple requirements hold:

1. A measurement of the execution conditions is available.
2. The system can temporarily switch into a safety mode for which stability can be verified.

These requirements are closely related to mixed-criticality scheduling (cf. page 25), which follows a similar scheme but is focused on temporal guarantees: For an exemplary control system, the tasks are partitioned into critical (control-related) and uncritical (other) tasks. If the critical tasks finish within their typical execution time, the uncritical tasks are executed with proper timing guarantees. In the rare case in which the critical tasks require more time than usual, the system operates in a "safety mode" where uncritical tasks are skipped to ensure that the critical tasks still meet their deadlines. The dynamic resource management shown in this chapter allows for a generalization of mixed-criticality scheduling from temporal guarantees, which restrict every single period, to physical safety requirements, which are more tolerant.

Summary The theory and experiments presented in this chapter substantiate the claim that abstraction-based dynamic resource management can lead to safe control systems with significantly relaxed real-time requirements. If there is a large difference between average and worst-case timing, as it typically results from the pessimism of worst-case execution time analysis,

then the timing bounds can be relaxed far beyond the limits of design-time techniques. The next step for future work is to implement the technique in a real-time operating system, building upon existing work for mixed-criticality scheduling. Further open questions are reducing the conservatism, such as by detecting varying disturbance amplitude.

8 Conclusion

The essence of this doctoral thesis is that safe control systems are possible without the classical restriction of exact input and output timing. This chapter summarizes the contributions of this thesis and gives an outlook on promising extensions for future research.

8.1 Summary

Motivation The design of real-time control systems is facing an increasing challenge due to ever-growing safety requirements, system complexity and cost pressure. In particular, it is hard to satisfy the exact input/output timing that is traditionally assumed for controller design and verification. A key reason is the general design conflict in computing systems between cost efficiency and predictable timing. Furthermore, sensors and actuators in modern control systems are often digital devices, which leads to communication delays and related timing deviations.

To avoid overly strict timing requirements and still ensure provable safety, this doctoral thesis presented mathematical techniques for the safety verification of real-time control systems with uncertain timing. A particular focus was on the case of multiple inputs and outputs with individual timing deviations. This case is not possible in all surveyed other work, which often assumed that all inputs are sampled at exactly the same time. However, that assumption generally does not hold if there are multiple separate sensors.

Background Chapter 2 summarized the background on real-time control systems: Considering the aspects of safety engineering, real-time computing and control theory leads to a variety of options for the *co-design of computing and control*. Two major options were considered in this thesis: The first is design-time analysis, where an upper bound on the worst-case control performance is obtained, given a fixed bound on the timing deviations. A second option is dynamic resource management, which means that the timing bound is adjusted dynamically at run-time, widening it as far as permitted by a specified worst-case control performance. Because dynamic resource management considers the effects of previous timing, larger short-term timing deviations are permitted if the recent timing was good enough. A helpful fact is that computing systems often have a large gap between average and

worst-case timing. In theory, dynamic resource management can therefore achieve greater timing flexibility than design-time analysis. In practice, the major obstacle is that the computational overhead at run-time can quickly outweigh the benefits of greater timing flexibility.

Formalization Chapter 3 translated these ideas into a mathematical system model and problem statement. The model formalizes the plant, the controller and the effects of varying timing for each input and output. Optionally, the model supports skipped executions and skipped input and output samples. In practice, these skips can either result from missed events due to timing overruns or from deliberately skipped events to save computing or communication resources. The problem statement formalizes the problems of design-time analysis and dynamic resource management. Furthermore, it defines two criteria for the control performance: safety, which means that the plant state remains within specified bounds, and exponential stability.

Framework Chapter 4 presented two theoretical frameworks for design-time analysis: linear impulsive systems (LIS) and hybrid automata (HA).

LIS combine the discrete-time and continuous-time cases of linear dynamics. The advantage of LIS is that they admit an explicit solution, which can be analyzed using matrix mathematics. However, the timing is merely considered as an external input that must be treated separately in analysis.

This is different for HA, which are a superset of discrete-event automata and continuous-time dynamical systems. HA can model the whole behavior of a real-time control system, including a timing model and optionally a mechanism for dynamic resource management. Nondeterministic behavior such as uncertain timing is directly supported by HA and the corresponding analysis methods. However, this generality makes the theoretical treatment more difficult than for LIS.

Design-Time Analysis Chapter 5 presented methods for design-time analysis in the case of bounded input/output timing deviations. Stability analysis was implemented using LIS and safety analysis was considered using HA.

The presented LIS-based approach is the first known method for the stability analysis of real-time control systems with multiple inputs and outputs (MIMO) with individual timing deviations. Reusing existing single-input-single-output (SISO) methods for the MIMO case is not practical, as the larger number of timing variables would lead to an overwhelming complexity. This is overcome by a time discretization and a decomposition that breaks down the

discrete-time dynamics matrix into a sum. The resulting summands depend on at most two timing variables, making it computationally feasible to obtain norm bounds. To allow for stability analysis, a norm is determined that corresponds to a quadratic Lyapunov candidate function. Thereby, the problem is reduced to a well-known case: the stability analysis of a discrete-time system with bounded model uncertainty. Finally, the discrete-time result is translated to the stability of the original continuous-time system.

Experimental results indicate that the method is feasible for MIMO systems with small timing deviations and a medium size, e.g., three sensors, four actuators, three physical and six controller states. For systems with larger dimensions or timing deviations, the method is rather pessimistic, which may be improved by a generalization to piecewise-quadratic Lyapunov functions. Furthermore, the method remains limited to the linear case.

For safety analysis, an HA-based method was considered that also supports nonlinear dynamics. In principle, worst-case bounds on the behavior of arbitrary HA can be determined using general-purpose software for reachability analysis. However, a practical experiment showed that a direct application of such tools is not viable. The key obstacle are the discrete transitions that represent the timing of multiple inputs and outputs. The high frequency of discrete transitions leads to rapidly accumulating approximation error, so that the resulting bounds grow to infinity.

Hence, the technique of continuization was employed, which transforms the automaton in order to mostly remove the discrete transitions. The result of this transformation is a pessimistic upper bound. The original technique of Bak and Johnson, 2015 was put on a more general formal basis using abstractions of hybrid automata. This allowed to uncover the limitation that the controller must be static, e.g., a proportional controller. A complementary variant, first-order continuization, was derived that supports certain dynamic controllers without feedthrough, e.g., a purely integral controller. Future work should consider merging both variants into a universal method.

For the basic case of ideal timing, experiments indicated that continuization is feasible if the sampling rate of the controller is sufficiently fast compared to the plant dynamics. Furthermore, this thesis sketched results for uncertain timing and an extension to the MIMO case. Therefore, continuization remains a promising field for future research.

Convergence Rate Abstractions Dynamic resource management can allow for larger timing flexibility than design-time analysis. However, its implementation is challenging because the decisions at run-time should be provably safe, not overly restrictive and still have little computational overhead. Particularly for the MIMO case considered in this work, the number of system states and timing variables is problematic.

Chapter 6 presented a solution that reduces the complexity of the original system dynamics while preserving safety guarantees: A one-dimensional dynamic system, termed *convergence rate abstraction*, is generated from the full-dimensional model of the real-time control system. The abstraction provides a pessimistic upper bound on the original system and is simple to analyze and predict due to its simple structure. In this thesis, one-dimensional abstractions were considered as the basic case and constructed from Lyapunov functions. Due to the reduced dimension, such abstractions are generally pessimistic, i.e., they tend to overestimate the impact of skipped executions or timing deviations. This can probably be reduced by higher-dimensional abstractions, which are a topic for future research.

A first application of the abstraction was for design-time analysis. This was exemplified for the case of (M, K)-weak execution, where the controller may be skipped in some of the periods.

Dynamic Resource Management More important is the case of dynamic resource management, which was considered in Chapter 7: Building upon the Lyapunov candidate function of the LIS-based analysis, a one-dimensional abstraction was derived whose simple structure allows for negligible run-time overhead. Then, an algorithm for dynamic resource management was constructed that evaluates the abstraction at run-time. The result is the first known technique for dynamic resource management that can handle the large number of timing variables occurring in the case of MIMO timing deviations and still provide safety guarantees and low computational overhead. Similar results were presented for the case of skipped controller execution and input/output sampling.

Simulations indicated that the technique is applicable to systems that are well-suited for the underlying LIS stability analysis, i.e., linear control systems with medium dimension and small timing deviations. A benefit compared to design-time analysis was shown for cases where the gap between average and worst-case timing was sufficiently large. As the performance of abstraction-based dynamic resource management critically depends on the quality of the abstraction, future research should consider improved abstractions.

8.2 Outlook

This doctoral thesis developed theoretical methods for the analysis and design of flexible timing mechanisms in real-time control systems, particularly for the MIMO case. The results point to a number of promising aspects for future research, which will be summarized in the following.

Theory From a theoretical perspective, one direction of improvement is to make analysis more precise. Mainly, this should be possible by a generalization to more complex mathematical objects used in the analysis: For the LIS-based method, the quadratic Lyapunov function could be replaced by piecewise-quadratic Lyapunov functions. For continuization, new interpolation variants beyond zero- and first-order-hold could be considered to reduce the approximation error. Furthermore, a generalized abstract proof of continuization would simplify an extension to complex timing scenarios. Regarding convergence rate abstractions, tighter bounds could be obtained using a higher-dimensional abstract state.

A second aspect is to consider a broader system model, as illustrated by the following examples: Nonlinear or hybrid plants and controllers occur in many practical applications. Network communication can lead to more general timing schemes, e.g., varying sampling rate or delays larger than half a period.

Practice Further research is also motivated by the growing number of applications in which classical timing requirements are no longer feasible. For example, automobiles are becoming "computers on wheels", where many features must be integrated on few computing units to keep the price affordable. Techniques such as mixed criticality and flexible timing play a critical role in solving this challenge. In this context, convergence rate abstractions and their implementation in real-time operating systems should be considered further.

Even if timing uncertainties are neglected, the verification of real-world control systems is challenging. With ever-growing system complexity, it has become impossible to cover all cases by simulation or testing. Computer-aided safety verification could solve the problem and thereby increase trust and shorten development times. This doctoral thesis provided insights on how to address the aspect of timing in verification. With further research, computer-aided verification could one day be as easy as simulation is today.

Appendix

A Example Data

Several example systems are used throughout this work. For reference, the systems are denoted by a letter, followed by a number indicating the variant. The naming is consistent throughout this thesis and the corresponding publications (cf. page 266). A previous version of this appendix was published in Gaukler and Ulbrich, 2019.

To limit implementation complexity and the number of variants, the examples are restricted to linear systems, and the experiments do not consider disturbance except as mentioned in Section 7.5.

A.1 One-dimensional Example (A1)

For an example of minimal dimensions $n = n_\mathrm{d} = m = p = 1$, a linear, weakly unstable plant $A_\mathrm{p} = 0.05, B_\mathrm{p} = 0.5, C_\mathrm{p} = 1$, without disturbance ($n_\mathrm{dist} = 0$) is controlled by the discrete-time controller $A_\mathrm{d} = -0.01$, $B_\mathrm{d} = -0.4$, $C_\mathrm{d} = 1$, $T = 1$. The timing is uncertain ($\underline{\Delta t}_\mathrm{y} = -0.1$, $\overline{\Delta t}_y = 0.002$, $\underline{\Delta t}_\mathrm{u} = -0.001$, $\overline{\Delta t}_\mathrm{u} = 0.002$) and the initial state is bounded within the interval $X_{\mathrm{p},0} = [-1; 1]$. It should be noted that even for this one-dimensional plant and controller, the resulting hybrid system has 4 continuous state variables $(x_\mathrm{p}, u, y_\mathrm{d}, x_\mathrm{d})$.

The parameters of this system were heuristically chosen such that the reachability analysis tool SpaceEx can verify stability via an invariant set within seconds with optimized settings and within minutes with almost arbitrary settings. Variants with increased and decreased difficulty are also provided:

A2 Negligible jitter: $\overline{\Delta}t_\mathrm{u} = \overline{\Delta}t_\mathrm{y} = -\underline{\Delta}t_\mathrm{u} = -\underline{\Delta}t_\mathrm{y} = 0.0001$

A3 Increased jitter: $\underline{\Delta}t_\mathrm{u} = -0.4, \overline{\Delta}t_\mathrm{u} = 0.1, \underline{\Delta}t_\mathrm{y} = -0.1, \overline{\Delta}t_\mathrm{y} = 0.4$

A4 Extreme jitter: Variant A3 with $\overline{\Delta}t_\mathrm{u} = 0.4$

A5 Increased dimension: Variant A3 is duplicated, which means that all dynamics matrices are diagonally repeated, e.g.,

$$A_{\mathrm{p},A5} = \begin{bmatrix} A_{\mathrm{p},A3} & 0 \\ 0 & A_{\mathrm{p},A3} \end{bmatrix}. \tag{303}$$

This increases the dimension by factor 2, which makes verification more difficult but does not change stability.

A6 (not used in this thesis) To add disturbance and measurement noise, the system may be changed to $n_{\text{dist}} = 2, G_{\text{p}} = \begin{bmatrix} 1 & 0 \end{bmatrix}, H_{\text{p}} = \begin{bmatrix} 0 & 1 \end{bmatrix}, D = [-1; 1]^2$.

A.2 Trivial Three-dimensional Example (B1)

For an academic example of higher order, a stable three-dimensional plant is combined with a controller that has negligible influence on the plant:

$$A_{\text{p}} = \begin{bmatrix} -1 & 0.002 & 0.003 \\ 0.004 & -5 & 0.006 \\ 0.007 & 0.008 & -9 \end{bmatrix}, \quad B_{\text{p}} = \begin{bmatrix} 0.001 & 0.0011 \\ 0.0012 & 0.0013 \\ 0.0014 & 0.0015 \end{bmatrix},$$

$$C_{\text{p}} = \begin{bmatrix} 16 & 17 & 18 \end{bmatrix}, \quad A_{\text{d}} = \begin{bmatrix} 0.019 & 0.02 \\ 0.021 & 0.022 \end{bmatrix}, \quad B_{\text{d}} = \begin{bmatrix} 0.023 \\ 0.024 \end{bmatrix},$$

$$C_{\text{d}} = \begin{bmatrix} 0.025 & 0.026 \\ 0.027 & 0.028 \end{bmatrix}, \quad T = 2, \quad \overline{\Delta t}_u = -\underline{\Delta t}_u = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix},$$

$$\overline{\Delta t}_{\text{y}} = -\underline{\Delta t}_{\text{y}} = 0.6, \quad \mathcal{X}_{\text{p},0} = [-1; 1]^3, \quad n_{\text{dist}} = 0. \tag{304a}$$

A.3 Quadrotor Helicopter (C, D)

The linearized dynamics of the angular rate control of a quadrotor helicopter, mostly based on a continuous-time benchmark by Cunha, 2015, are the basis for two models. Angular rate control is essential for flight stabilization, as it ensures that the rate of rotation r_ϕ, r_θ, r_ψ around the x, y and z axis follows the setpoint given by a higher-level flight control system. For simplicity, the latter is not considered here and instead the setpoint is assumed to be zero. For a detailed system description and physical model, see Cunha, 2015 and the references therein.

Single-axis Angular Rate Control (C) If only the rotation ϕ around the x-axis is considered, the mechanical dynamics of the angular rate r_ϕ are

$$J_\phi \dot{r}_\phi(t) = T_\phi(t) \tag{305}$$

with rotational inertia J_ϕ. The input is the time-varying torque $T_\phi(t)$ controlled by the motors and the output is the angular rate $r_\phi(t)$ measured by a gyroscope sensor.

The original, continuous-time controller

$$T_\phi(t) = -K_{p,\phi} r_\phi(t) - K_{I,\phi} \int_0^t r_\phi(\xi) \mathrm{d}\xi \tag{306}$$

given in Cunha, 2015 is discretized in a simplified way as

$$T_{\phi,k} = -K_{P,\phi} r_{\phi,k-1} - K_{I,\phi} \sum_{i=0}^{k-1} r_{\phi,i}. \tag{307}$$

To match the timing model, $T_{\phi,k}$ has no feedthrough, i.e., it does not depend on the current measurement $r_{\phi,k}$.

Three-axis Angular Rate Control (D) If all three axes of rotation are considered, the linearized dynamics are three independent integrators

$$J_\phi \dot{r}_\phi(t) = T_\phi(t), \tag{308a}$$
$$J_\theta \dot{r}_\theta(t) = T_\theta(t), \tag{308b}$$
$$J_\psi \dot{r}_\psi(t) = T_\psi(t). \tag{308c}$$

The torques $T_{\phi,\theta,\psi}$ and the vertical thrust F_z are controlled by the forces $u^{(1)}$, ..., $u^{(4)}$ of four propellers, which are defined as inputs to the system because rotor dynamics are neglected:

$$\begin{bmatrix} T_\phi(t) \\ T_\theta(t) \\ T_\psi(t) \\ F_z(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -l & 0 & l \\ -l & 0 & l & 0 \\ \gamma & -\gamma & \gamma & -\gamma \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{:=M_\mathrm{u}} \underbrace{\begin{bmatrix} u^{(1)}(t) \\ u^{(2)}(t) \\ u^{(3)}(t) \\ u^{(4)}(t) \end{bmatrix}}_{u(t)}. \tag{309}$$

This equation depends on the distance l between center and rotor and the ratio γ of torque to rotor thrust. Because altitude control is not considered here, the corresponding thrust force is set to $F_z = 0$, which results in

$$u(t) = M_{\mathrm{u}}^{-1} \begin{bmatrix} T_\phi(t) \\ T_\theta(t) \\ T_\psi(t) \\ 0 \end{bmatrix}, \tag{310}$$

where $T_{\phi,\theta,\psi}(t)$ is each computed by a PI controller in the structure of Equation (307) but with axis-dependent controller parameters.

It should be noted that for ideal timing, the system can be split into three decoupled subsystems, whereas if the components of $u(t)$ are not updated synchronously, the control torque of one axis slightly affects other axes.

While the presented model is highly simplified, it still remains challenging for verification. Possible extensions for future research include nonlinearities, rotor dynamics and position control.

Parameters All parameters given in Cunha, 2015 are used without modification:

$$\begin{aligned} J_\phi &= 9.036{\cdot}10^{-6}, & J_\theta &= 9.127{\cdot}10^{-6}, & J_\psi &= 1.937{\cdot}10^{-5}, \\ K_{P,\phi} &= 2.557{\cdot}10^{-4}, & K_{P,\theta} &= 2.581{\cdot}10^{-4}, & K_{P,\psi} &= 5.478{\cdot}10^{-4}, \\ K_{I,\phi} &= 3.614{\cdot}10^{-3}, & K_{I,\theta} &= 3.651{\cdot}10^{-3}, & K_{I,\psi} &= 7.747{\cdot}10^{-3}. \end{aligned} \tag{311}$$

The parameters missing in Cunha, 2015 are chosen as $l = 0.1$ and $\gamma = 0.01$, which is within the order of magnitude observed in other quadrotor helicopter models[1]. The controller period is chosen as $T = 0.01$, which is a compromise between a response similar to the original continuous-time controller and low processor utilization. The initial response of the original and the discretized controller are compared in Figure 45. It can be seen that the overshoot is increased from approximately 20 to approximately 40 percent.

The dynamics matrices for example C are

$$A_\mathrm{p} = 0, \quad B_\mathrm{p} = 1.107{\cdot}10^5, \quad C_\mathrm{p} = 1, \quad A_\mathrm{d} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$
$$B_\mathrm{d} = \begin{bmatrix} 1{\cdot}10^{-2} \\ 1 \end{bmatrix}, \quad C_\mathrm{d} = \begin{bmatrix} -3.614{\cdot}10^{-3} & -2.556{\cdot}10^{-4} \end{bmatrix}, \tag{320}$$

[1] For example, the quadrocopter "Parrot Rolling Spider" has $l = 0.0624$ and $\gamma \approx 0.0024$. These values can be obtained from the model of Lyu, 2017, pp. 63–64 using the following conversion:

Lyu uses the model

$$T_\phi = k_a d(-\tilde{u}^{(2)} + \tilde{u}^{(4)}), \tag{312}$$
$$T_\theta = k_a d(\tilde{u}^{(1)} - \tilde{u}^{(3)}), \tag{313}$$
$$T_\psi = k_m(\tilde{u}^{(1)} - \tilde{u}^{(2)} + \tilde{u}^{(3)} - \tilde{u}^{(4)}), \tag{314}$$
$$F_z = k_a(\tilde{u}^{(1)} + \tilde{u}^{(2)} + \tilde{u}^{(3)} + \tilde{u}^{(4)}), \tag{315}$$

where $\tilde{u}$ is a normalized input, $d = 0.0624$, $k_a = 4.720{\cdot}10^{-8}$ and $k_m = 1.139{\cdot}10^{-10}$. Comparing these equations with the structure of (309) leads to

$$\begin{bmatrix} u^{(1)} \\ u^{(2)} \\ u^{(3)} \\ u^{(4)} \end{bmatrix} = k_a \begin{bmatrix} \tilde{u}^{(3)} \\ \tilde{u}^{(2)} \\ \tilde{u}^{(1)} \\ \tilde{u}^{(4)} \end{bmatrix}. \tag{316}$$

The difference in indices occurs because the rotor numbering differs between Cunha, 2015 and Lyu, 2017. Applying (316) yields

$$T_\psi = \underbrace{k_m/k_a}_{\gamma}(u^{(1)} - u^{(2)} + u^{(3)} - u^{(4)}), \tag{317}$$
$$T_\phi = \underbrace{d}_{l}(-u^{(2)} + u^{(4)}), \tag{318}$$
$$T_\theta = \underbrace{d}_{l}(-u^{(1)} + u^{(3)}). \tag{319}$$

Therefore, $\gamma \approx 0.0024$ and $l = 0.0624$.

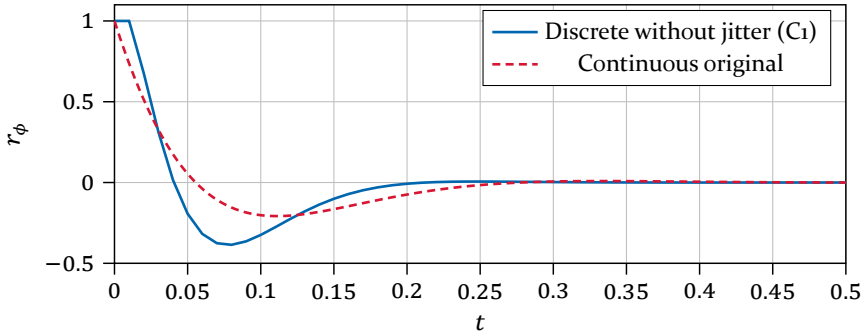


Figure 45: Initial response $r_\phi(t)$ of original and discretized controller for example C for an initial state $r_\phi(0) = 1$ and ideal timing.

and for example D are

$$A_\mathrm{p} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_\mathrm{p} = \begin{bmatrix} 0 & -1.107\cdot10^{4} & 0 & 1.107\cdot10^{4} \\ -1.096\cdot10^{4} & 0 & 1.096\cdot10^{4} & 0 \\ 5.163\cdot10^{2} & -5.163\cdot10^{2} & 5.163\cdot10^{2} & -5.163\cdot10^{2} \end{bmatrix},$$

$$C_\mathrm{p} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad A_\mathrm{d} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_\mathrm{d} = \begin{bmatrix} 1\cdot10^{-2} & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1\cdot10^{-2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1\cdot10^{-2} \\ 0 & 0 & 1 \end{bmatrix},$$

$$C_\mathrm{d} = \begin{bmatrix} 0 & 0 & 1.825\cdot10^{-2} & 1.291\cdot10^{-3} & -1.937\cdot10^{-1} & -1.370\cdot10^{-2} \\ 1.807\cdot10^{-2} & 1.279\cdot10^{-3} & 0 & 0 & 1.937\cdot10^{-1} & 1.370\cdot10^{-2} \\ 0 & 0 & -1.825\cdot10^{-2} & -1.291\cdot10^{-3} & -1.937\cdot10^{-1} & -1.370\cdot10^{-2} \\ -1.807\cdot10^{-2} & -1.279\cdot10^{-3} & -2.533\cdot10^{-18} & -1.791\cdot10^{-19} & 1.937\cdot10^{-1} & 1.370\cdot10^{-2} \end{bmatrix}. \tag{321}$$

Unless mentioned differently, no disturbance ($n_\mathrm{dist} = 0$) and an initial state within $\mathcal{X}_{\mathrm{p},0} = [-1; 1]^{n_\mathrm{p}}$ are assumed. The time window is $\pm c$ percent of the control period, where c is an adjustable parameter: $\overline{\Delta}t_{\mathrm{u},y} = -\underline{\Delta}t_{\mathrm{u},y} = 0.01cT\begin{bmatrix}1 & 1 & \dots\end{bmatrix}^\mathsf{T}$. Multiple variants are provided:

C1, D1 ideal timing: $c = 0$.

C2, D2 The timing may vary by $c = 1$ percent.

C3 – C5 Modified, see Section 5.2.6.2.

D3 This is a parametric version of the example. The percentage (c %) must be specified explicitly.

B Stability Analysis using Linear Impulsive Systems

This appendix gives details on the proofs and implementation of the LIS-based stability analysis of Section 5.1. A previous version was published in Gaukler, Roppenecker, et al., 2020 and the corresponding technical report (Gaukler, Roppenecker, et al., 2019).

B.1 Foundations

Proof of Theorem 5.1.7. Assume DGES(ρ, α) with any $\alpha > 0$. Then, for all k and $x_0 \neq 0$,

$$|x_k| \leq \alpha \rho^k |x_0| \quad \Rightarrow \quad \frac{|A^k x_0|}{|x_0|} \leq \alpha \rho^k. \tag{322}$$

Choose the x_0 that maximizes the left side:

$$\underbrace{\max_{x_0 \neq 0} \frac{|A^k x_0|}{|x_0|}}_{\|A^k\|_2 \text{ (Definition 4.1.6)}} \leq \alpha \rho^k. \tag{323}$$

Let $k > 0$. Rewrite the inequality and take the limit $k \to \infty$:

$$\underbrace{\|A^k\|_2^{1/k}}_{\to \rho\{A\}} \leq \underbrace{\alpha^{1/k}}_{\to 1} \rho. \tag{324}$$

(Bernstein, 2009, Proposition 9.2.6)

Hence, $\rho\{A\} \leq \rho$. □

Proof of Theorem 5.1.17. Assume a fixed A and $\bar{\rho}$ with $\rho\{A\} < 1$ and $\rho\{A\} < \bar{\rho} < 1$.

Consider the "destabilized" system $\tilde{x}_{k+1} = A\bar{\rho}^{-1}\tilde{x}_k$, for which all eigenvalues and therefore the spectral radius are scaled by $\bar{\rho}^{-1}$. It is still stable but almost unstable for $\bar{\rho} \to \rho\{A\}^+$:

$$\rho\{A\bar{\rho}^{-1}\} = \bar{\rho}^{-1}\rho\{A\} \in (\rho\{A\}; 1). \tag{325}$$

Applying Theorem 5.1.13 to $A\bar{\rho}^{-1}$ and any $Q \succ 0$ shows that there is a P with

$$V_P(\bar{\rho}^{-1}Ax) \prec V_P(x). \quad \Rightarrow \quad \bar{\rho}^{-2}V_P(Ax) \prec V_P(x) \tag{326}$$

$$\Rightarrow \quad \frac{V_P(Ax)}{V_P(x)} \prec \bar{\rho}^2 \quad \forall x \neq 0 \quad \Rightarrow \quad \sqrt{\frac{V_P(Ax)}{V_P(x)}} \leq \bar{\rho} \quad \forall x \neq 0 \tag{327}$$

$$\Rightarrow \quad \|A\|_P \leq \bar{\rho}. \tag{328}$$

Note that the resulting $V_P(x)$ is a Lyapunov function for both $A\bar{\rho}^{-1}$ and A. □

Proof of Remark 5.1.18. Due to Theorems 5.1.14 and 5.1.16, $\|A\|_P \geq \rho\{A\}$ always holds. The remainder of this proof is an example that "=" is not always possible:

$$\text{For } A = \begin{bmatrix} \rho & 1 \\ 0 & \rho \end{bmatrix} \text{ with } 0 < \rho < 1, \quad \|A\|_P \neq \rho\{A\} \quad \forall P \succ 0. \tag{329}$$

Assume A as given in the previous equation. Here, $\rho\{A\} = \rho < 1$. All $P \succ 0$ can be parameterized using Theorem 5.1.3 as

$$P = P^{1/2}(P^{1/2})^\top \text{ with } P^{1/2} = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix}, \; a > 0, \; c > 0, \; b \in \mathbb{R}. \tag{330}$$

By Theorem 5.1.15 and Definition 4.1.6,

$$\|A\|_P = \| \underbrace{(P^{1/2})^\top A (P^{1/2})^{-\top}}_{M} \|_2 = \max\{\sigma_1, \sigma_2\}, \tag{331}$$

where σ_i^2 are the eigenvalues of $M^\top M$, which are the solutions of

$$0 = \det(M^\top M - \sigma_i^2 I). \tag{332}$$

$$\Leftrightarrow \cdots \Leftrightarrow 0 = {\sigma_i}^4 - \underbrace{\left(\frac{a^2}{c^2} + 2\rho^2\right)}_{=:d > 2\rho^2 > 0} {\sigma_i}^2 + \rho^4 \tag{333}$$

$$\Leftrightarrow \sigma_i^2 = \frac{\overbrace{d}^{>2\rho^2} \pm (\overbrace{d^2 - 4\rho^4}^{>0})^{1/2}}{2} \quad \Rightarrow \max_{i \in \{1,2\}} \sigma_i^2 > \rho^2 \tag{334}$$

$$\Rightarrow \|A\|_P > \rho\{A\} \quad \text{for all possible } P. \tag{335}$$

In the limit $\|A\|_P \to \rho\{A\}^+$, this requires $d \to 2\rho^2$, resulting in $c \to \infty$ or $a \to 0$. Then, $P^{1/2}$ or $(P^{1/2})^{-1}$ become numerically problematic. This motivates that a numerical solution for P should stay away from this limit but rather keep some distance $\|A\|_P - \rho\{A\} > 0$ to ensure numerical robustness.

This example was validated by computer-aided symbolic computations and a numerical experiment, see Appendix F for the source code. □

B.2 Decomposition Theorem

This section contains the proofs that lead to the decomposition theorem (Theorem 5.1.4). This theorem states that the transition matrix A_k can be split into summands that depend on at most two timing variables.

B.2.1 Properties of Measurement and Actuation Event Matrices

To rewrite the transition matrix, structural properties of the event matrices $E_{\mathrm{u},i}$ and $E_{\mathrm{y},i}$ from Section 4.1.3 are used. These properties follow directly from block matrix multiplication. For each of the properties, a loose interpretation will be given, which is not to be taken as a formal statement on its own.

Notation In the following, $\forall i$ is shorthand for $\forall i \in \{1, \dots, m\}$ if it refers to $E_{\mathrm{u},i}$, and $\forall i \in \{1, \dots, p\}$ for $E_{\mathrm{y},i}$. The same holds for $\forall j$. Similarly, $\forall \delta$ is shorthand for $\forall \delta \in \mathbb{R}$. The notation $E_{a,\dots} = \dots \forall a \in \{\text{“u”}, \text{“y”}\}$ means that an equation is valid for both $E_{\mathrm{u},\dots}$ and $E_{\mathrm{y},\dots}$.

Properties of a Single Event

Lemma B.1. *Actuation is unaffected by prior delays, as*

$$(E_{\mathrm{u},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta} \overset{(40),(41)}{=} E_{\mathrm{u},i} - I \quad \forall i, \delta, \tag{336}$$

whereas measurement is unaffected by subsequent *delays:*

$$\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{\mathrm{y},i} - I) \overset{(40),(43)}{=} (E_{\mathrm{y},i} - I) \quad \forall i, \delta. \tag{337}$$

However, measurement *is* affected by prior delays, as

$$(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta} \overset{(40),(43)}{=} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ S_i C_\mathrm{p} \mathrm{e}^{A_\mathrm{p}\delta} & 0 & -S_i & S_i C_\mathrm{p} \int_0^\delta \mathrm{e}^{A_\mathrm{p}\xi} d\xi B_\mathrm{p} \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \forall i, \delta. \tag{338}$$

Properties of Two Subsequent Events

Lemma B.2 (Zero Products)**.** *Products of the form* $(E_{\dots} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{\dots} - I)$ *are zero, as long as the events are distinct and the combination is not "actuate, then measure":*

$$(E_{a,i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{b,j} - I) = 0 \quad \forall (a,i) \neq (b,j),\ \forall \delta, \\ \forall (a,b) \in \{\text{“u”}, \text{“y”}\}^2 \setminus \{(\text{“y”}, \text{“u”})\}. \tag{339}$$

Additionally, for $\delta = 0$*, i.e., no delay between the events, this product is always zero:*

$$(E_{a,i} - I)(E_{b,j} - I) = 0 \quad \forall (a,i) \neq (b,j), \quad \forall a, b \in \{\text{“u”}, \text{“y”}\}. \tag{340}$$

Proof. The lemma directly follows from block matrix computations for each case. First, note that

$$S_i S_j \overset{(42)}{=} 0 \quad \forall i \neq j. \tag{341}$$

For $i \neq j$, actuation of $u^{(j)}$ does not affect the subsequent actuation of $u^{(i)}$:

$$\underbrace{(E_{\mathrm{u},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}}_{\overset{(336)}{=} E_{\mathrm{u},i} - I}(E_{\mathrm{u},j} - I) \overset{(41)}{=} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -S_i S_j C_\mathrm{d} & 0 & S_i S_j \end{bmatrix} \overset{(341)}{=} 0 \quad \forall i \neq j, \forall \delta. \tag{342}$$

Measurement does not affect subsequent actuation:

$$(E_{\mathrm{u},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{\mathrm{y},j} - I) \overset{(336),(41),(43)}{=} 0 \quad \forall i, j, \delta. \tag{343}$$

For $j \neq i$, measuring $y^{(i)}$ does not affect a subsequent measurement of $y^{(j)}$:

$$(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{\mathrm{y},j} - I) \overset{(337),(43)}{=} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -S_i S_j C_{\mathrm{p}} & 0 & S_i S_j & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \overset{(341)}{=} 0 \quad \forall i \neq j, \forall \delta. \tag{344}$$

However, actuation *does* affect subsequent measurements, i.e.,

$$(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{\mathrm{u},j} - I) \overset{(338),(41)}{=} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & S_i C_{\mathrm{p}} \int_0^\delta \mathrm{e}^{A_{\mathrm{p}}\xi} d\xi B_{\mathrm{p}} S_j C_{\mathrm{d}} & 0 & -S_i C_{\mathrm{p}} \int_0^\delta \mathrm{e}^{A_{\mathrm{p}}\xi} d\xi B_{\mathrm{p}} S_j \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \forall i, j, \delta \tag{345}$$

can be nonzero, except if the measurement happens immediately after actuation:

$$(E_{\mathrm{y},i} - I)(E_{\mathrm{u},j} - I) \overset{(41),(43)}{=} 0 \quad \forall i, j. \tag{346}$$

□

Lemma B.3 (Commutativity)**.** *Measurement and actuation events commute:*

$$\forall i, j, \quad \forall a, b \in \{\text{“u”}, \text{“y”}\} : E_{a,i}E_{b,j} = E_{b,j}E_{a,i}. \tag{347}$$

Proof. For $(a, i) = (b, j)$, this is trivially true. Now consider $(a, i) \neq (b, j)$:

$$\begin{aligned} &\forall a, b \in \{\text{“u”}, \text{“y”}\}, \quad \forall (a, i) \neq (b, j) : \\ &E_{a,i}E_{b,j} = ((E_{a,i} - I) + I)((E_{b,j} - I) + I) \\ &\qquad = I + (E_{a,i} - I) + (E_{b,j} - I) + \underbrace{(E_{a,i} - I)(E_{b,j} - I)}_{0 \text{ due to } (340)} \end{aligned} \tag{348a}$$

$$\begin{aligned} E_{b,j}E_{a,i} &\overset{(348a)}{=} I + (E_{b,j} - I) + (E_{a,i} - I) + 0 \\ &\overset{(348a)}{=} E_{a,i}E_{b,j}. \end{aligned} \tag{348b}$$

□

Extension to Three and More Events The result (339) leads to a property of the longer chain $(E_{\dots} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{\dots} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta}(E_{\dots} - I)$, again assuming distinct events.

Lemma B.4 (Long Products are Zero)**.**

$$\begin{aligned}(E_{a,i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}(E_{b,j} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{c,k} - I) = 0 \\ \forall (a,b,c) \in \{\text{“u”}, \text{“y”}\}^3, \\ \forall (a,i) \neq (b,j) \neq (c,k),\ (a,i) \neq (c,k), \\ \forall \delta_1, \delta_2. \end{aligned} \tag{349}$$

The result implies that any such product of length three and above is zero.

Proof. There are 2^3 possibilities for the triple (a,b,c). For each, the chain contains at least one product that is zero due to (339):

1. $\underbrace{(E_{\mathrm{u},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}(E_{\mathrm{u},j} - I)}_{0}\,\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{u},k} - I) = 0$
2. $\underbrace{(E_{\mathrm{u},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}(E_{\mathrm{u},j} - I)}_{0}\,\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{y},k} - I) = 0$
3. $\underbrace{(E_{\mathrm{u},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}(E_{\mathrm{y},j} - I)}_{0}\,\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{u},k} - I) = 0$
4. $\underbrace{(E_{\mathrm{u},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}(E_{\mathrm{y},j} - I)}_{0}\,\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{y},k} - I) = 0$
5. $(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}\underbrace{(E_{\mathrm{u},j} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{u},k} - I)}_{0} = 0$
6. $(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}\underbrace{(E_{\mathrm{u},j} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{y},k} - I)}_{0} = 0$
7. $\underbrace{(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}(E_{\mathrm{y},j} - I)}_{0}\,\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{u},k} - I) = 0$
8. $\underbrace{(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}\delta_1}(E_{\mathrm{y},j} - I)}_{0}\,\mathrm{e}^{A_{\mathrm{cont}}\delta_2}(E_{\mathrm{y},k} - I) = 0$ □

B.2.2 General Expansion of Binomial Products

Subsequent proofs require a generic way to expand a product of binomials, such as

$$(A_3 + B_3)(A_2 + B_2)(A_1 + B_1) \\ = A_3A_2A_1 + A_3A_2B_1 + A_3B_2A_1 + A_3B_2B_1 + \cdots + B_3B_2B_1. \quad (350)$$

If "A" and "B" are interpreted as binary digits, where "A" is 0 and "B" is 1, the above sequence of summands is generated by counting in binary: 000 is $A_3A_2A_1$, 001 is $A_3A_2B_1$, 010 is $A_3B_2A_1$, ..., up to 111. To generalize the notation, these binary numbers are interpreted as a vector $\begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}^\top$ of binary digits $b_i \in \{0, 1\}$:

Lemma B.5. *Let $A_1, \ldots, A_N, B_1, \ldots, B_N \in \mathbb{R}^{n\times n}$ and $N \in \mathbb{N}_1$. Then,*

$$\widetilde{\prod_{i=1}^{N}}(A_i + B_i) = \sum_{[\,b_1\ b_2\ \ldots\ b_N\,]^\top \in \{0,1\}^N} \widetilde{\prod_{i=1}^{N}} \begin{cases} A_i, & b_i = 0, \\ B_i, & b_i = 1. \end{cases} \quad (351)$$

Note that the lemma is given for the reversed product $\widetilde{\Pi}$ but also valid for the normal product Π.

Proof. For a proof by induction, start with the observation that the lemma trivially holds for $N = 1$. As induction step, assume that the lemma holds for a fixed $N \geq 1$ to show that it holds for $N + 1$:

$$\widetilde{\prod_{i=1}^{N+1}}(A_i + B_i) = (A_{N+1} + B_{N+1}) \widetilde{\prod_{i=1}^{N}}(A_i + B_i) \quad (352)$$

$$\overset{\text{assume claim for } N}{=} A_{N+1} \sum_{[\,b_1\ b_2\ \ldots\ b_N\,]^\top \in \{0,1\}^N} \widetilde{\prod_{i=1}^{N}} \begin{cases} A_i, & b_i = 0 \\ B_i, & b_i = 1 \end{cases} \\ + B_{N+1} \sum_{[\,b_1\ b_2\ \ldots\ b_N\,]^\top \in \{0,1\}^N} \widetilde{\prod_{i=1}^{N}} \begin{cases} A_i, & b_i = 0 \\ B_i, & b_i = 1 \end{cases} \quad (353)$$

$$= \sum_{[\,b_1\ b_2\ \dots\ b_{N+1}\,]^\top \in \{0,1\}^N \times \{0\},} \widetilde{\prod_{i=1}^{N+1}} \begin{cases} A_i, & b_i = 0 \\ B_i, & b_i = 1 \end{cases}$$

$$+ \sum_{[\,b_1\ b_2\ \dots\ b_{N+1}\,]^\top \in \{0,1\}^N \times \{1\}} \widetilde{\prod_{i=1}^{N+1}} \begin{cases} A_i, & b_i = 0 \\ B_i, & b_i = 1 \end{cases} \tag{354}$$

$$= \sum_{[\,b_1\ b_2\ \dots\ b_{N+1}\,]^\top \in \{0,1\}^{N+1}} \widetilde{\prod_{i=1}^{N+1}} \begin{cases} A_i, & b_i = 0, \\ B_i, & b_i = 1. \end{cases} \tag{355}$$

By induction, the lemma holds for any $N \geq 1$. □

B.2.3 Proof of the Decomposition Theorem

Proof of Theorem 5.1.4. Assume that no events are skipped. Consider the transition matrix (56) of the k-th control period, as given in Section 4.1.4:

$$A(\Delta t) = E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev}}} - \tau_{N_{\text{ev}}-1})} \underbrace{\widetilde{\prod_{i=1}^{N_{\text{ev}}-1}} E_i \mathrm{e}^{A_{\text{cont}}(\tau_i - \tau_{i-1})}}_{=:X}. \tag{356}$$

Note that the period starts with the event counter $i = 0$ at $t = \tau_0 \overset{(51)}{=} kT - T/2$ and ends after event $i = N_{\text{ev}} \overset{(55)}{=} m + p + 1$ at $t_k = \tau_{N_{\text{ev}}} \overset{(52)}{=} t_{\text{ctrl},k} = kT + T/2$.

The factor X in the above transition matrix only contains measurement and actuation events: In the following analysis of X, all matrices E_i are either $E_i = E_{\text{u},\dots}$ or $E_i = E_{\text{y},\dots}$.

X can be rewritten as

$$X = \widetilde{\prod_{i=1}^{N_{\text{ev}}-1}} (I \mathrm{e}^{A_{\text{cont}}(\tau_i - \tau_{i-1})} + (E_i - I)\mathrm{e}^{A_{\text{cont}}(\tau_i - \tau_{i-1})}) \tag{357}$$

$$\overset{\text{Lemma B.5}}{=} \sum_{[\,b_1\ b_2\ \dots\,]^\top \in \{0,1\}^{N_{\text{ev}}-1}} \widetilde{\prod_{i=1}^{N_{\text{ev}}-1}} \left(\begin{cases} I, & b_i = 0 \\ (E_i - I), & b_i = 1 \end{cases} \cdot \mathrm{e}^{A_{\text{cont}}(\tau_i - \tau_{i-1})} \right). \tag{358}$$

This sum can be split by $\sum_i b_i$, the amount of how often a factor $(E_i - I)$ appears, to then apply the event matrix properties proven earlier:

$$
\begin{aligned}
X = & \overbrace{\prod_{i=1}^{N_{\mathrm{ev}}-1}} I \mathrm{e}^{A_{\mathrm{cont}}(\tau_i - \tau_{i-1})} \\
& + \sum_{j=1}^{N_{\mathrm{ev}}-1} \left(\left(\overbrace{\prod_{i=j+1}^{N_{\mathrm{ev}}-1}} I \mathrm{e}^{A_{\mathrm{cont}}(\tau_i - \tau_{i-1})} \right) (E_j - I) \mathrm{e}^{A_{\mathrm{cont}}(\tau_j - \tau_{j-1})} \right. \\
& \qquad \left. \cdot \left(\overbrace{\prod_{i=1}^{j-1}} I \mathrm{e}^{A_{\mathrm{cont}}(\tau_i - \tau_{i-1})} \right) \right) \\
& + \underbrace{\sum_{[\,b_1\ b_2\ \ldots\,]^\top \in \{0,1\}^{N_{\mathrm{ev}}-1},\, \sum_i b_i = 2} \overbrace{\prod_{i=1}^{N_{\mathrm{ev}}-1}} \left(\begin{cases} I, & b_i = 0 \\ (E_i - I), & b_i = 1 \end{cases} \cdot \mathrm{e}^{A_{\mathrm{cont}}(\tau_i - \tau_{i-1})} \right)}_{\substack{\text{all summands except for the combination } \ldots (E_{\mathrm{y},i} - I) \mathrm{e}^{A_{\mathrm{cont}} \delta} \ldots (E_{\mathrm{u},j} - I) \ldots \\ \text{are } = 0 \text{ due to (339) and } \mathrm{e}^{A_{\mathrm{cont}} \delta_0} I \mathrm{e}^{A_{\mathrm{cont}} \delta_1} = \mathrm{e}^{A_{\mathrm{cont}}(\delta_0 + \delta_1)}}} \\
& + \underbrace{\sum_{[\,b_1\ b_2\ \ldots\,]^\top \in \{0,1\}^{N_{\mathrm{ev}}-1},\, \sum_i b_i \geq 3} \overbrace{\prod_{i=1}^{N_{\mathrm{ev}}-1}} \left(\begin{cases} I, & b_i = 0 \\ (E_i - I), & b_i = 1 \end{cases} \cdot \mathrm{e}^{A_{\mathrm{cont}}(\tau_i - \tau_{i-1})} \right)}_{\substack{= 0 \text{ due to (349) and } \mathrm{e}^{A_{\mathrm{cont}} \delta_0} I \mathrm{e}^{A_{\mathrm{cont}} \delta_1} = \mathrm{e}^{A_{\mathrm{cont}}(\delta_0 + \delta_1)}}}
\end{aligned}
\tag{359}
$$

$$
\begin{aligned}
&= \overbrace{\prod_{i=1}^{N_{\mathrm{ev}}-1}} \mathrm{e}^{A_{\mathrm{cont}}(\tau_i-\tau_{i-1})} \\
&\quad + \sum_{j=1}^{N_{\mathrm{ev}}-1} \left(\left(\overbrace{\prod_{i=j+1}^{N_{\mathrm{ev}}-1}} I\mathrm{e}^{A_{\mathrm{cont}}(\tau_i-\tau_{i-1})} \right) (E_j - I)\mathrm{e}^{A_{\mathrm{cont}}(\tau_j-\tau_{j-1})} \right. \\
&\qquad\qquad \left. \cdot \left(\overbrace{\prod_{i=1}^{j-1}} I\mathrm{e}^{A_{\mathrm{cont}}(\tau_i-\tau_{i-1})} \right) \right) \\
&\quad + \sum_{i=1}^{p}\sum_{j=1}^{m} \begin{cases} 0, & t_{\mathrm{u}}^{(j)} \geq t_{\mathrm{y}}^{(i)}, \\ \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-t_{\mathrm{y}}^{(i)})}(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}(t_{\mathrm{y}}^{(i)}-t_{\mathrm{u}}^{(j)})} & \\ \qquad \cdot (E_{\mathrm{u},j} - I) \underbrace{\mathrm{e}^{A_{\mathrm{cont}}(t_{\mathrm{u}}^{(j)}-\tau_0)}}_{\text{omit due to (336)}}, & \text{else} \end{cases}
\end{aligned} \tag{360}
$$

$$
\begin{aligned}
&= \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-\tau_0)} + \sum_{j=1}^{N_{\mathrm{ev}}-1} \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-\tau_j)}(E_j - I)\mathrm{e}^{A_{\mathrm{cont}}(\tau_j-\tau_0)} \\
&\quad + \sum_{i=1}^{p}\sum_{j=1}^{m} \begin{cases} 0, & t_{\mathrm{u}}^{(j)} \geq t_{\mathrm{y}}^{(i)}, \\ \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-t_{\mathrm{y}}^{(i)})}(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}(t_{\mathrm{y}}^{(i)}-t_{\mathrm{u}}^{(j)})}(E_{\mathrm{u},j} - I), & \text{else} \end{cases}
\end{aligned} \tag{361}
$$

$$
\begin{aligned}
&= \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-\tau_0)} \\
&\quad + \sum_{i=1}^{m} \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-t_{\mathrm{u}}^{(i)})}(E_{\mathrm{u},i} - I) \underbrace{\mathrm{e}^{A_{\mathrm{cont}}(t_{\mathrm{u}}^{(i)}-\tau_0)}}_{\text{omit due to (336)}} \\
&\quad + \sum_{j=1}^{p} \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-t_{\mathrm{y}}^{(j)})}(E_{\mathrm{y},j} - I)\mathrm{e}^{A_{\mathrm{cont}}(t_{\mathrm{y}}^{(j)}-\tau_0)} \\
&\quad + \sum_{i=1}^{p}\sum_{j=1}^{m} \begin{cases} 0, & t_{\mathrm{u}}^{(j)} \geq t_{\mathrm{y}}^{(i)}, \\ \mathrm{e}^{A_{\mathrm{cont}}(\tau_{N_{\mathrm{ev}-1}}-t_{\mathrm{y}}^{(i)})}(E_{\mathrm{y},i} - I)\mathrm{e}^{A_{\mathrm{cont}}(t_{\mathrm{y}}^{(i)}-t_{\mathrm{u}}^{(j)})}(E_{\mathrm{u},j} - I), & \text{else.} \end{cases}
\end{aligned} \tag{362}
$$

Due to this splitting of X, $A_k = E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev}}} - \tau_{N_{\text{ev}}-1})} X$ can be rewritten as

$$\begin{aligned} A_k \overset{(110),(362)}{=} & E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev}}} - \tau_0)} \\ & + \sum_{i=1}^{m} E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev}}} - t_{\text{u}}^{(i)})} (E_{\text{u},i} - I) \\ & + \sum_{j=1}^{p} E_{\text{ctrl}} \underbrace{\mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev}}} - t_{\text{y}}^{(j)})}}_{\text{omit due to (337)}} (E_{\text{y},j} - I) \mathrm{e}^{A_{\text{cont}}(t_{\text{y}}^{(j)} - \tau_0)} \\ & + \sum_{i=1}^{p} \sum_{j=1}^{m} \begin{cases} 0, & t_{\text{u}}^{(j)} \geq t_{\text{y}}^{(i)}, \\ E_{\text{ctrl}} \underbrace{\mathrm{e}^{A_{\text{cont}}(\tau_{N_{\text{ev}}} - t_{\text{y}}^{(j)})}}_{\text{omit due to (337)}} (E_{\text{y},i} - I) \mathrm{e}^{A_{\text{cont}}(t_{\text{y}}^{(i)} - t_{\text{u}}^{(j)})} (E_{\text{u}}^{(j)} - I), & \text{else.} \end{cases} \end{aligned} \tag{363}$$

With $t_{\{u,y\}}^{(i)} \overset{(18),(19)}{=} \Delta t_{\{u,y\}}^{(i)} + kT$, this becomes

$$\begin{aligned} A_k = & E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}} T} \\ & + \sum_{i=1}^{m} \underbrace{E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}}(T/2 - \Delta t_{\text{u}}^{(i)})} (E_{\text{u},i} - I)}_{=: M_{\text{u},i}(\Delta t_{\text{u}}^{(i)})} \\ & + \sum_{j=1}^{p} \underbrace{E_{\text{ctrl}} (E_{\text{y},j} - I) \mathrm{e}^{A_{\text{cont}}(T/2 + \Delta t_{\text{y}}^{(j)})}}_{=: M_{\text{y},j}(\Delta t_{\text{y}}^{(j)})} \\ & + \sum_{j=1}^{p} \sum_{i=1}^{m} \underbrace{\begin{cases} 0, & \Delta t_{\text{y}}^{(j)} - \Delta t_{\text{u}}^{(i)} \leq 0, \\ E_{\text{ctrl}} (E_{\text{y},j} - I) \mathrm{e}^{A_{\text{cont}}(\Delta t_{\text{y}}^{(j)} - \Delta t_{\text{u}}^{(i)})} (E_{\text{u},i} - I), & \text{else} \end{cases}}_{=: M_{\text{uy},i,j}(\Delta t_{\text{y}}^{(j)} - \Delta t_{\text{u}}^{(i)})} \end{aligned} \tag{364}$$

$$\begin{aligned} = & E_{\text{ctrl}} \mathrm{e}^{A_{\text{cont}} T} + \sum_{i=1}^{m} M_{\text{u},i}(\Delta t_{\text{u}}^{(i)}) + \sum_{i=1}^{p} M_{\text{y},i}(\Delta t_{\text{y}}^{(i)}) \\ & + \sum_{i=1}^{m} \sum_{j=1}^{p} M_{\text{uy},i,j}(\Delta t_{\text{y}}^{(j)} - \Delta t_{\text{u}}^{(i)}). \end{aligned} \tag{365}$$

Setting this equal to the desired result of Theorem 5.1.4,

$$A_k = A(\Delta t = 0) + \sum_{i=1}^{m} \Delta A_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)}) + \sum_{j=1}^{p} \Delta A_{\mathrm{y},j}(\Delta t_{\mathrm{y}}^{(j)}) + \sum_{i=1}^{m}\sum_{j=1}^{p} \Delta A_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}), \tag{366}$$

leads to

$$A(\Delta t = 0) = A_k|_{\Delta t_{\mathrm{y}}=0, \Delta t_{\mathrm{u}}=0} \tag{367}$$

$$= E_{\mathrm{ctrl}} \mathrm{e}^{A_{\mathrm{cont}} T} + E_{\mathrm{ctrl}} \mathrm{e}^{A_{\mathrm{cont}} T/2} \left(\sum_{i=1}^{m} (E_{\mathrm{u},i} - I) \right) + E_{\mathrm{ctrl}} \left(\sum_{j=1}^{p} (E_{\mathrm{y},j} - I) \right) \mathrm{e}^{A_{\mathrm{cont}} T/2} \tag{368}$$

$$\overset{(336),(337)}{=} E_{\mathrm{ctrl}} \mathrm{e}^{A_{\mathrm{cont}} T/2} \left(I + \sum_{i=1}^{m} (E_{\mathrm{u},i} - I) + \sum_{j=1}^{p} (E_{\mathrm{y},j} - I) \right) \cdot \mathrm{e}^{A_{\mathrm{cont}} T/2}, \tag{369}$$

$$\Delta A_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)}) = M_{\mathrm{u},i}(\Delta t_{\mathrm{u}}^{(i)}) - M_{\mathrm{u},i}(0) \tag{370}$$

$$\overset{(364)}{=} E_{\mathrm{ctrl}} \mathrm{e}^{A_{\mathrm{cont}} T/2} (\mathrm{e}^{-A_{\mathrm{cont}} \Delta t_{\mathrm{u}}^{(i)}} - I)(E_{\mathrm{u},i} - I), \tag{371}$$

$$\Delta A_{\mathrm{y},j}(\Delta t_{\mathrm{y}}^{(j)}) = M_{\mathrm{y},j}(\Delta t_{\mathrm{y}}^{(j)}) - M_{\mathrm{y},j}(0) \tag{372}$$

$$\overset{(364)}{=} E_{\mathrm{ctrl}} (E_{\mathrm{y},j} - I) \mathrm{e}^{A_{\mathrm{cont}} T/2} (\mathrm{e}^{A_{\mathrm{cont}} \Delta t_{\mathrm{y}}^{(j)}} - I), \tag{373}$$

$$\Delta A_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}) = M_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}) - \underbrace{M_{\mathrm{uy},i,j}(0)}_{0 \text{ due to } (364)} \tag{374}$$

$$\overset{(364)}{=} \begin{cases} 0, & \Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)} \leq 0, \\ E_{\mathrm{ctrl}} (E_{\mathrm{y},j} - I) \mathrm{e}^{A_{\mathrm{cont}} (\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)})} (E_{\mathrm{u},i} - I), & \text{else.} \end{cases} \tag{375}$$

This proves the key equation of Theorem 5.1.4. With

$$\begin{aligned}(E_{\mathrm{y},j} - I)&(M - I)(E_{\mathrm{u},i} - I) \\ &= (E_{\mathrm{y},j} - I)M(E_{\mathrm{u},i} - I) - \underbrace{(E_{\mathrm{y},j} - I)(E_{\mathrm{u},i} - I)}_{=0 \text{ due to } (346)} \quad \forall M \in \mathbb{R}^{n\times n}, \end{aligned} \tag{376}$$

$\Delta A_{\mathrm{uy},i,j}$ can be rewritten as

$$\Delta A_{\mathrm{uy},i,j}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}) = \begin{cases} 0, & \Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)} \leq 0, \\ E_{\mathrm{ctrl}}(E_{\mathrm{y},j} - I)(\mathrm{e}^{A_{\mathrm{cont}}(\Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)})} - I) & \\ \cdot(E_{\mathrm{u},i} - I), & \text{else.} \end{cases} \tag{377}$$

Now, all $\Delta A_{\dots}$ are of the form

$$\Delta A_{\dots} = \begin{cases} M_1(\mathrm{e}^{A_{\mathrm{cont}}\delta(\Delta t)} - I)M_2, & \text{some condition}, \\ 0, & \text{else}, \end{cases} \tag{378}$$

where $M_{1,2} \in \mathbb{R}^{n\times n}$ are constants and δ is a scalar variable that depends on Δt such that $\lim_{|\Delta t|\to 0} \delta = 0$. Therefore,

$$\lim_{|\Delta t|\to 0} \Delta A_{\dots} = \left\{ \begin{cases} M_1(\mathrm{e}^0 - I)M_2, & \text{some condition} \\ 0, & \text{else} \end{cases} \right\} = 0. \tag{379}$$

This concludes the proof of Theorem 5.1.4. □

B.2.4 Parameters for Norm Bound

For a standardized implementation of the norm bounds, the form

$$\max_{\Delta t} \|\Delta A_{\dots}(\dots)\|_P \leq \max_{|\delta|<\bar{\delta}} \|M_1(\mathrm{e}^{A\delta} - I)M_2\|_P \tag{380}$$

is used in Section 5.1.6, where $\max_{\Delta t}$ denotes the maximum over the possible timing range. The parameters A, M_1, δ, $\bar{\delta}$ and M_2 are listed in the following. A is always A_{cont}, while the other parameters depend on which $\Delta A_{\dots}$ is considered.

For $\Delta A_{\mathrm{y},j}$, this is straightforward:

$$\Delta A_{\mathrm{y},j}(\underbrace{\Delta t_{\mathrm{y}}^{(j)}}_{\delta}) \overset{(373)}{=} \underbrace{E_{\mathrm{ctrl}}(E_{\mathrm{y},j} - I)\mathrm{e}^{A_{\mathrm{cont}}T/2}}_{M_1}(\mathrm{e}^{A_{\mathrm{cont}}\overbrace{\Delta t_{\mathrm{y}}^{(j)}}^{\delta}} - I)\underbrace{I}_{M_2} \tag{381}$$

with $\bar{\delta} = \max_{\Delta t} |\delta| = \max\{|\underline{\Delta t}_{\mathrm{y}}^{(j)}|, |\overline{\Delta t}_{\mathrm{y}}^{(j)}|\}$.

For $\Delta A_{\mathrm{u},i}$, the result is

$$\Delta A_{\mathrm{u},i}(\underbrace{\Delta t_{\mathrm{u}}^{(i)}}_{-\delta}) \overset{(371)}{=} \underbrace{E_{\mathrm{ctrl}} \mathrm{e}^{A_{\mathrm{cont}} T/2}}_{M_1} (\mathrm{e}^{A_{\mathrm{cont}} \cdot \overbrace{(-1)\cdot \Delta t_{\mathrm{u}}^{(i)}}^{\delta}} - I) \underbrace{(E_{\mathrm{u},i} - I)}_{M_2}, \tag{382}$$

with $\bar{\delta} = \max_{\Delta t} |\delta| = \max\{|\underline{\Delta t}_{\mathrm{u}}^{(i)}|, |\overline{\Delta t}_{\mathrm{u}}^{(i)}|\}$.

For $\Delta A_{\mathrm{uy},i,j}$, choosing $\delta = \Delta t_{\mathrm{y}}^{(j)} - \Delta t_{\mathrm{u}}^{(i)}$ yields

$$\Delta A_{\mathrm{uy},i,j}(\delta) \overset{(377)}{=} \begin{cases} 0, & \delta \leq 0, \\ \underbrace{E_{\mathrm{ctrl}}(E_{\mathrm{y},j} - I)}_{M_1} (\mathrm{e}^{A_{\mathrm{cont}}(\delta)} - I) \underbrace{(E_{\mathrm{u},i} - I)}_{M_2}, & \text{else.} \end{cases} \tag{383}$$

The negative range of δ is not relevant due to the zero case. Therefore, $\bar{\delta}$ is not chosen as $\max_{\Delta t} |\delta|$, but as the potentially smaller value

$$\bar{\delta} = \max_{\Delta t} \left(\begin{Bmatrix} 0, & \delta \leq 0 \\ \delta, & \text{else} \end{Bmatrix} \right) = \begin{cases} 0, & \overline{\Delta t}_{\mathrm{y}}^{(j)} \leq \underline{\Delta t}_{\mathrm{u}}^{(i)} \\ \overline{\Delta t}_{\mathrm{y}}^{(j)} - \underline{\Delta t}_{\mathrm{u}}^{(i)}, & \text{else} \end{cases} \tag{384}$$

that ignores negative δ.

B.3 Implementation

B.3.1 Heuristic Search for β

The following algorithm is used to optimize β in the optimization of P (Section 5.1.7.4):

1. Initially, $\beta = \frac{1}{4} \frac{1-\overline{\rho}}{m+p+mp}$ and $\kappa = 2$, where κ will be explained later.
2. Repeat the following three times:
   a) Compute P by (159).
   b) Compute the resulting stability bound $\tilde{\rho}$ by (140) and Section 5.1.6.
   c) In the exceptional case of $\|A(0)\|_P > 1$, the system is probably unstable. Then, retry with smaller β (or exit with error).
   d) If $\gamma < 10^{-5}$, update $\kappa := 0.45\kappa$ to increase numerical robustness.

e) Update $\beta := \kappa\beta \frac{1-\|A(0)\|_P}{\tilde{\rho}-\|A(0)\|_P}$.

3. Return the lowest $\tilde{\rho}$ found and the corresponding P.

For $\kappa = 1$, the update step 2e decreases β on $\tilde{\rho} > 1$ and increases it otherwise. Experiments suggest that a larger value of κ helps to speed up convergence and achieves lower $\tilde{\rho}$ but also lower robustness γ.

B.3.2 LMI Preconditioning

To improve speed and accuracy of the LMI solver, a state transformation $\tilde{A}(0) = R^{-1}A(0)\,R$ and $\Delta\tilde{\mathcal{A}} = \{R^{-1}DR | D \in \Delta\mathcal{A}\}$ is applied. By the definition (158) of the robustness parameter γ, the ideal numerical robustness $\gamma = 1$ would be achieved if the CQLF matrix after the transformation is $\tilde{P} = I$. Assuming $\Delta A_i \approx 0$ and $\bar{\rho} \approx 1$, $\tilde{P} = I$ is a solution if

$$\|\tilde{A}(0)\|_2 \overset{\text{Theorem 5.1.15}}{=} \|\tilde{A}(0)\|_{P=I} \overset{(154)}{<} \bar{\rho} \approx 1. \tag{385}$$

Therefore, R should be chosen such that $\|\tilde{A}(0)\|_2 < 1$.

The computations of Section 5.1.7.3 are denoted as $P_{\text{LMI}}(A(0), \Delta\mathcal{A}, \bar{\rho}, \beta)$. This original LMI is reused as follows:

1. Compute a quadratic Lyapunov function for the nominal case:

$$P_{\text{nominal}} = P_{\text{LMI}}(A(0) = A(0), \Delta\mathcal{A} = \{\}, \bar{\rho} = 1, \beta = 0). \tag{386}$$

   Therefore, assuming that the nominal case is stable and the numerical error is not too large, $\|A(0)\|_{P_{\text{nominal}}} < 1$. In practice, this value is ≈ 1.

2. Choose $R^{-1} = (P_{\text{nominal}}^{1/2})^\top$, which is nonsingular due to $P_{\text{nominal}} \succ 0$ and Theorem 5.1.3. Then, $\|\tilde{A}(0)\|_2 < 1$, as

$$\|A(0)\|_{P_{\text{nominal}}} \overset{\text{Theorem 5.1.15}}{=} \left\| \underbrace{(P_{\text{nominal}}^{1/2})^\top}_{R^{-1}} A(0) \underbrace{(P_{\text{nominal}}^{1/2})^{-\top}}_{R} \right\|_2 \tag{387}$$

$$= \|\tilde{A}(0)\|_2\,. \tag{388}$$

3. Compute $\tilde{P} = P_{\text{LMI}}(A(0) = \tilde{A}(0), \Delta\mathcal{A} = \Delta\tilde{\mathcal{A}}, \bar{\rho}, \beta)$.

4. Transform the resulting $\tilde{P}$ back to the original problem.

   Proposition: The inverse transform is

$$P = R^{-\top}\tilde{P}R^{-1}. \tag{389}$$

Proof: The LMI condition (154) concerning $\bar{\rho}$ holds unchanged, since

$$\begin{aligned}
\text{Section 5.1.7.3} \quad &\Rightarrow \quad \tilde{A}(0)^\top \tilde{P}\, \tilde{A}(0) \prec \bar{\rho}^2 \tilde{P} && (390)\\
&\Leftrightarrow \quad R^\top A(0)^\top R^{-\top} \tilde{P} R^{-1} A(0) R \prec \bar{\rho}^2 \tilde{P} && (391)\\
&\overset{(152)}{\Leftrightarrow} \quad A(0)^\top \underbrace{R^{-\top} \tilde{P} R^{-1}}_{P} A(0) \prec \bar{\rho}^2 R^{-\top} \tilde{P} R^{-1} && (392)\\
&\Leftrightarrow \quad A(0)^\top P A(0) \prec \bar{\rho}^2 P. && (393)
\end{aligned}$$

The same argument holds for the condition (155) for β. □

As the computation never uses P but only $P^{1/2}$, it is desirable to derive an inverse transform for the Cholesky decomposition.

Proposition: The Cholesky decomposition of the inverse transform (389) is $(P^{1/2})^\top = (\tilde{P}^{1/2})^\top R^{-1}$.

Proof: The statement is true because the Cholesky decomposition is unique (Theorem 5.1.3) and the proposed value of $P^{1/2}$ fulfills all three conditions of the definition of the Cholesky decomposition:

1. $P \succ 0$ due to (152) and $\tilde{P} \succ 0$.
2. $P^{1/2}$ fulfills $P^{1/2}(P^{1/2})^\top \overset{(389)}{=} P$, so it is either the Cholesky decomposition or a transformed (e.g., transposed) variant.
3. $(P^{1/2})^\top$ is upper triangular with positive diagonal entries (UT_+) because:
   - It is the product of UT_+ matrices: By the definition of the Cholesky decomposition, $(\tilde{P}^{1/2})^\top$ and $R^{-1} = (P_{\text{nominal}}^{1/2})^\top$ are UT_+.
   - The product of two UT_+ matrices is UT_+ (Bernstein, 2009, Fact 3.23.12.ii). □

B.3.3 Verified Numerical Implementation

A standard computer implementation of the LMI optimization uses floating-point arithmetic with finite precision and is therefore prone to numerical errors in P and the norm bounds. The danger of significant numerical error is exemplified by the fact that P can only be computed successfully if a pre-conditioning transformation is applied. Also, approximations were used in deriving the LMIs.

To ensure that the stability result is still valid, P is considered as an inaccurate guess and the resulting stability bound $\tilde{\rho}$ in (140) is verified in interval arithmetic. This leads to an overapproximated, i.e., pessimistic but guaranteed result. If $\tilde{\rho} < 1$, the system is stable. Otherwise, no conclusion can be drawn: The instability might be real or only apparent due to pessimism.

In detail, the computations implemented in interval arithmetic are $\|A(0)\|_P^{\mathrm{ub}}$ and the norm bounds of timing-dependent summands per Section 5.1.6. For this, the P-ellipsoidal norm is converted to a spectral norm by Theorem 5.1.15. The interval computation of the spectral norm is detailed later.

Regarding P, it is important to note that the final stability result is valid no matter how P was determined, as long as $P \succ 0$: The underlying theorems are valid for any P-ellipsoidal norm $\|\cdot\|_P$ with $P \succ 0$. In the implementation, the numerical result P is checked for $P \succ 0$ during the computation of $(P^{1/2})^{-1}$. Hence, approximations and numerical error in P can not lead to false-positive stability results.

The use of interval arithmetic has the advantage that small uncertainties in the plant model $A_\mathrm{p}, B_\mathrm{p}, C_\mathrm{p}$ and the period T can be explicitly considered. Because the result of the presented approach is a common quadratic Lyapunov function, the same stability result also holds if the uncertainties are time-varying (Theorem 5.1.19).

Interval Computation of the Spectral Norm Interval computations are implemented using the Python *mpmath* library, which does not directly support the spectral norm. Hence, the norm must be reduced to elementary computations. A simple upper bound for the spectral norm is

$$\|A\|_2 \leq \sqrt{\sum_{i,j} \left(A_{(i,j)}\right)^2} \quad \forall A \in \mathbb{R}^{n\times n}. \tag{394}$$

(Rump, 2010, p. 5). While this bound is rather pessimistic, it can be reused to derive a precise bound for general matrices:

Consider the singular value decomposition

$$\Sigma = U^\mathsf{T} A V, \tag{395}$$

where Σ is the diagonal matrix of singular values of A, i.e.,

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n), \tag{396}$$

and U, V are orthogonal matrices, i.e.,

$$UU^\top = U^\top U = I \quad \text{and} \quad VV^\top = V^\top V = I \tag{397}$$

(Bernstein, 2009, Theorem 5.6.3 and Fact 3.11.4). Let $\tilde{V}$ be a numerical approximation of V with unknown accuracy. All following computations must be in interval arithmetic and are due to Rump, 2010, Theorem 3.2. Note that ideally,

$$V^\top A^\top AV = V^\top A^\top \underbrace{UU^\top}_{I} AV = \Sigma^\top \Sigma = \operatorname{diag}(\sigma_1^2, \dots, \sigma_n^2). \tag{398}$$

This is approximated by computing $D + E = \tilde{V}^\top A^\top A\tilde{V} \approx \Sigma$, where $D = \operatorname{diag}(b_1, \dots, b_n) \approx \Sigma$ is the diagonal part of $\tilde{V}^\top A^\top A\tilde{V}$ and $E \approx 0$ the rest. To bound the error, apply (394) to compute α such that $\|I - \tilde{V}^\top \tilde{V}\|_2 \leq \alpha < 1$ and ϵ such that $\|E\|_2 < \epsilon$. Then,

$$\sqrt{\frac{\max_i b_i - \epsilon}{1 + \alpha}} \leq \|A\|_2 \leq \sqrt{\frac{\max_i b_i + \epsilon}{1 - \alpha}}. \tag{399}$$

C Continuization

This appendix presents proofs for the continuization-based analysis of Section 5.2. These proofs mostly use induction on the trajectories of hybrid automata. In principle, there exist specialized induction-like proof techniques for hybrid systems, such as equational certificates and differential invariants (Clarke et al., 2018, Section 30.6). However, to keep the proofs self-contained and accessible to a broader audience, such techniques are avoided here, which however requires more verbosity and some simplifying assumptions.

A previous version of this appendix appeared in Gaukler, 2020a.

C.1 Formal Basis

C.1.1 Abstraction Within Bound

Proof of Theorem 5.2.1. Assume the given conditions (166a) and (166b) are true. Consider any trajectory

$$(l_0, x_0) \xrightarrow{\beta_1} (l_1, x_1) \xrightarrow{\beta_2} \dots \xrightarrow{\beta_n} (l_n, x_n) \tag{400}$$

of H_1 with length $n \geq 0$. The proof idea is to show by induction that every transition $(l_i, x_i) \xrightarrow{\beta_{i+1}} (l_{i+1}, x_{i+1})$ of this trajectory also exists in H_2 and consequently $\text{Reach}_x(H_2) \supseteq \text{Reach}_x(H_1)$.

Induction Assumption IA(i): For given i, where $0 \leq i \leq n$, the first i transitions of the trajectory (400) of H_1 are also a valid trajectory of H_2, i.e.,

$$(l_0, x_0) \xrightarrow{\beta_1} (l_1, x_1) \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_i} (l_i, x_i) \tag{401}$$

is a trajectory of H_2. This implies that $(l_0, x_0) \in \mathit{Init}(H_2)$ and $x_i \in \text{Reach}_x(H_2)$.

Start of Induction H_1 and H_2 have the same initial set and invariant, so their trajectories start the same. Hence, IA(0) is true.

Induction Step Assume IA(i) to show IA($i+1$): By IA(i), $x_i \in \text{Reach}_x(H_2)$, so $x_i \in \mathcal{X}$. Consider the following cases from the definition of an HA:

- The next transition $(l_i, x_i) \xrightarrow{\beta_i} (l_{i+1}, x_{i+1})$ in (400) is a discrete transition. Then, it is present both in H_1 and H_2, because they have the same guard condition and transition function.
- The next transition $(l_i, x_i) \xrightarrow{\delta} (l_{i+1}, x_{i+1})$ in (400) is a continuous transition with duration $\delta \geq 0$. By the definition of continuous transitions, $l_i = l_{i+1}$ and there is a witness $\xi(t)$ defined on $0 \leq t \leq \delta$ such that $\xi(t)$ is continuous on $[0; \delta]$ and differentiable on $(0; \delta)$,

$$\xi(0) = x_i, \tag{402a}$$
$$\xi(\delta) = x_{i+1}, \tag{402b}$$
$$\dot{\xi}(t) \in \mathcal{F}_1(\xi(t)) \quad \forall t \in (0; \delta) \quad \text{and} \tag{402c}$$
$$\mathit{inv}(\xi(t)) = \mathit{true} \qquad \forall t \in (0; \delta). \tag{402d}$$

  To show that $\xi(t)$ is a witness for the same transition in H_2, it must be shown that also

$$\dot{\xi}(t) \in \mathcal{F}_2(\xi(t)) \quad \forall t \in (0; \delta). \tag{403}$$

  - If $\delta = 0$, the transition is the trivial transition $(l_i, x_i) \xrightarrow{0} (l_i, x_i)$, which is also present in H_2.
  - If $\delta > 0$ and $\xi(t) \in \mathcal{X} \oplus \mathbb{B}_\epsilon$ for all $t \in [0; \delta]$, then also $\dot{\xi}(t) \in \mathcal{F}_1(\xi(t)) \overset{(166a)}{\subseteq} \mathcal{F}_2(\xi(t))$ for all $t \in (0; \delta)$. Consequently, $\xi(t)$ is a witness of a continuous transition $(l_i, x_i) \xrightarrow{\delta} (l_i, x_{i+1})$ of H_2.

- The remaining case is that $\delta > 0$ and $\xi(t) \notin \mathcal{X} \oplus \mathbb{B}_\epsilon$ for some $t \in [0; \delta]$. Informally, ξ has left $\mathcal{X}$ by a distance of more than ϵ, as shown in Figure 46. Then, there is a time of violation t_1 with $0 \leq t_1 \leq \delta$ such that $\xi(t_1) \notin \mathcal{X} \oplus \mathbb{B}_\epsilon$. By the induction assumption IA(i), $x_i \in \text{Reach}_x(H_2)$, so $\xi(0) \in \text{Reach}_x(H_2)$. With

$$\text{Reach}_x(H_2) \overset{(166b)}{\subseteq} \mathcal{X} \subset \mathcal{X} \oplus \mathbb{B}_\epsilon \tag{404}$$

  it follows that $\xi(0) \in \mathcal{X} \oplus \mathbb{B}_\epsilon$, so $t_1 > 0$.

  In summary, $\xi(t)$ is continuous, starts at $\xi(0) \in \mathcal{X}$ and reaches $\xi(t_1) \notin \mathcal{X} \oplus \mathbb{B}_\epsilon$ at $t_1 > 0$. Consequently, as illustrated in Figure 46, there is a time t_0 between 0 and t_1 such that ξ has already left $\mathcal{X}$ but only by a distance of at most ϵ: There exists t_0 with $0 < t_0 < t_1$, $\xi(t_0) \notin \mathcal{X}$ and $\xi(t) \in \mathcal{X} \oplus \mathbb{B}_\epsilon \; \forall t \in [0; t_0]$.

  Consider the same $\xi(t)$ defined on a shortened time span $0 \leq t \leq t_0$. As it fulfills all conditions given in the definition of an HA, it is a witness of the shorter continuous transition $(l_i, x_i) \xrightarrow{t_0} (l_i, \xi(t_0))$ of H_1. Within that shorter time span, $\xi(t)$ stays in $\mathcal{X} \oplus \mathbb{B}_\epsilon$, so it fulfills all conditions of the previous case for $\delta = t_0$. Therefore, H_2 has the same transition $(l_i, x_i) \xrightarrow{t_0} (l_i, \xi(t_0))$.

  This leads to a contradiction: Because of this transition, $\xi(t_0) \in \text{Reach}_x(H_2)$. By the assumed condition (166b) of the theorem, $\text{Reach}_x(H_2) \subseteq \mathcal{X}$, so $\xi(t_0) \in \mathcal{X}$. However, in the current case, $\xi(t_0) \notin \mathcal{X}$, so the case is impossible.

Conclusion IA(i) holds for all $i \geq 0$ up to the length n of the given trajectory of H_1. Therefore, any trajectory of H_1 is a trajectory of H_2, so $H_1 \subseteq_x H_2$. □

Remark C.1. The ϵ-boundary in Theorem 5.2.1 is required, which is illustrated by the following example with "impossible" flow:

$$\mathcal{X}_0 = \{0\}, \quad \mathcal{F}_1(x) = \{1\}, \quad \mathcal{F}_2(x) = \begin{Bmatrix} \{1\}, & x \leq 1, \\ \{-1\}, & x > 1 \end{Bmatrix}, \quad \mathcal{X} = [-1; 1],$$
$$h(x) = \mathit{false}, \quad g(x) = x, \quad \mathit{inv}(x) = \mathit{true}. \tag{405}$$

H_1 is equivalent to $\dot{x} = 1, x(0) = 0$, which yields $x(t) = t$ for $t \geq 0$. H_2 has the solution $x(t) = t$ only within $0 \leq t \leq 1$. At $t = 1$, there is a deadlock, so the trajectories do not continue any further, as will be explained later. Consequently, $\text{Reach}_x(H_1) = \mathbb{R}_{\geq 0}$ and $\text{Reach}_x(H_2) = [0; 1] \subseteq \mathcal{X}$. However, for $\epsilon = 0$, Theorem 5.2.1 would state that $H_1 \subseteq_x H_2$, which is clearly false.

To see the deadlock of H_2 at $t = 1$, consider all possible transitions starting from $x(1) = 1$. First, there are no possible discrete transitions. Second, there is a continuous transition of duration $\delta = 0$, but it has no effect on x and t, so it can be ignored. Lastly, there is also no continuous transition of duration $\delta > 0$: This would require a witness $\xi(t)$ with $\xi(0) = x(1) = 1$ that is differentiable on $(0; \delta)$ and fulfills $\dot{\xi}(t) \in \mathcal{F}_2(\xi(t))$ on $(0; \delta)$. The derivative of a differentiable function is continuous, but $\dot{\xi} \in \mathcal{F}_2(\xi) \subseteq \{+1, -1\}$, so the only two candidates for $\dot{\xi}(t)$ are the constant functions $\dot{\xi}(t) = \pm 1$. Consequently, the only two candidate witnesses are $\xi(t) = 1 \pm t$. Consider an arbitrary time $t \in (0; \delta)$ to see that neither candidate fulfills $\dot{\xi}(t) \in \mathcal{F}_2(\xi(t))$: For the case "+", $\xi > 1$, so $\dot{\xi} = 1 \notin \mathcal{F}_2(\xi) = \{-1\}$. For the case "−", $\xi < 1$, so $\dot{\xi} = -1 \notin \mathcal{F}_2(\xi) = \{1\}$. Hence, there is no valid witness, so H_2 has no solutions beyond $t = 1$. ◁

Remark C.2. Theorem 5.2.1 makes no restrictions on the set $\mathcal{X}$. The set may be non-connected, non-convex or could even contain isolated points. Similarly, no special restrictions apply on the dynamics $F(x)$, such as the existence or continuity of solutions. This generality is a benefit of defining hybrid automata by a chain of transitions. ◁

Conjecture C.3. *Theorem 5.2.1 can be generalized to hybrid automata with multiple locations in a way such that there is a different assumption for each location.*

Proof idea. As a naive proof sketch, number all locations and treat the location number as an additional continuous state variable with zero derivative. Then, merge all locations into one whose flow, invariant, guard and transition depend on the location number. Similarly, merge the set of assumptions by introducing the location number as new dimension. Modify the dynamics appropriately to support the ϵ-bloating for the location number, e.g., by rounding the mode to the nearest integer. Then apply the original theorem. □

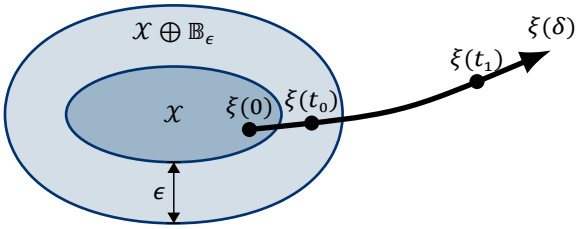


Figure 46: A continuous trajectory from inside $\mathcal{X}$ to outside $\mathcal{X} \oplus \mathbb{B}_\epsilon$ must first pass through the "boundary" $(\mathcal{X} \oplus \mathbb{B}_\epsilon) \setminus \mathcal{X}$.

C.2 Interval Bound for Continuous Evolution

Proof of Lemma 5.2.2. Without loss of generality, let $x_0 = 0$ and $t_0 = 0$. Assume $t \in [0; T]$, so

$$\dot{x} \in \mathcal{F}. \tag{406}$$

The interval hull is an outer bound for a set, i.e., $\square\mathcal{F} \supseteq \mathcal{F}$. Therefore,

$$\dot{x} \in \square\mathcal{F}. \tag{407}$$

Because any interval $\square\mathcal{F}$ can be represented by a conjunction of linear inequalities, this is a special case of the "piecewise-constant dynamics" defined in Frehse, 2015, p. 69. Note that the name "piecewise-constant" may be misleading here because $f(t)$ does *not* need to be piecewise constant. By Frehse, 2015, Lemma 1, (407) has a trajectory from $x(0)$ to $x(t_1)$, $t_1 \in [0; T]$, if and only if it also has a constant-derivative trajectory

$$\tilde{x}(t) = \underbrace{x(0)}_{0} + ct, \quad c \in \square\mathcal{F}, \quad 0 \le t \le t_1 \tag{408}$$

with the same start and end points $x(0) = \tilde{x}(0)$ and $\tilde{x}(t_1) = x(t_1)$. Therefore, as illustrated in Figure 47, the set of reachable $x(t)$ can be determined from

$$x(t) \in t\,\square\mathcal{F}. \tag{409}$$

Taking the union over all $t \in [0; T]$ yields

$$x(t) \in \bigcup_{t \in [0;T]} t\,\square\mathcal{F} = [0; T] \otimes \square\mathcal{F}. \tag{410}$$

□

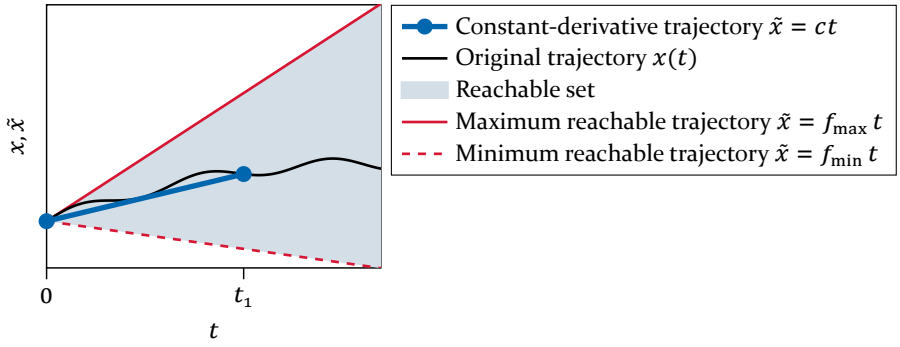


Figure 47: Illustration of Frehse, 2015, Lemma 1 for the one-dimensional system $\dot{x} \in [f_{\min}; f_{\max}]$, $x(0) = 0$. For every "original" trajectory from $x(0)$ to $x(t_1)$ there is a corresponding constant-derivative trajectory connecting the same points. As a result, the reachable set can be determined easily. Here, its bounds correspond to $\tilde{x} = f_{\min} t$ and $\tilde{x} = f_{\max} t$.

Alternate Proof Sketch for Lemma 5.2.2. Without loss of generality, let $x_0 = 0$ and $t_0 = 0$. Assume $t \in [0; T]$. Consider the solution as an integral, which can be represented as weighted sum of $n \to \infty$ summands:

$$x(t) = \int_0^t f(\tau)\,\mathrm{d}\tau = \lim_{n\to\infty} t \underbrace{\sum_{i=0}^{n-1} \frac{1}{n} f\left(\frac{it}{n}\right)}_{(*)}. \tag{411}$$

In this product, the first factor t is in $[0; T]$ by assumption. For finite n, the second factor $(*)$ is a convex combination of values of $f(\tau) \in \mathcal{F}$, which by definition is inside the convex hull of $\mathcal{F}$. Assume that by appropriate technical conditions, this argument also holds in the limit $n \to \infty$. Then,

$$x(t) \in t \operatorname{conv} \mathcal{F}. \tag{412}$$

This proves a stricter version of the theorem, using the convex hull instead of the integral hull. Because $\operatorname{conv} \mathcal{F} \subseteq \square \mathcal{F}$, this leads to

$$x(t) \in t \square \mathcal{F} \tag{413}$$

as in the previous proof. □

C.3 First-Order Continuization

This appendix presents the proof of first-order continuization. At first, a lemma is required to describe two differential equations whose states are related by $x = q(\tilde{x})$:

Lemma C.4 (Consistent Continuous Evolution)**.** *Consider two differential equations with state vectors x and $\tilde{x}$:*

$$\dot{x} = f(x, t), \tag{414}$$

$$\dot{\tilde{x}} = \tilde{f}(\tilde{x}, t). \tag{415}$$

Let these dynamics be consistent with regard to $x = q(\tilde{x})$, i.e.,

$$x(t_0^+) = q(\tilde{x}(t_0^+)) \quad \textit{and} \tag{416}$$

$$\underbrace{f(x,t)\big|_{x=q(\tilde{x}(t))}}_{\dot{x}} = \frac{\mathrm{d}q(\tilde{x}(t))}{\mathrm{d}t} \quad \textit{for all } t \in (t_0; t_1). \tag{417}$$

As a technical assumption, let f be sufficiently regular, e.g., Lipschitz continuous, to ensure that the initial value problem of (414) with the initial value $x(t_0^+)$ has a unique solution. (For detailed conditions, see Mattheij and Molenaar, 2002, Chap. 2.) Furthermore, let all employed functions and derivatives be defined for all $x, \tilde{x}$ and all $t \in [t_0; t_1]$.

Then, the solutions of x and $\tilde{x}$ are consistent with regard to the above transformation, i.e.,

$$x(t) = q(\tilde{x}(t)) \quad \textit{for all } t \in (t_0; t_1). \tag{418}$$

Proof. Assume that the conditions of the theorem are true. In the following, all equations hold for all $t \in (t_0; t_1)$. Denote

$$z(t) \coloneqq q(\tilde{x}(t)), \tag{419}$$

which leads to

$$\dot{z}(t) \overset{(419)}{=} \frac{\mathrm{d}q(\tilde{x}(t))}{\mathrm{d}t} \overset{(417)}{=} f(x,t)\big|_{x=q(\tilde{x}(t))} \overset{(419)}{=} f(z(t)), \tag{420}$$

and

$$z(t_0^+) \overset{(419),(416)}{=} x(t_0^+). \tag{421}$$

Therefore, z and x are defined by the same initial value problem, except for an irrelevant change of the variable name. The initial state is defined at $t = t_0^+$ by the value of (416). As assumed, the initial value problem for x has a unique solution. Because $z(t)$ is defined by the same initial value problem, it has the same unique solution, so $z(t) = x(t)$. □

Proof that $H_0 =_x H_1$ in Theorem 5.2.10. Consider the value of $\tilde{x}$ along a trajectory of H_1 to show that there is a one-to-one correspondence $x = q(\tilde{x})$ (as given in Figure 34 on page 135) with the state x along a trajectory of H_0. The following proof proceeds by induction, extending the trajectory by one transition in every induction step. For the sake of readability, the proof steps are not strictly formalized but sketched in terms of $x(t)$ and $\tilde{x}(t)$. To clarify the meaning of $x(t)$ at discrete transitions, $x(t^-)$ refers to the value before and $x(t^+)$ to the value after a possible discrete transition at t. This notation is possible because at most one discrete transition occurs at a given time instant. As there is no discrete transition at $t = 0$, both $t = 0^-$ and $t = 0^+$ refer to the initial state.

Simplifying Assumption: Well-Behaved Automaton Assume that in H_0 and H_1

1. every trajectory can be extended to infinite duration,
2. every continuous transition can be extended to duration T,
3. all functions are defined globally and
4. the continuous transitions have a unique solution, so that Lemma C.4 can be applied.

This excludes a number of problematic scenarios. It can be conjectured that most of these assumptions are without loss of generality: As sketched in the following, if there is a "problem", it is typically the same for H_0 and H_1, and except for that point in time, the proof can be reused:

1. Trajectories ending due to impossible flow (cf. H_2 in Remark C.1), finite escape time or a similar deadlock will admit the same effect in H_0 and H_1 at the same time.
2. Piecewise-continuous solutions due to discontinuous derivatives are built from piecing together multiple continuous transitions (cf. the definition in Section 4.2.2). This is the same in H_0 as in H_1. For each piece, the proof can be reused.

3. Trajectories of H_0 ending due to a transition with undefined f_d will render the state transformation $q(\tilde{x})$ of H_1 undefined from that time on, as it also contains f_d. However, this is not a problem since the derivative in H_1 will also become undefined, so both automata reach a deadlock.
4. As will be seen in the following proof, Lemma C.4 is only applied in specific cases such as $f(x,t) = 0$ or $q(\tilde{x}) = \tilde{x}$. This severely limits the possibility of pathological cases where the differential equation has different solutions in H_0 than in H_1.

Definitions H_0 and H_1 share the state names x_p and τ, though it must still be proven that these actually have equal value. For τ, it is obvious that both automata behave exactly the same. For x_p, however, this is not immediately clear, so its values will be distinguished by x_p^{H0} and x_p^{H1}.

In this notation, the definitions from Figure 34 become

$$x_\mathrm{d}^- \coloneqq \tilde{x}_\mathrm{d} + \delta_\mathrm{d}, \tag{422}$$
$$x_\mathrm{p}^- \coloneqq x_\mathrm{p}^{H1} + \delta_\mathrm{p} \tag{423}$$

and the states and transformation can be restated as

$$x \coloneqq \begin{bmatrix} x_\mathrm{p}^{H0} \\ x_\mathrm{d} \\ \tau \end{bmatrix}, \tag{424a}$$

$$\tilde{x} \coloneqq \begin{bmatrix} x_\mathrm{p}^{H1} \\ \tilde{x}_\mathrm{d} \\ \delta_\mathrm{p} \\ \delta_\mathrm{d} \\ \tau \end{bmatrix}, \tag{424b}$$

$$q(\tilde{x}) \coloneqq \begin{bmatrix} x_\mathrm{p}^{H1} \\ f_\mathrm{d}(\underbrace{x_\mathrm{p}^{H1} + \delta_\mathrm{p}}_{x_\mathrm{p}^-}, \underbrace{\tilde{x}_\mathrm{d} + \delta_\mathrm{d}}_{x_\mathrm{d}^-}) \\ \tau \end{bmatrix}. \tag{424c}$$

The dynamics of H_i are not restated here. Derivation steps referring to these dynamics are denoted by $\overset{H_i}{=}$.

Induction Assumption IA(k): For given $k \geq 0$, the trajectory is consistent until $t = kT^+$, i.e.,

$$x(t^+) = q(\tilde{x}(t^+)) \quad \text{for } 0 \leq t \leq kT, \tag{425a}$$
$$x(t^-) = q(\tilde{x}(t^-)) \quad \text{for } 0 \leq t \leq kT. \tag{425b}$$

Start of Induction ($k = 0$) At $t = 0$, the initialization is consistent, and no discrete transition occurs:

$$q(\tilde{x}(0^-)) \overset{H_1}{=} q(\tilde{x}(0^+)) \overset{H_1,(424b)}{=} q\left(\begin{bmatrix} x_{\mathrm{p},0} \\ x_{\mathrm{d},-1} \\ 0 \\ 0 \\ 0 \end{bmatrix}\right) \overset{(424c)}{=} \begin{bmatrix} x_{\mathrm{p},0} \\ f_{\mathrm{d}}(x_{\mathrm{p},0}, x_{\mathrm{d},-1}) \\ 0 \end{bmatrix}, \tag{426}$$

$$x(0^-) \overset{H_0}{=} x(0^+) \overset{H_0,(424a)}{=} \begin{bmatrix} x_{\mathrm{p},0} \\ f_{\mathrm{d}}(x_{\mathrm{p},0}, x_{\mathrm{d},-1}) \\ 0 \end{bmatrix} \overset{(426)}{=} q(\tilde{x}(0^-)) \overset{(426)}{=} q(\tilde{x}(0^+)). \tag{427}$$

Therefore, IA(0) holds.

Induction Step Assume IA(k), where $k \geq 0$.

1. *Continuous Evolution from $t = kT^+$ to $t = (k+1)T^-$:* The first goal is to show that the continuous evolution is consistent per

$$x(t) = q(\tilde{x}(t)) \quad \text{for} \quad kT < t < (k+1)T. \tag{428}$$

   This will be shown by Lemma C.4 for each component x_{p}, x_{d} and τ:

   - *Step 1: Initial Condition:* The "initial condition" at $t = kT^+$ is consistent per IA(k), which implies $x(kT^+) \overset{(425a)}{=} q(\tilde{x}(kT^+))$.
   - *Step 2: Flow:* The flow is consistent as well, i.e.,

$$f(x)\big|_{x=q(\tilde{x}(t))} = \frac{\mathrm{d}q(\tilde{x}(t))}{\mathrm{d}t} \quad \text{for} \quad kT < t < (k+1)T, \tag{429}$$

     where $f(x) \coloneqq \dot{x}$ denotes the flow of x in H_0.
   - *Step 3: Conclusion:* The evolution is consistent per (428).

a) *Consistency of τ:* As noted before, τ is obviously equal in both automata.

b) *Consistency of x_d:* The following steps show that for the evolution from $t = kT^+$ to $t = (k+1)T^-$, the state x_d in H_0 is consistent with its counterpart $f_\mathrm{d}(x_\mathrm{p}^{H1} + \delta_\mathrm{p}, \tilde{x}_\mathrm{d} + \delta_\mathrm{d})$ in H_1.

   i. *Initial Condition:* For the initial condition, IA(k) implies that

$$x_\mathrm{d}(kT^+) \overset{(425a),(424)}{=} f_\mathrm{d}(x_\mathrm{p}^{H1}(kT^+) + \delta_\mathrm{p}(kT^+), \tilde{x}_\mathrm{d}(kT^+) + \delta_\mathrm{d}(kT^+)\ ). \tag{430}$$

   ii. *Flow:* Next, the time derivatives of x_d and its equivalent in H_1 are shown to be equal. For the continuous evolution between $t = kT^+$ and $t = (k+1)T^-$, the following holds: In H_0,

$$\dot{x}_\mathrm{d} \overset{H_0}{=} 0, \tag{431}$$

and in H_1,

$$\left.\begin{aligned} \dot{x}_\mathrm{d}^- &\overset{(422)}{=} \dot{\tilde{x}}_\mathrm{d} + \dot{\delta}_\mathrm{d} \overset{H_1}{=} 0 \\ \dot{x}_\mathrm{p}^- &\overset{(423)}{=} \dot{x}_\mathrm{p}^{H1} + \dot{\delta}_\mathrm{p} \overset{H_1}{=} 0 \end{aligned}\right\} \Rightarrow \frac{\mathrm{d} f_\mathrm{d}(\overbrace{x_\mathrm{p}^{H1} + \delta_\mathrm{p}}^{\text{const}}, \overbrace{\tilde{x}_\mathrm{d} + \delta_\mathrm{d}}^{\text{const}})}{\mathrm{d}t} = 0. \tag{432}$$

Therefore, the x_d-component of the flow consistency (429) is true:

$$\dot{x}_\mathrm{d} \overset{(431),(432)}{=} \frac{\mathrm{d}\overbrace{f_\mathrm{d}(x_\mathrm{p}^{H1} + \delta_\mathrm{p}, \tilde{x}_\mathrm{d} + \delta_\mathrm{d})}^{x_\mathrm{d}\text{-component of } q(\tilde{x})}}{\mathrm{d}t} = 0 \quad \text{for} \quad kT < t < (k+1)T. \tag{433}$$

   iii. *Conclusion:* The initial conditions and flows are consistent, and x_d is constant within the period. Therefore,

$$\begin{aligned} x_\mathrm{d}(t) &\overset{\text{Lemma C.4, } (430),(433)}{=} f_\mathrm{d}(x_\mathrm{p}^{H1}(t) + \delta_\mathrm{p}(t), \tilde{x}_\mathrm{d}(t) + \delta_\mathrm{d}(t)) \\ &\overset{(431)}{=} x_\mathrm{d}(kT^+) \overset{(431)}{=} x_\mathrm{d}((k+1)T^-) \\ &\qquad \text{for} \quad kT < t < (k+1)T. \end{aligned} \tag{434}$$

This result will now be used to show the consistency of x_p.

c) *Consistency of x_p:* Analogous to the considerations for x_d, consider x_p from $t = kT^+$ to $t = (k+1)T^-$:

i. *Initial Condition:* By IA(k),

$$x_\mathrm{p}^{H0}(kT^+) \overset{(425a),(424)}{=} x_\mathrm{p}^{H1}(kT^+), \tag{435}$$

so the initial condition matches.

ii. *Flow:* In H_0,

$$\dot{x}_\mathrm{p}^{H0} \overset{H_0}{=} f_\mathrm{p}(x_\mathrm{p}^{H0}, x_\mathrm{d}). \tag{436}$$

In H_1,

$$\begin{aligned}\dot{x}_\mathrm{p}^{H1} &\overset{H_1}{=} f_\mathrm{p}(x_\mathrm{p}^{H1}, f_\mathrm{d}(x_\mathrm{p}^{H1} + \delta_\mathrm{p}, \tilde{x}_\mathrm{d} + \delta_\mathrm{d})) \\ &\overset{(434)}{=} f_\mathrm{p}(x_\mathrm{p}^{H1}, x_\mathrm{d}) \\ &\overset{(436)}{=} \dot{x}_\mathrm{p}^{H_0}\Big|_{x_\mathrm{p}^{H_0} = x_\mathrm{p}^{H_1}} \quad \text{for} \quad kT < t < (k+1)T. \end{aligned} \tag{437}$$

iii. *Conclusion:* Therefore, x_p evolves consistently:

$$x_\mathrm{p}^{H0} \overset{\text{Lemma C.4, } (435),(437)}{=} x_\mathrm{p}^{H1} \quad \text{for} \quad kT < t < (k+1)T. \tag{438}$$

From items 1a to 1c, it follows that

$$x(t) \overset{(434),(438),(424)}{=} q(\tilde{x}(t)) \quad \text{for} \quad kT < t < (k+1)T. \tag{439}$$

As discrete transitions occur only at $t = kT$, this can be rewritten as

$$x(t^-) \overset{H_0}{=} x(t^+) \overset{(439)}{=} q(\tilde{x}(t^-)) \overset{H_1}{=} q(\tilde{x}(t^+)) \quad \text{for} \quad kT < t < (k+1)T \tag{440}$$

and

$$x((k+1)T^-) \overset{(439),H_0,H_1}{=} q(\tilde{x}((k+1)T^-)). \tag{441}$$

Together with IA(k), this shows (425b) for $k+1$.

2. *Helper Variables x_p^- and x_d^- in H_1:*

a) *Preliminary Considerations:* At $t = 0$, there is no discrete transition, so

$$\delta_\mathrm{p}(0^+) \overset{H_1}{=} \delta_\mathrm{p}(0^-) \overset{H_1}{=} 0 \quad \text{and} \tag{442}$$

$$\delta_\mathrm{d}(0^+) \overset{H_1}{=} \delta_\mathrm{d}(0^-) \overset{H_1}{=} 0. \tag{443}$$

Due to the discrete transition at $t = kT$, $k \geq 1$,

$$\delta_\mathrm{p}(kT^+) \overset{H_1}{=} 0 \tag{444}$$

$$\delta_\mathrm{d}(kT^+) \overset{H_1}{=} 0. \tag{445}$$

Because $\tilde{x}_\mathrm{d}$ and x_p^{H1} are unaffected by discrete transitions,

$$\tilde{x}_\mathrm{d}(t^+) \overset{H_1}{=} \tilde{x}_\mathrm{d}(t^-) \quad \text{and} \tag{446}$$

$$x_\mathrm{p}^{H1}(t^+) \overset{H_1}{=} x_\mathrm{p}^{H1}(t^-) \tag{447}$$

hold for all t. Therefore, for any $k \geq 0$,

$$\begin{aligned} x_\mathrm{p}^-(kT^+) &\overset{(423)}{=} x_\mathrm{p}^{H1}(kT^+) + \delta_\mathrm{p}(kT^+) \overset{(442),(444)}{=} x_\mathrm{p}^{H1}(kT^+) \\ &\overset{(447)}{=} x_\mathrm{p}^{H1}(kT^-) \end{aligned} \tag{448}$$

and

$$\begin{aligned} x_\mathrm{d}^-(kT^+) &\overset{(422)}{=} \tilde{x}_\mathrm{d}(kT^+) + \delta_\mathrm{d}(kT^+) \overset{(443),(445)}{=} \tilde{x}_\mathrm{d}(kT^+) \\ &\overset{(446)}{=} \tilde{x}_\mathrm{d}(kT^-). \end{aligned} \tag{449}$$

b) *Interpretation of x_p^-:* The variable $x_\mathrm{p}^- \overset{(423)}{:=} x_\mathrm{p}^{H1} + \delta_\mathrm{p}$ remains constant at the current sample-and-hold value of x_p: For $kT < t < (k+1)T$,

$$\dot{x}_\mathrm{p}^- \overset{(423)}{=} \dot{x}_\mathrm{p}^{H1} + \dot{\delta}_\mathrm{p} \overset{H_1}{=} 0 \tag{450}$$

$$\Rightarrow x_\mathrm{p}^-(t) \overset{(450)}{=} x_\mathrm{p}^-(kT^+) \overset{(423)}{=} x_\mathrm{p}^{H1}(kT^+) + \delta_\mathrm{p}(kT^+) \tag{451}$$

$$\overset{(448)}{=} x_\mathrm{p}^{H1}(kT^-) + 0 \overset{\mathrm{IA}(k)}{=} x_\mathrm{p}^{H0}(kT^-). \tag{452}$$

c) *Interpretation of x_d^-:* Similarly, $x_\mathrm{d}^- \overset{(422)}{\coloneqq} \tilde{x}_\mathrm{d} + \delta_\mathrm{d}$ is a sample-and-hold value of $\tilde{x}_\mathrm{d}$: For $kT < t < (k+1)T$,

$$\dot{x}_\mathrm{d}^- \overset{(422)}{=} \dot{\tilde{x}}_\mathrm{d} + \dot{\delta}_\mathrm{d} \overset{H_1}{=} 0 \tag{453}$$

$$\Rightarrow x_\mathrm{d}^-(t) \overset{(453)}{=} x_\mathrm{d}^-(kT^+) \overset{(449)}{=} \tilde{x}_\mathrm{d}(kT^+). \tag{454}$$

d) *Interpolator state $\tilde{x}_\mathrm{d}$:* The state $\tilde{x}_\mathrm{d}$ will now be shown to be a specific linear interpolation of x_d: Within the period $kT < t < (k+1)T$,

$$\dot{\tilde{x}}_\mathrm{d} \overset{H_1}{=} \frac{1}{T}\Big(\underbrace{f_\mathrm{d}(x_\mathrm{p}^{H1} + \delta_\mathrm{p}, \tilde{x}_\mathrm{d} + \delta_\mathrm{d})}_{\overset{(434)}{=} x_\mathrm{d}(kT^+)} - \underbrace{(\tilde{x}_\mathrm{d} + \delta_\mathrm{d})}_{\overset{(454)}{=} \tilde{x}_\mathrm{d}(kT^+)} \Big) = \text{const.} \tag{455}$$

holds. Integrating this result over the period leads to

$$\begin{aligned}
\tilde{x}_\mathrm{d}((k+1)T^-) &= \tilde{x}_\mathrm{d}(kT^+) + \int_{kT^+}^{(k+1)T^-} \dot{x}_\mathrm{d}(\tau)\,\mathrm{d}\tau && (456)\\
&\overset{(455)}{=} \tilde{x}_\mathrm{d}(kT^+) + T \cdot \frac{1}{T}\left(x_\mathrm{d}(kT^+) - \tilde{x}_\mathrm{d}(kT^+)\right) && (457)\\
&= x_\mathrm{d}(kT^+) && (458)\\
&\overset{(434)}{=} x_\mathrm{d}((k+1)T^-). && (459)
\end{aligned}$$

In summary, $\tilde{x}_\mathrm{d}$

- is continuous (446),
- matches x_d at the *end* $t = (k+1)T^-$ of the period (459), just before the next controller execution, and
- has constant derivative within the period (455).

If, as shown later, the above holds for all k and not just for the current value of k, then the above defines a linear interpolation with supporting points at $t = (k+1)T^-$.

e) *Summary:* If IA(k) is later shown to hold for all k, then the trajectory of H_1 matches the illustration in Figure 34 (page 135, top right):

- $x_\mathrm{p}^- \overset{(423)}{\coloneqq} x_\mathrm{p}^{H1} + \delta_\mathrm{p}$ is the newest sample-and-hold value of $x_\mathrm{p}^{H0} = x_\mathrm{p}^{H1}$.

- $x_{\mathrm{d}}^{-} \overset{(422)}{\coloneqq} \tilde{x}_{\mathrm{d}} + \delta_{\mathrm{d}}$ is the *previous* discrete-time value of x_{d}.
- Consequently, $f_{\mathrm{d}}(x_{\mathrm{p}}^{-}, x_{\mathrm{d}}^{-})$ reconstructs the *current* value of x_{d}.
- $\tilde{x}_{\mathrm{d}}$ is linear interpolation of x_{d} with supporting points at $t = (k+1)T^{-}$, i.e., with a delay of (almost) one period.

3. *Discrete Transition at $t = (k+1)T$:* Next, it is shown that the transition of H_0 and H_1 at $t = (k+1)T$ is consistent with $x = q(\tilde{x})$. In H_0, this transition is

$$x_{\mathrm{p}}^{H0}((k+1)T^{+}) \overset{H_0}{=} x_{\mathrm{p}}^{H0}((k+1)T^{-}), \tag{460a}$$

$$x_{\mathrm{d}}((k+1)T^{+}) \overset{H_0}{=} f_{\mathrm{d}}(x_{\mathrm{p}}^{H0}((k+1)T^{-}),\, x_{\mathrm{d}}((k+1)T^{-})), \tag{460b}$$

$$\tau((k+1)T^{+}) \overset{H_0}{=} 0. \tag{460c}$$

In H_1, the corresponding transition is

$$x_{\mathrm{p}}^{H1}((k+1)T^{+}) \overset{H_1}{=} x_{\mathrm{p}}^{H1}((k+1)T^{-}), \tag{461a}$$

$$\tilde{x}_{\mathrm{d}}((k+1)T^{+}) \overset{H_1}{=} \tilde{x}_{\mathrm{d}}((k+1)T^{-}) \tag{461b}$$

$$\delta_{\mathrm{d}}((k+1)T^{+}) \overset{H_1}{=} 0, \tag{461c}$$

$$\delta_{\mathrm{p}}((k+1)T^{+}) \overset{H_1}{=} 0, \tag{461d}$$

$$\tau((k+1)T^{+}) \overset{H_1}{=} 0. \tag{461e}$$

Therefore,

$$q(\tilde{x}((k+1)T^{+}))$$

$$\overset{(424)}{=} \begin{bmatrix} x_{\mathrm{p}}^{H1}((k+1)T^{+}) \\ f_{\mathrm{d}}(x_{\mathrm{p}}^{H1}((k+1)T^{+}) + \delta_{\mathrm{p}}((k+1)T^{+}),\, \tilde{x}_{\mathrm{d}}((k+1)T^{+}) + \delta_{\mathrm{d}}((k+1)T^{+})) \\ \tau((k+1)T^{+}) \end{bmatrix} \tag{462}$$

$$\overset{(461)}{=} \begin{bmatrix} x_{\mathrm{p}}^{H1}((k+1)T^{-}) \\ f_{\mathrm{d}}(x_{\mathrm{p}}^{H1}((k+1)T^{-}) + 0,\, \tilde{x}_{\mathrm{d}}((k+1)T^{-}) + 0) \\ 0 \end{bmatrix} \tag{463}$$

$$\overset{(441)}{=} \begin{bmatrix} x_{\mathrm{p}}^{H0}((k+1)T^{-}) \\ f_{\mathrm{d}}(x_{\mathrm{p}}^{H0}((k+1)T^{-}),\, \tilde{x}_{\mathrm{d}}((k+1)T^{-})) \\ 0 \end{bmatrix} \tag{464}$$

$$\overset{(459)}{=} \begin{bmatrix} x_{\mathrm{p}}^{H0}((k+1)T^-) \\ f_{\mathrm{d}}(x_{\mathrm{p}}^{H0}((k+1)T^-),\ x_{\mathrm{d}}((k+1)T^-)) \\ 0 \end{bmatrix} \tag{465}$$

$$\overset{(460)}{=} \begin{bmatrix} x_{\mathrm{p}}^{H0}((k+1)T^+) \\ x_{\mathrm{d}}((k+1)T^+) \\ \tau((k+1)T^+) \end{bmatrix} \overset{(424a)}{=} x((k+1)T^+). \tag{466}$$

Hence, the trajectory is consistent at $t = (k+1)T^+$.

4. *Conclusion:* IA$(k+1)$ consists of two parts, the left-side consistency (425b) and the right-side consistency (425a). Both are proven by the previous equations: IA(k), (440) and (441) show the left-side consistency (425b). IA(k), (440) and (466) show the right-side consistency (425a). Therefore, IA$(k+1)$ is true.

Conclusion of Induction IA(k) holds for all $k \geq 0$. Therefore, $H_0 =_x H_1$ with $x = q(\tilde{x})$. □

D Abstractions

This appendix contains proofs detached from Chapter 6.

Proof of Theorem 6.4.3: EISS ⇒ DGES. Setting $d \equiv 0$ in the definition of EISS(ρ, α, β) (Definition 6.4.1) leads to the definition of DGES(ρ, α) (Definition 4.1.12). This also holds for nonlinear systems. □

Proof of Theorem 6.4.3: DGES ⇒ EISS. Unrolling the recursion (226) yields

$$x_k = A^k x_0 + \sum_{i=0}^{k-1} A^i G d_{k-1-i} \tag{467}$$

$$\Rightarrow |x_k| \leq |A^k x_0| + \sum_{i=0}^{k-1} |A^i G d_{k-1-i}|. \tag{468}$$

It should be noted that this step explicitly requires the superposition property and is not directly applicable to nonlinear systems. For linear systems, the exponential stability (215) can be concretized to

$$\forall x \in \mathbb{R}^n, i \in \mathbb{N}_0 : \quad |A^i x| \leq \alpha \rho^i |x|. \tag{469}$$

Applying this to each summand of (468) yields

$$|x_k| \leq \alpha\rho^k|x_0| + \sum_{i=0}^{k-1} \alpha\rho^i|Gd_{k-1-i}| \tag{470}$$

$$\leq \alpha\rho^k|x_0| + \alpha\|G\|_2 \sum_{i=0}^{k-1} \rho^i|d_{k-1-i}|. \tag{471}$$

This matches the definition of $\mathrm{EISS}(\rho, \alpha, \alpha\|G\|_2)$. □

Proof for Theorem 6.6.2. Assume σ obeys (M, K)-weak execution and $0 < \rho_0 \leq \rho_1$. First, an upper bound on the number of skips in k periods is constructed by partitioning the sequence $i = 0, 1, \ldots, k-1$ into $\lfloor k/K \rfloor$ chunks of length K (first: $0 \leq i \leq K-1$, second: $K \leq i \leq 2K-1$, and so on) and a remainder ($k - (k \bmod K) \leq i \leq k-1$) of length $k \bmod K$.

$$\Rightarrow\ \forall k \geq 0: \quad \sum_{i=0}^{k-1} \sigma_i = \left(\sum_{j=1}^{\lfloor k/K \rfloor} \underbrace{\sum_{i=(j-1)K}^{jK-1} \sigma_i}_{\leq \overline{m} \text{ due to (267b)}} \right) + \underbrace{\sum_{i=k-(k \bmod K)}^{k-1} \sigma_i}_{\substack{\leq k \bmod K \text{ because } \sigma_i \leq 1, \\ \leq \overline{m} \text{ due to (267b)}}}$$

$$\leq \underbrace{\overline{m}\lfloor k/K \rfloor}_{\overline{m} \text{ per integer multiple of } K} + \underbrace{\min(\overline{m}, k \bmod K)}_{\substack{\text{remaining} < K \text{ periods:} \\ \text{at most } \overline{m}}}. \tag{472}$$

This bound is tight as it is reached for $\sigma_k = \begin{cases} 1, & k \bmod K \leq \overline{m}, \\ 0, & \text{else.} \end{cases}$

Evaluate κ_k from Theorem 6.6.1:

$$\kappa_k = \prod_{i=0}^{k-1} \rho_{\sigma_i} \overset{(267a)}{=} \prod_{i=0}^{k-1} \begin{cases} \rho_0, & \sigma_i = 0 \\ \rho_1, & \sigma_i = 1. \end{cases} \tag{473}$$

Using $\sigma_i \in \{0, 1\}$ and the (M, K)-constraint (267b), an upper bound $\bar{\kappa}_k$ can be derived:

$$\kappa_k \overset{(473)}{=} \rho_0^{\overbrace{k-\sum_{i=0}^{k-1}\sigma_i}^{\sum_{i=0}^{k-1}(1-\sigma_i)}} \rho_1^{\sum_{i=0}^{k-1}\sigma_i} = \rho_0^k \Big(\underbrace{\rho_0^{-1}\rho_1}_{\text{assumed} \geq 1}\Big)^{\overbrace{\sum_{i=0}^{k-1}\sigma_i}^{\text{upper bounded by (472)}}} \tag{474}$$

$$\overset{(472)}{\leq} \rho_0^k \left(\rho_0^{-1}\rho_1\right)^{\overline{m}\lfloor k/K \rfloor + \min(\overline{m}, k \bmod K)} =: \bar{\kappa}_k. \tag{475}$$

For $k \to \infty$, the maximum average ratio of skips ($\sigma_k = 1$) is $\overline{m}/K$:

$$\lim_{k\to\infty} \frac{1}{k} \sum_{i=0}^{k-1} \sigma_i \overset{(472)}{\leq} \lim_{k\to\infty} \frac{\overline{m}\lfloor k/K \rfloor + \min(\overline{m}, k \bmod K)}{k} \tag{476}$$

$$= \lim_{k\to\infty} \left(\overline{m} \underbrace{\frac{\lfloor k/K \rfloor}{k}}_{\to \frac{1}{K}} + \underbrace{\frac{\min(\overline{m}, k \bmod K)}{k}}_{\to 0} \right) = \frac{\overline{m}}{K}. \tag{477}$$

The following derivation uses the above statements to determine the minimum $\tilde{\rho}$ and corresponding $\tilde{\alpha}$ that fulfill the ansatz

$$\bar{\kappa}_k \leq \tilde{\alpha}\tilde{\rho}^k \quad \forall k \geq 0, \tag{478}$$

which shall later be used to show the abstracted stability criterion (266). Assuming $\tilde{\rho} > 0$, the ansatz is equivalent to

$$\bar{\kappa}_k^{1/k} \leq \underbrace{\tilde{\alpha}^{1/k}}_{\to 1 \text{ for } k\to\infty} \tilde{\rho} \quad \forall k \geq 0, \tag{479}$$

which motivates that a candidate for the minimum $\tilde{\rho}$ is

$$\tilde{\rho} \coloneqq \lim_{k\to\infty} \bar{\kappa}_k^{1/k} \overset{(475)}{=} \lim_{k\to\infty} \rho_0 \left(\rho_0^{-1}\rho_1\right)^{\frac{\overline{m}\lfloor k/K \rfloor + \min(\overline{m}, k \bmod K)}{k}} \tag{480}$$

$$\overset{(477)}{=} \rho_0^{1-\frac{\overline{m}}{K}} \rho_1^{\frac{\overline{m}}{K}} \overset{(267b)}{=} \rho_0^{\frac{M}{K}} \rho_1^{\frac{K-M}{K}}. \tag{481}$$

The result can be interpreted as a special "weighted average" of the stability exponent for the extreme cases: Never skipping ($M = K$) yields the nominal case $\tilde{\rho} = \rho_0$ and always skipping ($M = 0$) results in $\tilde{\rho} = \rho_1$.

The validity of this candidate $\tilde{\rho}$ is then implicitly proven by determining the corresponding overshoot factor $\tilde{\alpha}$ and showing that $\tilde{\alpha} < \infty$: For $\tilde{\alpha}$, consider

$$\bar{\kappa}_k \overset{!}{\leq} \tilde{\alpha}\tilde{\rho}^k \quad \forall k \geq 0 \quad \Leftrightarrow \quad \tilde{\alpha} \geq \bar{\kappa}_k \tilde{\rho}^{-k} \quad \forall k \geq 0 \quad \Leftrightarrow \quad \tilde{\alpha} \geq \max_{k \geq 0} \bar{\kappa}_k \tilde{\rho}^{-k} \tag{482}$$

and choose $\tilde{\alpha}$ as a finite upper bound for the right-hand side:

$$\max_{k\geq 0} \tilde{\rho}^{-k}\bar{\kappa}_k \overset{(475),\,(481)}{=} \max_{k\geq 0} \rho_0^{-\frac{K-\overline{m}}{K}k} \rho_1^{-\frac{\overline{m}}{K}k} \rho_0^k \Big(\underbrace{\rho_0^{-1}\rho_1}_{\text{assumed} \geq 1}\Big)^{\overline{m}\overbrace{\lfloor k/K \rfloor}^{\leq k/K} + \overbrace{\min(\overline{m}, k \bmod K)}^{\leq \overline{m}}} \tag{483}$$

$$\leq \max_{k\geq 0} \rho_0^{-\frac{K-\overline{m}}{K}k} \rho_1^{-\frac{\overline{m}}{K}k} \rho_0^k \left(\rho_0^{-1}\rho_1\right)^{\overline{m}k/K+\overline{m}} \tag{484}$$

$$= \max_{k\geq 0} \rho_0^{\overbrace{k\left(-\frac{K-\overline{m}}{K}+1-\frac{\overline{m}}{K}\right)}^{0}-\overline{m}} \rho_1^{\overbrace{k\left(-\frac{\overline{m}}{K}+\frac{\overline{m}}{K}\right)}^{0}+\overline{m}} \tag{485}$$

$$= \rho_0^{-\overline{m}}\rho_1^{\overline{m}} = \left(\frac{\rho_1}{\rho_0}\right)^{\overline{m}} \overset{(267b)}{=} \left(\frac{\rho_1}{\rho_0}\right)^{K-M} \overset{(268)}{=:} \tilde{\alpha}. \tag{486}$$

Note that $\tilde{\alpha} = (\tilde{\rho}/\rho_0)^K$.

With $\tilde{\alpha}$ and $\tilde{\rho}$ as per Equations (481) and (486), one can prove the ansatz

$$\forall k \geq 0 : \quad \kappa_k \overset{(475)}{\leq} \bar{\kappa}_k \overset{(482),\,(486)}{\leq} \tilde{\alpha}\tilde{\rho}^k, \tag{487}$$

which leads to DGES($\tilde{\rho}, \tilde{\alpha}\alpha$) by Theorem 6.6.1. □

E Dynamic Resource Management

This appendix discusses implementation details of the abstraction-based dynamic resource management of Chapter 7. A previous version of this appendix appeared in an internal technical report (Gaukler, 2020b).

E.1 Bounds on The Worst-Case Disturbance Effect

To investigate the pessimism incurred by abstractions, the actual worst-case effect of disturbance and initial state on $|s_k|$ must be considered, that is,

$$\max_{\substack{\tilde{x}_0, d_0, d_1, \ldots, d_{k-1} \\ |\tilde{x}_0| \leq |x_0|_{\max}, |d_i| \leq |d|_{\max}}} |s_k|. \tag{488}$$

Unrolling the discrete-time dynamics (213) dynamics leads to

$$x_1 = A(\sigma_0)x_0 + G(\sigma_0)d_0, \tag{489}$$
$$x_2 = A(\sigma_1)A(\sigma_0)x_0 + A(\sigma_1)G(\sigma_0)d_0 + G(\sigma_1)d_1, \tag{490}$$
$$\vdots$$
$$x_k = \left(\widetilde{\prod_{j=0}^{k-1}} A(\sigma_j)\right)x_0 + \sum_{i=0}^{k-1}\left(\left(\widetilde{\prod_{j=i+1}^{k-1}} A(\sigma_j)\right)G(\sigma_i)d_i\right), \tag{491}$$

where $\widetilde{\prod}$ denotes the reversed product. Consequently,

$$s_k = \bar{C}_\mathrm{s} x_k \overset{(491),(213c)}{=} \underbrace{\bar{C}_\mathrm{s}\left(\widetilde{\prod_{j=0}^{k-1}} A(\sigma_j)\right)C_0}_{M_\mathrm{x}}\,\tilde{x}_0 + \sum_{i=0}^{k-1}\underbrace{\bar{C}_\mathrm{s}\left(\widetilde{\prod_{j=i+1}^{k-1}} A(\sigma_j)\right)G(\sigma_i)}_{M_{\mathrm{d},i}}\,d_i. \tag{492}$$

In the following, k is considered fixed and therefore not denoted in the matrix subscripts.

E.1.1 Difficulty

Determining the exact maximum of $|s_k|$ is nontrivial. The problem can be interpreted as a question of discrete-time reachability analysis. Geometrically, the set of reachable s_k becomes more complex with increasing k, so a precise analysis is outside of the scope of this thesis.

In the following, the question is interpreted as an optimization problem and an induced matrix norm. Nevertheless, the problem remains difficult. For illustration, assume $\tilde{x}_0 = 0$, $|d|_{\max} = 1$ and consider the remaining problem

$$c = \max_{|d_0| \leq 1, \ldots, |d_{k-1}| \leq 1} \left|\sum_{i=0}^{k} M_{\mathrm{d},i} d_i\right|. \tag{493}$$

For the single-input single-output case $n_\mathrm{s} = n_\mathrm{dist} = 1$, $M_{\mathrm{d},i}$ is scalar and per Balakrishnan and Boyd, 1992 the solution is $c = \sum |M_{\mathrm{d},i}|$. For the multivariable case considered here, the problem is more difficult: Denote

$$\bar{d} = \begin{bmatrix} d_0 \\ \vdots \\ d_{k-1} \end{bmatrix} \in \mathbb{R}^{n_\mathrm{dist} k}, \quad \bar{M} = \begin{bmatrix} M_{d,0} & \dots & M_{d,k-1} \end{bmatrix} \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{dist} k}. \tag{494}$$

The problem represents the matrix norm induced by a highly unusual vector norm $|\bar{d}|_*$:

$$c = \max_{|\bar{d}|_* \leq 1} |\bar{M}\bar{d}| \tag{495}$$

$$\text{with} \quad |\bar{d}|_* \coloneqq \max_{i \in \{0,\dots,k-1\}} \sqrt{\sum_{j=i \cdot n_\mathrm{dist}+1}^{(i+1)n_\mathrm{dist}} (\bar{d}^{(j)})^2} = \max_{i \in \{0,\dots,k-1\}} |d_i|. \tag{496}$$

Unlike for many other induced matrix norms, no exact result could be found in the literature for this problem. Therefore, upper and lower bounds are considered instead. First, two simple bounds based on well-known induced matrix norms of $\bar{M}$ are shown for comparison, before tighter but more complicated bounds are derived.

Simple Upper Bound The maximum norm $|d_i|_\infty \coloneqq \max_j |d_i^{(j)}|$ is a lower bound for the euclidean norm, i.e., $|d_i|_\infty \leq |d_i|$. Therefore, changing $|d_i| \leq 1$ to $|d_i|_\infty \leq 1$ yields an upper bound for the original problem c:

$$c \leq \max_{|d_0|_\infty \leq 1, \dots, |d_{k-1}|_\infty \leq 1} \left| \sum_{i=0}^{k-1} M_{\mathrm{d},i} d_i \right|. \tag{497}$$

Also, this new norm allows for a simplification since $|d_0|_\infty \leq 1 \wedge \dots \wedge |d_{k-1}|_\infty \leq 1$ is equivalent to $|\bar{d}|_\infty \leq 1$. Then, standard results for matrix and vector norms lead to an upper bound:

$$c \leq \max_{|\bar{d}|_\infty \leq 1} |\bar{M}\bar{d}| \overset{\text{Bernstein, 2009, Fact 9.7.29}}{\leq} \max_{|\bar{d}|_\infty \leq 1} \sqrt{n_\mathrm{s}}\, |\bar{M}\bar{d}|_\infty \tag{498}$$

$$\overset{\text{Bernstein, 2009, eq. (9.4.21)}}{=} \sqrt{n_\mathrm{s}} \max_{i \in \{1,\dots,n_\mathrm{s}\}} \sum_{j=1}^{n_\mathrm{dist} k} |\bar{M}_{(i,j)}|. \tag{499}$$

Simple Lower Bound For a second variant, the bound instead considers the sum $|d_0|^2 + \cdots + |d_{k-1}|^2 = |\bar{d}|^2$: The condition $\sum |d_i|^2 \leq 1$ is stronger than $|d_i| \leq 1$, which leads to a lower bound for c:

$$c \geq \max_{|d_0|^2+\cdots+|d_{k-1}|^2 \leq 1} \left| \sum_{i=0}^{k-1} M_{\mathrm{d},i} d_i \right| = \max_{|\bar{d}| \leq 1} |\bar{M}\bar{d}| = \|\bar{M}\|_2. \tag{500}$$

E.1.2 Upper Bound

Consider $\tilde{x}_0 \neq 0$ and an arbitrary $|d|_{\max}$. An upper bound on s_k is obtained from (492) and the properties of the spectral norm $\|\cdot\|_2$ as

$$|s_k| \leq \|M_{\mathrm{x}}\|_2 \, |\tilde{x}_0|_{\max} + \sum_{i=0}^{k-1} \left\|M_{\mathrm{d},i}\right\|_2 \, |d|_{\max}. \tag{501}$$

Note that this bound is not feasible for run-time implementation because the computation and memory requirement grows unbounded with increasing time k.

To illustrate the improvement by this new bound, let $\tilde{x}_0 = 0, |d|_{\max} = 1, n_{\mathrm{dist}} = n_{\mathrm{s}} > 1$ and $M_{\mathrm{d},i} = I$. Then, the new bound (501) is $|s_k| \leq k$, which is tighter than the simple bound (500) of $c \leq \sqrt{n_{\mathrm{s}}}\, k$.

E.1.3 Lower Bound

In principle, the randomized simulation used in the experiments yields a lower bound on the worst case $|s_k|_{\mathrm{worst}}$. However, the probability of reaching a result close to the worst case is typically low. To derive an alternative lower bound that is independent of random chance, the initial state is optimistically neglected and then a lower bound for the resulting sum is applied:

$$|s_k|_{\mathrm{worst}} = \max_{|d_i| \leq d_{\max}, |\tilde{x}_0| \leq |\tilde{x}_0|_{\max}} |s_k| \tag{502}$$

$$\geq \max_{|d_i| \leq |d|_{\max}, \tilde{x}_0 = 0} \left| \sum_{i=0}^{k-1} M_{\mathrm{d},i} d_i \right| \overset{(*)}{\geq} |d|_{\max} \sqrt{\sum_{i=0}^{k-1} \|M_{\mathrm{d},i}\|_2^2}, \tag{503}$$

where $(*)$ is due to Theorem E.5, which is derived in the following sections.

It can be shown that this result is at least as good as the simple lower bound (500): Using the fact that $|v| = \|v\|_2 = \|v^\top\|_2$ holds for any vector v, the result can be written as

$$\sqrt{\sum_{i=0}^{k-1} \|M_{\mathrm{d},i}\|_2^2} = \left|\begin{bmatrix}\|M_{d,0}\|_2 & \dots & \|M_{d,k-1}\|_2\end{bmatrix}^\top\right| \tag{504}$$

$$= \left\|\begin{bmatrix}\|M_{d,0}\|_2 & \dots & \|M_{d,k-1}\|_2\end{bmatrix}\right\|_2 \tag{505}$$

$$\overset{\text{Bernstein, 2009, Fact 9.10.1.v}}{\geq} \left\|\begin{bmatrix}M_{d,0} & \dots & M_{d,k-1}\end{bmatrix}\right\|_2 = \|\bar{M}\|_2 \tag{506}$$

The following example motivates that the new bound is tighter: Let

$$\tilde{x}_0 = 0, k = 2, n_\mathrm{s} = n_\mathrm{dist} = 2, M_{d,0} = \begin{bmatrix}1 & \\ & 0\end{bmatrix}, M_{d,1} = \begin{bmatrix}0 & \\ & 1\end{bmatrix} \tag{507}$$

and $|d|_\mathrm{max} = 1$, for which the worst case can be derived explicitly as

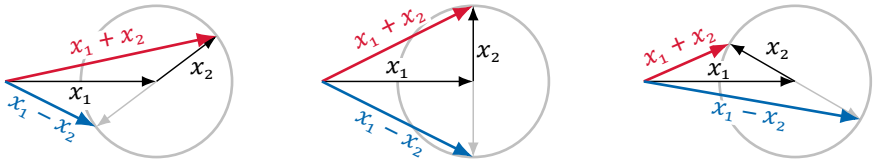
$$d_0 = \begin{bmatrix}1\\0\end{bmatrix}, \quad d_1 = \begin{bmatrix}0\\1\end{bmatrix}, \quad s_1 \overset{(492)}{=} \begin{bmatrix}d_0^{(1)}\\d_1^{(2)}\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}, \quad |s_1|_\mathrm{worst} = \sqrt{2}. \tag{508}$$

The simple bound (500) is $\|\bar{M}\|_2 = 1 < |s_1|_\mathrm{worst}$, while (503) matches the maximum $\sqrt{2}$.

Lower Bound on Sum of Vectors The starting point for the new bound is the following lemma:

Lemma E.1. *For any N vectors $x_1, \dots, x_N \in \mathbb{R}^n$, there is a combination of signs $\pm$ such that $\pm x_1 \pm x_2 \pm \dots \pm x_N$ has an euclidean norm of at least $\sqrt{|x_1|^2 + \dots + |x_N|^2}$:*

$$\max_{s_1,\dots,s_N \in \{-1,1\}} \left|\sum_{i=1}^N s_i x_i\right| \geq \sqrt{\sum_{i=1}^N |x_i|^2}. \tag{509}$$



$$|x_1 + x_2| \approx |x_1| + |x_2| \qquad |x_1 \pm x_2| = \sqrt{|x_1|^2 + |x_2|^2} \qquad |x_1 - x_2| \approx |x_1| + |x_2|$$

Figure 48: Geometrical illustration of Lemma E.1: As detailed in Remark E.2, the bound holds with equality if the vectors are orthogonal.

This bound is tight: For any $n \geq 1$ and $N \geq n$ it is met with equality for

$$x_i = \begin{cases} e_i, & i \leq n, \\ 0, & else, \end{cases} \tag{510}$$

where e_i is the i-th base vector.

Remark E.2. This claim and the following proof are motivated by geometrical observations for two vectors x_1 and x_2 in the plane ($n = N = 2$) shown as shown in Figure 48. Consider $\max\{|x_1 + x_2|, |x_1 - x_2|\}$ for fixed lengths $|x_1|$ and $|x_2|$, and a varying angle ϕ between these vectors: It can be seen that $|x_1 + x_2|$ is largest for $\phi = 0$, $|x_1 - x_2|$ is largest for $\phi = \pi$, and the maximum of both is smallest for $\phi = \pi/2$, i.e., if the vectors are orthogonal. This orthogonal case $|x_1 \pm x_2| = \sqrt{|x_1|^2 + |x_2|^2}$ can therefore be used as a lower bound for the general case of the maximum. ◁

Proof of Lemma E.1. For a constructive proof, consider the following algorithm that computes the signs s_i such that $|\sum s_i x_i|$ is at least as large as the lower bound (509):

1. Start with $s_1 = 1$ and $i = 2$.
2. The previously determined variables $s_1, \ldots, s_{i-1}$ are held fixed. Optimize over the single variable $s_i \in \{-1, 1\}$ such that $|\sum_{k=1}^{i} s_k x_k|$ is maximized. This is a greedy strategy because the choice of s_i is only locally optimal but not with respect to the global goal.
3. Increment i and repeat the previous step.

An explicit formula for determining locally optimal s_i can be derived from the fact that, by Bernstein, 2009, Fact 9.7.4.viii,

$$|a \pm b| = \sqrt{|a|^2 + |b|^2 \pm 2a^\top b} \quad \forall a, b \in \mathbb{R}^n : \tag{511}$$

The value $|a \pm b|$ is maximal if the sign $\pm$ is chosen such that $\pm 2a^\top b \geq 0$. Applying this to the algorithm, where $a = \sum_{k=1}^{i-1} s_k x_k$, $b = s_i x_i$ and $s_i = \pm 1$ is the unknown sign, leads to

$$s_1 = 1, \tag{512a}$$

$$s_i = \operatorname{sign}\left(\left(\sum_{k=1}^{i-1} s_k x_k\right)^\top x_i\right), \quad i \geq 2, \tag{512b}$$

where in this context $\operatorname{sign}(0) \coloneqq +1$. Now it can be shown by induction that the chosen s_i is guaranteed to meet the lower bound.

Induction Assumption IA(i): With the above choice of $s_1, \ldots, s_i$,

$$\left|\sum_{k=1}^{i} s_k x_k\right| \geq \sqrt{\sum_{k=1}^{i} |x_k|^2}. \tag{513}$$

Start of Induction: IA(1) is trivially true:

$$|s_1 x_1| = |x_1| \geq \sqrt{|x_1|^2}. \tag{514}$$

Induction Step Let $1 \leq i \leq N$. Assume IA(i) and apply s_{i+1} from (512b) to show IA($i+1$):

$$\left|\sum_{k=1}^{i+1} s_k x_k\right| = \left|\underbrace{\left(\sum_{k=1}^{i} s_k x_k\right)}_{a} + \underbrace{s_{i+1}x_{i+1}}_{b}\right| \tag{515}$$

$$\overset{(511)}{=} \sqrt{\underbrace{\left|\sum_{k=1}^{i} s_k x_k\right|^2}_{\overset{\text{IA}(i)}{\geq} \sum_{k=1}^{i}|x_k|^2} + \underbrace{|s_{i+1}x_{i+1}|^2}_{=|x_{i+1}|^2} + \underbrace{2\left(\sum_{k=1}^{i} s_k x_k\right)^{\mathsf{T}} s_{i+1}x_{i+1}}_{\geq 0 \text{ by } (512)}} \tag{516}$$

$$\geq \sqrt{\sum_{k=1}^{i+1}|x_k|^2} \quad \Rightarrow \text{IA}(i+1). \tag{517}$$

Conclusion IA(i) holds for all $i = 1, \ldots, N$. Consequently,

$$\max_{s_1,\ldots,s_N \in \{-1,1\}} \left|\sum_{i=1}^{N} s_i x_i\right| \geq \left|\sum_{i=1}^{N} \left(s_i x_i\big|_{s_i \text{ per } (512)}\right)\right| \overset{\text{IA}(N)}{\geq} \sqrt{\sum_{i=1}^{N}|x_i|^2}. \tag{518}$$

□

Lower Bound on Generalized Input-to-Output Gain To extend the previous result towards matrix-vector multiplication, consider the vector v that achieves the largest gain from $|v|$ to $|Mv|$:

Lemma E.3 (Maximum Right Singular Vector)**.** *For any matrix $M \in \mathbb{R}^{n\times m}$, there is a unity vector $v_{\max}(M) \in \mathbb{R}^m$, $|v_{\max}(M)| = 1$, such that*

$$|Mv_{\max}(M)| = \|M\|_2 = \max_{x\in\mathbb{R}^m,\ |x|\leq 1} |Mx|. \tag{519}$$

Proof. Any M admits a singular value decomposition (Bernstein, 2009, Theorem 5.6.3) into

$$\underset{n\times m}{M} = \underset{n\times n}{L}\ \underset{n\times m}{S}\ \underset{m\times m}{R}. \tag{520}$$

Herein, L, R are unitary matrices, i.e., $L^\top = L^{-1}$, $R^\top = R^{-1}$ (Bernstein, 2009, Fact 3.11.4.iv) and therefore

$$L^\top L = LL^\top = \underset{n\times n}{I}, \tag{521}$$

$$R^\top R = RR^\top = \underset{m\times m}{I}. \tag{522}$$

S is a $n \times m$ matrix whose $\min\{n, m\}$ diagonal entries are the ordered singular values $\sigma_1(M) \geq \sigma_2(M) \geq \ldots \geq \sigma_{\min\{n,m\}} \geq 0$ of M:

$$S = \begin{bmatrix} \begin{bmatrix} \sigma_1(M) & & \\ & \ddots & \\ & & \sigma_{\min\{n,m\}}(M) \end{bmatrix} & \underbrace{\begin{matrix} 0 & \ldots & 0 \end{matrix}}_{\text{if } m>n} \\ \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} \text{if } n>m & \end{bmatrix} \in \mathbb{R}^{n\times m}. \tag{523}$$

The spectral norm $\|M\|_2$ is defined as the largest singular value of M (Bernstein, 2009, (9.2.15) and Fact 9.13.1):

$$\sigma_1(M) = \|M\|_2 = \max_{x\in\mathbb{R}^m,\ |x|\leq 1} |Mx| \tag{524}$$

Choose $v_{\max}$ as the first column of $R^\top$. This is the right singular vector corresponding to the largest singular value:

$$v_{\max} = \underset{m\times m}{R^\top}\ \underset{m\times 1}{e_1} \quad \Rightarrow \quad |v_{\max}|^2 = v_{max}^\top v_{max} = e_1^\top RR^\top e_1 \overset{(522)}{=} e_1^\top e_1 = 1, \tag{525}$$

$$Rv_{max} = RR^\top e_1 \overset{(522)}{=} \underset{m\times 1}{e_1}, \tag{526}$$

$$SRv_{\max} = \underset{m\times 1}{S\ e_1} \overset{(523)}{=} \sigma_1(M)\ \underset{n\times 1}{e_1} = \|M\|_2\ \underset{n\times 1}{e_1} \tag{527}$$

and therefore,

$$\begin{aligned} |Mv_{\max}|^2 &\overset{(520)}{=} |LSRv_{\max}|^2 \overset{(527)}{=} \left|L\|M\|_2\, e_1\right|^2 \\ &= \|M\|_2^2 e_1^\top L^\top L e_1 \overset{(521)}{=} \|M\|_2^2 e_1^\top e_1 = \|M\|_2^2. \end{aligned} \tag{528}$$

This $v_{\max}$ is a unity vector due to (525) and achieves the maximum of (524) by (528), so the lemma is true. □

Lemma E.4. *For any $M_1, \dots, M_N \in \mathbb{R}^{n\times m}$, $N \geq 1$,*

$$\max_{\substack{x_1,\dots,x_N\in\mathbb{R}^m \\ |x_1|\leq 1,\dots,|x_N|\leq 1}} \left|\sum_{i=1}^{N} M_i x_i\right| \geq \sqrt{\sum_{i=1}^{N} \|M_i\|_2^2}. \tag{529}$$

Proof. Consider $x_i = \pm v_{\max}(M_i)$ with $v_{\max}$ according to Lemma E.3, which is a possible value for x_i since $|v_{\max}(\cdot)| = 1$. Leave the sign $\pm$ as a variable for maximization and invoke Lemma E.1:

$$\max_{\substack{x_1,\dots,x_N\in\mathbb{R}^m \\ |x_1|\leq 1,\dots,|x_N|\leq 1}} \left|\sum_{i=1}^{N} M_i x_i\right| \overset{\substack{\text{restrict to} \\ x_i=\pm v_{\max}(M_i)}}{\geq} \max_{s_1,\dots,s_N\in\{-1,1\}} \left|\sum_{i=1}^{N} \underbrace{s_i}_{\pm 1} M_i v_{\max}(M_i)\right| \tag{530}$$

$$\overset{\text{Lemma E.1}}{\geq} \sqrt{\sum_{i=1}^{N} |M_i v_{\max}(M_i)|^2} \tag{531}$$

$$\overset{\text{Lemma E.3}}{=} \sqrt{\sum_{i=1}^{N} \|M_i\|_2^2}. \tag{532}$$

□

Theorem E.5. *For any given $N \geq 1$ matrices $M_1, \dots, M_N \in \mathbb{R}^{n\times m}$ and bounds $|x_1|_{\max}, \dots, |x_N|_{\max} \geq 0$,*

$$\max_{\substack{x_1,\dots,x_N\in\mathbb{R}^m \\ |x_1|\leq |x_1|_{\max},\dots,|x_N|\leq |x_N|_{\max}}} \left|\sum_{i=1}^{N} M_i x_i\right| \geq \sqrt{\sum_{i=1}^{N} \left(|x_i|_{\max} \|M_i\|_2\right)^2}. \tag{533}$$

Proof. This is a direct consequence of Lemma E.4 and the homogeneity of norms ($\|aM\| = |a|\|M\|$ by (29b)): Assume without loss of generality that $|x_i|_{\max} \neq 0$. Rewrite $M_i x_i = \tilde{M}_i \tilde{x}_i$, where $\tilde{M}_i = M_i |x_i|_{\max}$ and $\tilde{x}_i = x_i / |x_i|_{\max}$, so that $|\tilde{x}_i| \leq |\tilde{x}_i|_{\max} = 1$. Then apply Lemma E.4. □

E.1.4 Pessimism of Bounds

To illustrate the accuracy of upper and lower bounds, consider two examples: First, let $C_0 = 0$, $A(\sigma_i) = \bar{C}_\mathrm{s} = I$,

$$G(\sigma_0) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad G(\sigma_1) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad p = n_d = 2, \quad |d|_\mathrm{max} = 1. \tag{534}$$

While this example does not strictly match the system model, valid examples with equivalent results can be constructed by appropriately padding the matrices. Let $d_0 = \begin{bmatrix} d_{01} & d_{02} \end{bmatrix}^\mathsf{T}$, $d_1 = \begin{bmatrix} d_{11} & d_{12} \end{bmatrix}^\mathsf{T}$, so that $s_2 = \begin{bmatrix} d_{01} & d_{12} \end{bmatrix}^\mathsf{T}$. Therefore, given $|d_0| \leq 1$ and $|d_1| \leq 1$, $|s_2|$ has a maximum of $\sqrt{2}$. Here, the lower bound $\max |s_2| \geq \sqrt{2}$ from (503) is exact, while the upper bound $\max |s_2| \leq 2$ from (501) is pessimistic by a factor of $\sqrt{2}$.

Second, modify the example by setting

$$G(\sigma_1) = G(\sigma_0) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{535}$$

Then, $s_2 = \begin{bmatrix} d_{01} + d_{11} & 0 \end{bmatrix}^\mathsf{T}$, so the maximum is $|s_2| = 2$. Now the lower bound $\max |s_2| \geq \sqrt{2}$ is optimistic by a factor of $\sqrt{2}$ and the upper bound $\max |s_2| \leq 2$ is exact.

E.1.5 Outlook on Possible Improvements

In future work, an optimized lower bound could be derived by optimization methods for constrained quadratic programs. Due to finite numerical precision, these methods usually return an approximate result with unknown error. The amount of error is hard to judge due to the complex and iterative nature of the algorithms involved in constrained optimization.

For the lower bound, numerical error is no problem, as *any* valid disturbance sequence leads to a, possibly suboptimal, lower bound. The upper bound is more difficult: First, one needs to show that the optimization converges towards a global maximum. Second, a rigorous error bound on the approximate result is required, which is supported only by a small number of specialized optimization solvers (Jansson, 2009).

Future work may consider the use of SMT solvers, which were explained in Section 2.5.2.2.

E.2 Abstraction for Varying I/O Timing

For the real-time control system with uncertain timing but without skips (Section 7.5.3), a detailed abstraction can be determined from Theorem 6.5.8 as

$$\rho(\sigma) \geq \max_{|\Delta t_{\{u,y\},\{1,2,\dots\},k}| \leq \sigma \Delta t_{\text{nominal}}} \|A(\Delta t)\|_P =: \rho_{\text{actual}}(\sigma). \tag{536}$$

For a fixed value of σ, $\max \|A\|_P$ is computed according to Section 5.1 as an upper bound using validated numerics: The function $\rho_{\text{actual}}(\sigma)$ is nondecreasing by construction. By the following lemma, it can be approximated as $\rho(\sigma) = c_3\sigma + c_4 \geq \rho_{\text{actual}}(\sigma)$, which is easier to use in decision rules. The approximation uses $N = 150$ samples over a range of $0 \leq \sigma \leq 10$.

Lemma E.6 (Affine Upper Approximation of Nondecreasing Function)**.** *Let $f(x) : [x_0; x_N] \mapsto \mathbb{R}$ be a nondecreasing function and $x_0 < x_1 < \cdots < x_N$ a sequence of $N \geq 2$ sampling points. Choose*

$$\alpha \geq \max_{i=2\dots N} \frac{f(x_i) - f(x_1)}{x_{i-1} - x_0}. \tag{537}$$

Then, as illustrated in Figure 49,

$$f(x) \leq g(x) \coloneqq f(x_1) + \alpha(x - x_0) \quad \textit{for all } x \in [x_0; x_N]. \tag{538}$$

The difference in indices between numerator and denominator is intentional and required for the upper approximation.

Proof. Let f be nondecreasing, i.e.,

$$\bar{x} \geq x \quad \Rightarrow \quad f(\bar{x}) \geq f(x) \tag{539}$$

for all $x, \bar{x} \in [x_0; x_N]$. First, consider $x_0 \leq x \leq x_1$.Then,

$$g(x) = f(x_1) + \alpha(x - x_0) \tag{540}$$

$$\overset{(537)}{\geq} f(x_1) + \frac{\overbrace{f(x_2) - f(x_1)}^{\geq 0 \text{ by } (539)}}{\underbrace{x_1 - x_0}_{>0}} \underbrace{(x - x_0)}_{\geq 0} \geq f(x_1) \overset{(539)}{\geq} f(x). \tag{541}$$

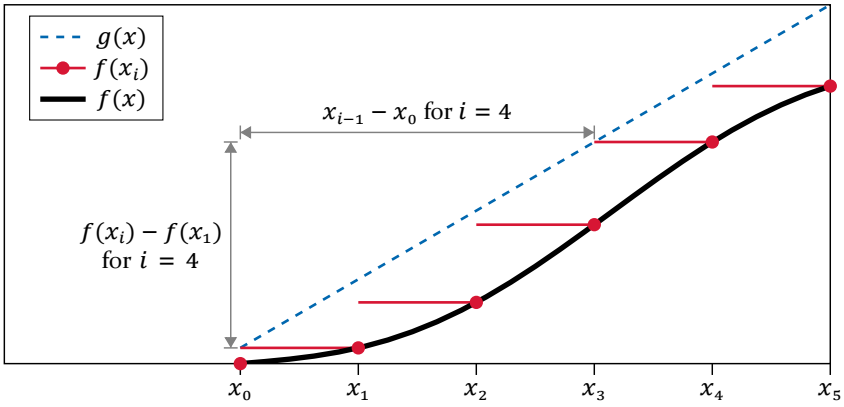


Figure 49: The function $f(x)$ is upper bounded by the affine function $g(x)$ within $x_0 \le x \le x_5$. This bound is obtained by Lemma E.6 from a finite number of samples $f(x_i)$. In this example, the maximum of (537) is attained at $i = 4$.

Next, consider $x_i \le x \le x_{i+1}$ for $1 \le i \le N-1$. Then,

$$g(x) = f(x_1) + \alpha(x - x_0) \tag{542}$$
$$\overset{(537)}{\ge} f(x_1) + (f(x_{i+1}) - f(x_1)) \underbrace{\frac{x - x_0}{x_i - x_0}}_{\ge 1 \text{ because } x \ge x_i > x_0} \ge f(x_{i+1}) \overset{(539)}{\ge} f(x). \qquad \square$$

F Source Code

The source code used for the numerical experiments in this thesis is available as an archive at https://doi.org/10.5281/zenodo.6373637 and as a GIT repository at the branch "mg-diss" of https://github.com/qronos-project/timing-stability-lmi/tree/mg-diss.

The linked archive contains the source code, the output files and a list of which figure corresponds to which files. In particular, the output includes the example systems in SpaceEx HA format. These can be used to benchmark reachability analysis tools.

Bibliography

Adhikary, S., Gurung, A., Thakkar, J., Costa, A. B. D., Dey, S., Hazra, A., Dasgupta, P. (2020): SMT-based Verification of Safety-Critical Embedded Control Software. In: *IEEE Embedded Systems Letters*. DOI: 10.1109/les.2020.3035560.

Akesson, B., Nasri, M., Nelissen, G., Altmeyer, S., Davis, R. I. (2020): *An Empirical Survey-Based Study into Industry Practice in Real-Time Systems*. In: *2020 IEEE Real-Time Systems Symposium (RTSS 2020)*. DOI: 10.1109/RTSS49844.2020.00012.

Al Khatib, M., Girard, A., Dang, T. (2016): Stability Verification and Timing Contract Synthesis for Linear Impulsive Systems Using Reachability Analysis. In: *Nonlinear Analysis: Hybrid Systems*. DOI: 10.1016/j.nahs.2016.08.007.

Althoff, M. (2010): Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars. Dissertation. Technische Universität München. URL: http://mediatum.ub.tum.de/doc/1287517/1287517.pdf.

– (2015): *An Introduction to CORA 2015*. In: *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, 120–151. DOI: 10.29007/zbkv.

Althoff, M., Bak, S., Bao, Z., Forets, M., Frehse, G., Freire, D., Kochdumper, N., Li, Y., Mitra, S., Ray, R., Schilling, C., Schupp, S., Wetzlinger, M. (2020): *ARCH-COMP20 Category Report: Continuous and Hybrid Systems with Linear Continuous Dynamics*. In: *7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*. DOI: 10.29007/7dt2.

Althoff, M., Kochdumper, N. (2018): CORA Manual. URL: https://tumcps.github.io/CORA/data/Cora2018Manual.pdf.

Althoff, M., Yaldiz, S., Rajhans, A., Li, X., Krogh, B. H., Pileggi, L. (2011): *Formal Verification of Phase-locked Loops Using Reachability Analysis and Continuization*. In: *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. DOI: 10.1109/iccad.2011.6105400.

Altisen, K., Gössler, G., Sifakis, J. (2002): Scheduler Modeling Based on the Controller Synthesis Paradigm. In: *Real-Time Systems* 23.1/2, 55–84. DOI: 10.1023/a:1015346419267.

Andersen, M. S., Dahl, J., Vandenberghe, L., et al. (2020): CVXOPT: Python Software for Convex Optimization. URL: http://cvxopt.org/.

Årzén, K.-E., Bernhardsson, B., et al. (1999): *Integrated Control and Scheduling*. Department of Automatic Control, Lund Institute of Technology. URL: http://www.control.lth.se/documents/1999/arz+99t.pdf.

Åström, K. J., Murray, R. M. (2021): *Feedback systems : an introduction for scientists and engineers*. 2nd ed. Princeton University Press.

Åström, K. J., Wittenmark, B. (1996): *Computer-Controlled Systems: Theory and Design*. Prentice Hall.

Bahig, G., El-Kadi, A. (2017): Formal Verification of Automotive Design in Compliance With ISO 26262 Design Verification Guidelines. In: *IEEE Access* 5, 4505–4516. DOI: 10.1109/access.2017.2683508.

Bak, S., Beg, O. A., Bogomolov, S., Johnson, T. T., Nguyen, L. V., Schilling, C. (2017): Hybrid Automata: From Verification to Implementation. In: *International Journal on Software Tools for Technology Transfer*, 1–18. DOI: 10.1007/s10009-017-0458-1.

Bak, S., Bogomolov, S., Johnson, T. T. (2015): *HYST*. In: *18th International Conference on Hybrid Systems Computation and Control - HSCC '15*. DOI: 10.1145/2728606.2728630.

Bak, S., Duggirala, P. S. (2017): *HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems*. In: *20th International Conference on Hybrid Systems: Computation and Control*. Pittsburgh, Pennsylvania, USA, 173–178. DOI: 10.1145/3049797.3049808.

Bak, S., Johnson, T. T. (2015): *Periodically-Scheduled Controller Analysis Using Hybrid Systems Reachability and Continuization*. In: *2015 IEEE Real-Time Systems Symposium*. IEEE. DOI: 10.1109/rtss.2015.26.

Baker, T. P., Shaw, A. (1989): The Cyclic Executive Model and ADA. In: *Real-Time Systems* 1.1, 7–25.

Balakrishnan, V., Boyd, S. (1992): *On Computing the Worst-Case Peak Gain of Linear Systems*. In: *31st IEEE Conference on Decision and Control*. DOI: 10.1109/cdc.1992.371407.

Bauer, N. W., Loon, S. J. L. M. van, Donkers, M. C. F., Wouw, N. van de, Heemels, W. P. M. H. (2012): *Networked Control Systems Toolbox: Robust Stability Analysis Made Easy*. In: *IFAC Proceedings Volumes*. 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems. Vol. 45. 26. Elsevier BV, 55–60. DOI: 10.3182/20120914-2-us-4030.00014.

Ben Gaid, M., Simon, D., Sename, O. (2008): *A Design Methodology for Weakly-Hard Real-Time Control*. In: *IFAC Proceedings Volumes*. 17th IFAC World Congress. Vol. 41. 2, 10258–10264. DOI: 10.3182/20080706-5-KR-1001.01736.

Bernstein, D. S. (2009): *Matrix Mathematics : Theory, Facts, and Formulas*. 2nd ed. Princeton University Press.

– (2018): *Scalar, Vector, and Matrix Mathematics*. Princeton University Press.

Blanchini, F. (2008): *Set-Theoretic Methods in Control*. Birkhäuser. DOI: 10.1007/978-0-8176-4606-6.

Blind, R., Allgöwer, F. (2015): *Towards Networked Control Systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process*. In: *54th IEEE Conference on Decision and Control (CDC)*. DOI: 10.1109/cdc.2015.7403405.

Blondel, V. D., Nesterov, Y., Theys, J. (2005): On the Accuracy of the Ellipsoid Norm Approximation of the Joint Spectral Radius. In: *Linear Algebra and its Applications* 394, 91–107. DOI: 10.1016/j.laa.2004.06.024.

Boyd, S., El Ghaoui, L., Feron, E., Balakrishnan, V. (1994): *Linear matrix inequalities in system and control theory*. Society for Industrial Mathematics. DOI: 10.1137/1.9781611970777.

Briat, C., Seuret, A. (2012): Convex Dwell-Time Characterizations for Uncertain Linear Impulsive Systems. In: *IEEE Transactions on Automatic Control* 57.12, 3241–3246. DOI: 10.1109/tac.2012.2200379.

Bund, T. (2017): Verifikation sicherheitskritischer Regelsysteme unter Beachtung des Zeitverhaltens einer verteilten Rechenplattform. [Verification of Safety-Critical Control Systems Considering the Timing of a Distributed Computing Platform]. PhD thesis. Universität Ulm. DOI: 10.18725/oparu-4463.

Burns, A., Davis, R. (2017): *Mixed Criticality Systems – A Review*. Tech. rep. 9th ed. Department of Computer Science, University of York. URL: https://www-users.cs.york.ac.uk/burns/review.pdf.

Cannon, M., Buerger, J., Kouvaritakis, B., Rakovic, S. (2011): Robust Tubes in Nonlinear Model Predictive Control. In: *IEEE Transactions on Automatic Control* 56.8, 1942–1947. DOI: 10.1109/tac.2011.2135190.

Castane, R., Marti, P., Velasco, M., Cervin, A., Henriksson, D. (2006): *Resource Management for Control Tasks Based on the Transient Dynamics of Closed-loop Systems*. In: *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*, 171–182. DOI: 10.1109/ECRTS.2006.24.

Cervin, A. (2012): *Stability and Worst-Case Performance Analysis of Sampled-Data Control Systems with Input and Output Jitter*. In: *2012 American Control Conference (ACC)*, 3760–3765. DOI: 10.1109/ACC.2012.6315304.

Cervin, A. (2003): Integrated Control and Real-Time Scheduling. PhD thesis. Lund Institute of Technology. URL: https://portal.research.lu.se/files/4543281/8569531.pdf.

Chen, X., Ábrahám, E., Sankaranarayanan, S. (2013): *Flow*: An analyzer for non-linear hybrid systems*. In: *25th International Conference on Computer Aided Verification (CAV 2013)*, 258–263. DOI: 10.1007/978-3-642-39799-8_18.

Clarke, E. M., Henzinger, T. A., Veith, H., Bloem, R., eds. (2018): *Handbook of Model Checking*. Springer. DOI: 10.1007/978-3-319-10575-8.

Cloosterman, M. B. G., Wouw, N. van de, Heemels, W. P. M. H., Nijmeijer, H. (2009): Stability of Networked Control Systems With Uncertain Time-Varying Delays. In: *IEEE Transactions on Automatic Control* 54.7, 1575–1580. DOI: 10.1109/tac.2009.2015543.

Cunha, A. E. C. D. (2015): *Benchmark: Quadrotor Attitude Control*. In: *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*. Vol. 34. EPiC Series in Computing. EasyChair, 57–72. DOI: 10.29007/dc68.

Diamond, S., Boyd, S. (2016): CVXPY: A Python-embedded Modeling Language for Convex Optimization. In: *Journal of Machine Learning Research* 17.83, 1–5. URL: http://jmlr.org/papers/v17/15-408.html.

Donzé, A. (2010): Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In: *22nd International Conference on Computer Aided Verificatio (CAV 2010)*, 167–170. DOI: 10.1007/978-3-642-14295-6_17.

Dorf, R. (2005): *Modern Control Systems*. Pearson Prentice Hall.

el Hakim, V. S., Bekooij, M. J. G. (2018): *Stability Verification of Self-Timed Control Systems Using Model-Checking*. In: *21st Euromicro Conference on Digital System Design (DSD)*. DOI: 10.1109/dsd.2018.00062.

Fan, C., Qi, B., Mitra, S., Viswanathan, M., Duggirala, P. S. (2016): *Automatic Reachability Analysis for Nonlinear Hybrid Models with C2E2*. In: *Computer Aided Verification*. Springer, 531–538. DOI: 10.1007/978-3-319-41528-4_29.

Felicioni, F., Jia, N., Simonot-Lion, F., YeQiong, S. (2008): *Optimal On-line (m,k)-Firm Constraint Assignment for Real-time Control Tasks Based on Plant State Information*. In: *IEEE Intl. Conf. on Emerging Technologies and Factory Automation (ETFA '08)*, 908–915. DOI: 10.1109/ETFA.2008.4638504.

Fontanelli, D., Greco, L., Palopoli, L. (2013): Soft Real-time Scheduling for Embedded Control Systems. In: *Automatica* 49.8, 2330–2338. DOI: 10.1016/j.automatica.2013.04.036.

Frehse, G. (2015): An Introduction to Hybrid Automata, Numerical Simulation and Reachability Analysis. In: *Formal Modeling and Verification of Cyber-Physical Systems*. Springer, 50–81. DOI: 10.1007/978-3-658-09994-7_3.

Frehse, G., Hamann, A., Quinton, S., Woehrle, M. (2014): *Formal Analysis of Timing Effects on Closed-Loop Properties of Control Software*. In: *2014 IEEE Real-Time Systems Symposium*. DOI: 10.1109/rtss.2014.28.

Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O. (2011): *SpaceEx: Scalable Verification of Hybrid Systems*. In: *International Conference on Computer Aided Verification*. Springer, 379–395. DOI: 10.1007/978-3-642-22110-1_30.

Fridman, E., Seuret, A., Richard, J.-P. (2004): Robust Sampled-Data Stabilization of Linear Systems: An Input Delay Approach. In: *Automatica* 40.8, 1441–1446. DOI: 10.1016/j.automatica.2004.03.003.

Gaukler, M. (2015–2020): → See page 266.

Geretti, L., Sandretto, J. A. D., Althoff, M., Benet, L., Chapoutot, A., Chen, X., Collins, P., Forets, M., Freire, D., Immler, F., Kochdumper, N., Sanders, D. P., Schilling, C. (2020): *ARCH-COMP20 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics*. In: *7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*. DOI: 10.29007/zkf6.

Girard, A., Le Guernic, C., Maler, O. (2006): *Efficient Computation Of Reachable Sets Of Linear Time-Invariant Systems With Inputs*. In: *International Workshop on Hybrid Systems: Computation and Control*. Springer, 257–271. DOI: 10.1007/11730637_21.

Giunchiglia, F., Walsh, T. (1992): A Theory of Abstraction. In: *Artificial Intelligence* 57.2-3, 323–389. DOI: 10.1016/0004-3702(92)90021-0.

Greco, L., Fontanelli, D., Bicchi, A. (2011): Design and Stability Analysis for Anytime Control via Stochastic Scheduling. In: *IEEE Transactions on Automatic Control* 56.3, 571–585. DOI: 10.1109/TAC.2010.2058497.

Gruber, F., Althoff, M. (2021): Computing Safe Sets of Linear Sampled-Data Systems. In: *IEEE Control Systems Letters* 5.2, 385–390. DOI: 10.1109/lcsys.2020.3002476.

Guéguen, H., Lefebvre, M.-A., Zaytoon, J., Nasri, O. (2009): Safety Verification and Reachability Analysis for Hybrid Systems. In: *Annual Reviews in Control* 33.1, 25–36. DOI: 10.1016/j.arcontrol.2009.03.002.

Guernic, C. L., Girard, A. (2010): Reachability Analysis of Linear Systems Using Support Functions. In: *Nonlinear Analysis: Hybrid Systems* 4.2, 250–262. DOI: 10.1016/j.nahs.2009.03.002.

Hamdaoui, M., Ramanathan, P. (1995): A Dynamic Priority Assignment Technique for Streams with (m, K)-Firm Deadlines. In: *IEEE Transactions on Computers* 44.12, 1443–1451. DOI: 10.1109/12.477249.

Hansen, J., Hissam, S., Moreno, G. A. (2009): *Statistical-Based WCET Estimation and Validation*. In: *9th International Workshop on Worst-Case Execution Time Analysis (WCET'09)*. DOI: 10.4230/OASIcs.WCET.2009.2291.

Heemels, W., Johansson, K., Tabuada, P. (2012): *An Introduction to Event-Triggered and Self-Triggered Control*. In: *51st Annual Conference on Decision and Control (CDC)*, 3270–3285. DOI: 10.1109/CDC.2012.6425820.

Heemels, W., van de Wouw, N., Gielen, R. H., Donkers, M. C. F., Hetel, L., Olaru, S., Lazar, M., Daafouz, J., Niculescu, S. (2010): *Comparison of Overapproximation Methods for Stability Analysis of Networked Control Systems*. In: *13th ACM international conference on Hybrid systems: computation and control - HSCC '10*. DOI: 10.1145/1755952.1755979.

Henriksson, D., Cervin, A., Åkesson, J., Årzén, K.-E. (2002): *Feedback Scheduling of Model Predictive Controllers*. In: *8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2002)*, 207–216. DOI: 10.1109/RTTAS.2002.1137395.

Henzinger, T. A., Kirsch, C. M., Sanvido, M. A. A., Pree, W. (2003): From Control Models to Real-Time Code Using Giotto. In: *IEEE Control Systems Magazine* 23.1, 50–64. DOI: 10.1109/MCS.2003.1172829.

Henzinger, T. A. (2000): The Theory of Hybrid Automata. In: *Verification of Digital and Hybrid Systems*. Ed. by Inan, M. K.; Kurshan, R. P. Chap. 13, 265–292. DOI: 10.1007/978-3-642-59615-5_13.

Henzinger, T. A., Kopke, P. W., Puri, A., Varaiya, P. (1998): What's Decidable about Hybrid Automata? In: *Journal of Computer and System Sciences* 57.1, 94–124. DOI: 10.1006/jcss.1998.1581.

Hetel, L., Daafouz, J., Iung, C. (2006): Stabilization of Arbitrary Switched Linear Systems With Unknown Time-Varying Delays. In: *IEEE Transactions on Automatic Control* 51.10, 1668–1674. DOI: 10.1109/tac.2006.883030.

Hetel, L. (2007): Robust Stability and Control of Switched Linear Systems. PhD thesis. Institut National Polytechnique de Lorraine. URL: https://tel.archives-ouvertes.fr/tel-00202479/file/Phd_Hetel.pdf.

Hetel, L., Fiter, C., Omran, H., Seuret, A., Fridman, E., Richard, J.-P., Niculescu, S. I. (2017): Recent Developments on the Stability of Systems with Aperiodic Sampling: An Overview. In: *Automatica* 76, 309–335. DOI: 10.1016/j.automatica.2016.10.023.

Horssen, E. P. van, Behrouzian, A. R. B., Goswami, D., Antunes, D., Basten, T., Heemels, W. P. M. H. (2016): *Performance Analysis and Controller Improvement for Linear Systems with (m, K)-Firm Data Losses*. In: *2016 European Control Conference (ECC)*. DOI: 10.1109/ecc.2016.7810677.

Huang, C., Li, W., Zhu, Q. (2019): *Formal Verification of Weakly-Hard Systems*. In: *22nd ACM International Conference on Hybrid Systems Computation and Control - HSCC '19*. DOI: 10.1145/3302504.3311811.

Immler, F. (2015): *Verified Reachability Analysis of Continuous Systems*. In: *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015*. Vol. 9035. Lecture Notes in Computer Science, 37–51. DOI: 10.1007/978-3-662-46681-0_3.

ISO (2018): *Road vehicles — Functional safety — Part 1: Vocabulary*. Stanndard ISO 26262-1:2018(en). International Organization for Standardization.

Jansson, C. (2009): On Verified Numerical Computations in Convex Programming. In: *Japan Journal of Industrial and Applied Mathematics* 26.2-3, 337–363. DOI: 10.1007/bf03186539.

Johansson, F. et al. (2018): mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.1). URL: http://mpmath.org/.

Jungers, R. (2009): *The joint spectral radius: theory and applications*. Springer Science & Business Media. DOI: 10.1007/978-3-540-95980-9.

Kalra, N., Paddock, S. M. (2016): *Driving to Safety. How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* Tech. rep. RR-1478-RC. RAND Corporation. DOI: 10.7249/RR1478.

Kao, C.-Y., Rantzer, A. (2007): Stability Analysis of Systems with Uncertain Time-Varying Delays. In: *Automatica* 43.6, 959–970. DOI: 10.1016/j.automatica.2006.12.006.

Kapinski, J., Krogh, B. H., Maler, O., Stursberg, O. (2003): On Systematic Simulation of Open Continuous Systems. In: *Hybrid Systems: Computation and Control*, 283–297. DOI: 10.1007/3-540-36580-x_22.

Kühn, W. (1998): Rigorously Computed Orbits of Dynamical Systems without the Wrapping Effect. In: *Computing* 61.1, 47–67. DOI: 10.1007/bf02684450.

Lappe, A. (2018a): Erreichbarkeitsanalyse für realitätsnahe Abtastsysteme. [Reachability Analysis for Realistic Sampled-Data Systems]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Lee, E. A., Seshia, S. A. (2016): *Introduction to Embedded Systems*. 2nd ed. MIT Press.

Leva, A., Seva, S., Terraneo, F., Papadopoulos, A. V., Maggio, M. (2020): *How control-friendly is a computing system? And how control-friendly could it be?* In: *IFAC-PapersOnLine*. 21st IFAC World Congress. Vol. 53. 2. Elsevier BV, 7857–7864. DOI: 10.1016/j.ifacol.2020.12.1962.

Linsenmayer, S., Allgöwer, F. (2017): *Stabilization of networked control systems with weakly hard real-time dropout description*. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 4765–4770. DOI: 10.1109/CDC.2017.8264364.

Liu, J. W.-S. (2000): *Real-time systems*. Prentice Hall.

Lohner, R. J. (2001): On the Ubiquity of the Wrapping Effect in the Computation of Error Bounds. In: *Perspectives on Enclosure Methods*. Ed. by Kulisch, U.; Lohner, R.; Facius, A. Springer, 201–216. DOI: 10.1007/978-3-7091-6282-8_12.

Lu, C., Stankovic, J. A., Tao, G., Son, S. H. (1999): *Design and evaluation of a feedback control EDF scheduling algorithm*. In: *20th IEEE Real-Time Systems Symposium*. DOI: 10.1109/real.1999.818828.

Lynch, N., Segala, R., Vaandrager, F., Weinberg, H. B. (1996): *Hybrid I/O automata*. In: *Hybrid Systems III*, 496–510. DOI: 10.1007/bfb0020971.

Lyu, H. (2017): Multivariable control of a rolling spider drone. Master Thesis. University of Rhode Island. DOI: 10.23860/thesis-lyu-haifeng-2017.

Maggio, M., Hamann, A., Mayer-John, E., Ziegenbein, D. (2020): *Control-System Stability Under Consecutive Deadline Misses Constraints*. In: *32nd Euromicro Conference on Real-Time Systems, ECRTS 2020*. Vol. 165. LIPIcs, 21:1–21:24. DOI: 10.4230/LIPIcs.ECRTS.2020.21.

Marti, P., Fuertes, J. M., Fohler, G., Ramamritham, K. (2001): *Jitter Compensation for Real-Time Control Systems*. In: *22nd IEEE Real-Time Systems Symposium (RTSS 2001)*, 39–48. DOI: 10.1109/REAL.2001.990594.

Marti, P., Villa, R., Fuertes, J., Fohle, G. (2001): *On real-time control tasks schedulability*. In: *2001 European Control Conference (ECC)*. IEEE. DOI: 10.23919/ecc.2001.7076255.

Mason, P., Sigalotti, M., Daafouz, J. (2007): *On Stability Analysis of Linear Discrete-Time Switched Systems Using Quadratic Lyapunov Functions*. In: *2007 46th IEEE Conference on Decision and Control*, 5629–5633. DOI: 10.1109/CDC.2007.4434798.

Mattheij, R., Molenaar, J. (2002): *Ordinary Differential Equations in Theory and Practice*. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898719178.

Maurer, M., Gerdes, J. C., Lenz, B., Winner, H., eds. (2016): *Autonomous Driving*. Springer. DOI: 10.1007/978-3-662-48847-8.

Michel, A. N., Wang, K., Hu, B. (2001): *Qualitative Theory of Dynamical Systems*. 2nd ed. Taylor & Francis. DOI: 10.1201/9780203908297.

Miller, J. D. (2020): *Automotive System Safety: Critical Considerations for Engineering and Effective Management*. Wiley.

Miskowicz, M., ed. (2015): *Event-Based Control and Signal Processing*. CRC Press. DOI: 10.1201/b19013.

Möhlmann, E., Theel, O. (2013): *Stabhyli – A Tool for Automatic Stability Verification of Non-Linear Hybrid Systems*. In: *16th international conference on Hybrid systems: computation and control - HSCC '13*. DOI: 10.1145/2461328.2461347.

Nešić, D., Teel, A. R., Sontag, E. D. (1999): Formulas Relating KL Stability Estimates of Discrete-Time and Sampled-Data Nonlinear Systems. In: *Systems & Control Letters* 38.1, 49–60. DOI: 10.1016/s0167-6911(99)00046-8.

Nilsson, J., Bernhardsson, B., Wittenmark, B. (1998): Stochastic Analysis and Control of Real-Time Systems with Random Time Delays. In: *Automatica* 34.1, 57–64. DOI: 10.1016/S0005-1098(97)00170-2.

Pazzaglia, P., Mandrioli, C., Maggio, M., Cervin, A. (2019): *DMAC: Deadline-Miss-Aware Control*. In: *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*. Vol. 133, 1:1–1:24. DOI: 10.4230/LIPIcs.ECRTS.2019.1.

Philippe, M., Athanasopoulos, N., Angeli, D., Jungers, R. M. (2019): On Path-Complete Lyapunov Functions: Geometry and Comparison. In: *IEEE Trans. Autom. Control.* 64.5, 1947–1957. DOI: 10.1109/TAC.2018.2863380.

Platzer, A. (2018): *Logical Foundations of Cyber-Physical Systems*. Springer.

Raković, S. V., Levine, W. S., Açikmese, B. (2016): *Elastic Tube Model Predictive Control*. In: *2016 American Control Conference (ACC)*. DOI: 10.1109/acc.2016.7525471.

Ramanathan, P. (1997): *Graceful Degradation in Real-Time Control Applications Using (m, k)-Firm Guarantee*. In: *Proceedings of IEEE 27th International Symposium on Fault Tolerant Computing*, 132–141. DOI: 10.1109/FTCS.1997.614086.

Ravanbakhsh, H. (2018): Inductive Certificate Synthesis for Control Design. PhD thesis. University of Colorado, arXiv: 1805.01822v1 [cs.SY].

Reissig, G., Weber, A., Rungger, M. (2017): Feedback Refinement Relations for the Synthesis of Symbolic Controllers. In: *IEEE Transactions on Automatic Control* 62.4, 1781–1796. DOI: 10.1109/tac.2016.2593947.

Rheinfels, T. (2019): Leveraging Machine-Learning Techniques for Quality-Aware Real-Time Scheduling. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Roux, P., Jobredeaux, R., Garoche, P.-L., Féron, É. (2012): *A Generic Ellipsoid Abstract Domain for Linear Time Invariant Systems*. In: *15th ACM international conference on Hybrid Systems: Computation and Control - HSCC '12*. DOI: 10.1145/2185632.2185651.

Rump, S. M. (2010): Verified Bounds for Singular Values, in Particular for the Spectral Norm of a Matrix and Its Inverse. In: *BIT Numerical Mathematics* 51.2, 367–384. DOI: 10.1007/s10543-010-0294-0.

Schinkel, M., Chen, W.-H., Rantzer, A. (2002): *Optimal Control for Systems with Varying Sampling Rate*. In: *American Control Conference 2002*. Vol. 4, 2979–2984. DOI: 10.1109/ACC.2002.1025245.

Scokaert, P. O. M., Mayne, D. Q., Rawlings, J. B. (1999): Suboptimal model predictive control (feasibility implies stability). In: *IEEE Transactions on Automatic Control* 44.3, 648–654. DOI: 10.1109/9.751369.

Seto, D., Krogh, B. H., Sha, L., Chutinan, A. (1998): Dynamic Control System Upgrade Using the Simplex Architecture. In: *IEEE Control Systems* 18.4, 72–80. DOI: 10.1109/37.710880.

Seto, D., Lehoczky, J. P., Sha, L., Shin, K. G. (1996): *On Task Schedulability in Real-Time Control Systems*. In: *Real-Time Systems Symposium, 1996., 17th IEEE*, 13–21. DOI: 10.1109/REAL.1996.563693.

Seuret, A. (2011): *Stability Analysis of Networked Control Systems with Asynchronous Sampling and Input Delay*. In: *2011 American Control Conference*. DOI: 10.1109/acc.2011.5990854.

– (2012): A Novel Stability Analysis of Linear Systems under Asynchronous Samplings. In: *Automatica* 48.1, 177–182. DOI: 10.1016/j.automatica.2011.09.033.

Sha, L., Goodenough, J. B. (1989): *Real-Time Scheduling Theory and ADA*. Tech. rep. CMU/SEI-89-TR-14. Carnegie-Mellon University, Pittsburgh PA, Software Engineering Institute.

Shorten, R., Wirth, F., Mason, O., Wulff, K., King, C. (2007): Stability Criteria for Switched and Hybrid Systems. In: *SIAM Review* 49.4, 545–592. DOI: 10.1137/05063516x.

Simon, D., Seuret, A., Sename, O. (2012): *On Real-Time Feedback Control Systems: Requirements, Achievements and Perspectives*. In: *Systems and Computer Science (ICSCS), 2012 1st International Conference on*, 1–6. DOI: 10.1109/IConSCS.2012.6502458.

Simon, D., Seuret, A., Sename, O. (2017): Real-Time Control Systems: Feedback, Scheduling and Robustness. In: *International Journal of Systems Science* 48.11, 2368–2378. DOI: 10.1080/00207721.2017.1316879.

Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Sastry, S. (2005): *An LQG Optimal Linear Controller for Control Systems with Packet Losses*. In: *44th IEEE Conference on Decision and Control*. DOI: 10.1109/cdc.2005.1582198.

Stover, C. (2019): Limit. MathWorld. Retrieved 2019-09-23. URL: http://mathworld.wolfram.com/Limit.html.

Tanenbaum, A. (2009): *Modern Operating Systems*. Pearson.

Tomlin, C. J., Mitchell, I., Bayen, A. M., Oishi, M. (2003): Computational techniques for the verification of hybrid systems. In: *Proceedings of the IEEE* 91.7, 986–1001. DOI: 10.1109/JPROC.2003.814621.

Tran, H.-D., Nguyen, L. V., Xiang, W., Johnson, T. T. (2017): Order-Reduction Abstractions for Safety Verification of High-Dimensional Linear Systems. In: *Discrete Event Dynamic Systems* 27.2, 443–461. DOI: 10.1007/s10626-017-0244-y.

Ulbrich, P., Gaukler, M. (2019): → See page 266.

VanAntwerp, J. G., Braatz, R. D. (2000): A Tutorial on Linear and Bilinear Matrix Inequalities. In: *Journal of Process Control* 10.4, 363–385. DOI: 10.1016/s0959-1524(99)00056-6.

Vankeerberghen, G., Hendrickx, J., Jungers, R. M. (2014): *JSR*. In: *17th international conference on Hybrid systems: computation and control - HSCC '14*. DOI: 10.1145/2562059.2562124.

Walden, D. D., Roedler, G. J., Forsberg, K., Hamelin, R. D., Shortell, T. M., eds. (2015): *Systems Engineering Handbook*. Wiley.

Wittenmark, B. (2001): *Sample-induced delays in synchronous multirate systems*. In: *2001 European Control Conference (ECC)*. DOI: 10.23919/ecc.2001.7076438.

Wu, J. (2019): Sicherheitsverifikation von Echtzeitregelungen mittels Erreichbarkeitsanalyse und Continuization. [Safety Verification of Real-Time Control by Reachability Analysis and Continuization]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Xia, F., Tian, Y.-C., Sun, Y., Dong, J. (2008): Control-Theoretic Dynamic Voltage Scaling for Embedded Controllers. In: *IET Computers & Digital Techniques* 2.5, 377. DOI: 10.1049/iet-cdt:20070112.

Xia, F., Sun, Y. (2008): *Control and Scheduling Codesign: Flexible Resource Management in Real-Time Control Systems*. Springer Zhejiang University Press.

Ye, H., Michel, A. N., Hau, L. (1996): *Stability Analysis of Discontinuous Dynamical Systems with Applications*. In: *IFAC Proceedings Volumes*. 13th World Congress of IFAC. Vol. 29. 1, 2335–2340. DOI: 10.1016/s1474-6670(17)58022-2.

Yurichev, D. (2018): SAT/SMT by example. Version 2018-09-22. URL: https://yurichev.com/writings/SAT_SMT_by_example.pdf.

Zamani, M., Pola, G., Mazo, M., Tabuada, P. (2012): Symbolic Models for Nonlinear Control Systems Without Stability Assumptions. In: *IEEE Transactions on Automatic Control* 57.7, 1804–1809. DOI: 10.1109/tac.2011.2176409.

Own Publications

Gaukler, M. (2015): Ein Modell zur Bewertung der Regelgüte bei Auftreten von Ein-Ausgangs-Latenzen in digitalen Regelkreisen. [A Model to Evaluate Quality-Of-Control in Digital Control Loops with Input/Output Latencies]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg. URL: https://nbn-resolving.org/urn:nbn:de:bvb:29-opus4-69079.

– (2020a): *Analysis of Real-Time Control Systems using First-Order Continuization*. In: *7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*. Vol. 74. EPiC Series in Computing. EasyChair, 209–241. DOI: 10.29007/8nq6.

– (2020b): *Hard Real-Time not Required: Safe Control Systems with Flexible Timing*. Internal report (not published). Version 2. Friedrich-Alexander-Universität Erlangen-Nürnberg, Lehrstuhl für Regelungstechnik.

Gaukler, M., Michalka, A., Ulbrich, P., Klaus, T. (2018): *A New Perspective on Quality Evaluation for Control Systems with Stochastic Timing*. In: *21st International Conference on Hybrid Systems: Computation and Control (HSCC '18)*. ACM Press. DOI: 10.1145/3178126.3178134.

Gaukler, M., Rheinfels, T., Ulbrich, P., Roppenecker, G. (2019): *Convergence Rate Abstractions for Weakly-Hard Real-Time Control*. Tech. rep. DOI: https://doi.org/10.48550/arXiv.1912.09871.

Gaukler, M., Roppenecker, G., Ulbrich, P. (2019): Details and Proofs for: Stability Analysis of Multivariable Digital Control Systems with Uncertain Timing. Extended version of (Gaukler, Roppenecker, and Ulbrich, 2020). DOI: https://doi.org/10.48550/arXiv.1911.02537.

– (2020): *Stability Analysis of Multivariable Digital Control Systems with Uncertain Timing*. In: *IFAC-PapersOnLine*. 21st IFAC World Congress. Vol. 53. 2, 3085–3091. DOI: 10.1016/j.ifacol.2020.12.1019.

Gaukler, M., Ulbrich, P. (2019): *Worst-Case Analysis of Digital Control Loops with Uncertain Input/Output Timing*. In: *ARCH19. International Workshop on Applied veRification for Continuous and Hybrid Systems*. DOI: 10.29007/c4zl.

Ulbrich, P., Gaukler, M. (2019): *QRONOS: Towards Quality-Aware Responsive Real-Time Control Systems*. In: *Brief Presentations Track of the 25th IEEE Real-Time and Embedded Technology and Applications Symposium (BP-RTAS'19)*, 21–24. URL: http://2019.rtas.org/wp-content/uploads/2019/04/RTAS19_BP_proceedings.pdf#page=27.

Student Works

Albrecht, F. (2017): Bedarfsgesteuerte Abtastung in Echtzeitregelungssystemen. [Demand-Based Sampling in Real-Time Control Systems]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Apfel, Q. (2019): Sichere Regelung eines Quadrokopters. [Safe Quadrocopter Control]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Buttazzo, L. (2020): Kamera-basierte Regelung und Sicherheitsüberwachung für den Quadrokopter „Parrot Mambo". [Camera-Based Control and Safety Monitoring for the Quadrocopter Parrot Mambo]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Fleischmann, A. (2019): Automatische Stabilitätsuntersuchung nichtlinearer Systeme mittels Sum-of-Squares Lyapunov-Funktionen. [Automatic Stability Analysis of Nonlinear Systems Using Sum-of-Squares Lyapunov Functions]. Seminar report. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Frieß, L. (2018a): Evaluation von Software zur Erreichbarkeitsanalyse. [Evaluation of Software for Reachability Analysis]. Bachelor thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

– (2018b): Modellierung und Erreichbarkeitsanalyse hybrider Systeme. [Modeling and Reachability Analysis of Hybrid Systems]. Seminar report. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Köhler, C. (2019): Timing-Unsicherheiten in Echtzeitregelungen: Theorie und Praxis. [Timing Uncertainties in Real-Time Control: Theory and Practice]. Bachelor thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Lappe, A. (2018a): Erreichbarkeitsanalyse für realitätsnahe Abtastsysteme. [Reachability Analysis for Realistic Sampled-Data Systems]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

– (2018b): Mengenwertige Störgrößenrekonstruktion am Versuch Verladebrücke. [Set-Valued Disturbance Reconstruction for the Bridge Crane Experiment]. Research internship. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Lindner, T. (2020): Using a One Dimensional Growth Bound Abstraction for Quality-Aware Scheduling of Real-Time Control Systems. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Mehlich, J. (2019): Weiterentwicklung eines Quadrocopter-Versuchsstands. [Further Development of a Quadrocopter Test Rig]. Project thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Rauh, A. (2016): Konzeption und Aufbau eines Versuchsstands für einen Quadrocopter. [Design and Implementation of a Quadrocopter Test Rig]. Project thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Reichenstein, T. (2018): Weiterentwicklung eines Quadrocopter-Versuchsstands. [Further Development of a Quadrocopter Test Rig]. Project thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Rheinfels, T. (2019): Leveraging Machine-Learning Techniques for Quality-Aware Real-Time Scheduling. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Spenger, P. (2017a): Ermittlung von Timing-Anforderungen für Echtzeitregelungssysteme mittels Erreichbarkeitsanalyse. [Using Reachability Analysis to Determine Timing Requirements for Real-Time Control Systems]. Bachelor thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

– (2017b): Erreichbarkeitsanalyse am Beispiel linearer Systeme. [Reachability Analysis for the Example of Linear Systems]. Seminar report. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Wu, J. (2019): Sicherheitsverifikation von Echtzeitregelungen mittels Erreichbarkeitsanalyse und Continuization. [Safety Verification of Real-Time Control by Reachability Analysis and Continuization]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Yang, R. (2019): Modellierung und Regelung eines Quadrokopters. [Modelling and Control of a Quadrocopter]. Master thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Zhou, Z. (2018): Erstellung von Rechnerübungen zur Vorlesung „Optimalsteuerung“. [Computer Exercises for the Lecture on Optimal Control]. Research internship. Friedrich-Alexander-Universität Erlangen-Nürnberg.

The topic of this work is the analysis of control loops under the influence of input/output timing deviations. Such deviations from the ideal periodic timing can lead to increased control error or even to instability. To address this issue, methods are presented to determine a safe tolerance band for the timing deviation. Two alternatives are addressed: stability analysis at design time and timing adaptation at run time. In both cases, the methods support multiple inputs and outputs with individual timing uncertainties.

For the analysis at design time, the work presents methods to determine stability and maximal control error at design time for a given input/output time window. The linear case is analyzed using linear impulsive systems. Progress towards solving the nonlinear case is made by reachability analysis of hybrid automata in connection with the Continuization method.

For a flexible adaptation at run time, the framework of convergence rate abstractions is developed. By this, the decision about the permissible input/output timing can be made according to the current situation and with little computational overhead. Hence, larger temporal deviations can be permitted for a short time as long as the timing remains good enough in the long-term average to preserve stability.

ISBN 978-3-96147-637-4

FAU UNIVERSITY PRESS 2023