Lehrstuhl für Steuerungs- und Regelungstechnik
Technische Universität München
Univ.-Prof. Dr.-Ing. (Univ. Tokio) Martin Buss

Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars

Matthias Althoff

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Ralph Kennel
Prüfer der Dissertation:		
	1.	Univ.-Prof. Dr.-Ing. (Univ. Tokio) Martin Buss
	2.	Univ.-Prof. Dr.-Ing. Olaf Stursberg, Universität Kassel

Die Dissertation wurde am 22.02.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07.07.2010 angenommen.

Foreword

This thesis summarizes my research during the last four years at the Institute of Automatic Control Engineering (LSR) of the Technische Universität München. First of all, I would like to thank all of my colleagues for the wonderful time we shared at the institute. Besides this thank-you, I would like to mention some people who contributed to this thesis.

Clearly, this thesis would not have been possible without my Ph.D. advisor Prof. Martin Buss who offered me a Ph.D. position and showed great trust in my abilities. I would also like to thank Prof. Olaf Stursberg for his valuable advice and his discussions with me, especially at the beginning of my research. Many thanks also go to Mrs. Lukowski, Mrs. Schmid, Mrs. Werner, Mrs. Renner, and Dirk Wollherr for the administration of my traveling, my students, and all other things that keep the institute running. I am also grateful for the work of the system administrators Mr. Jaschik, Jens Hölldampf, Ulrich Unterhinninghofen, Matthias Rungger, Thomas Schauß, and Stefan Sosnowski.

This thesis has been supported by the Master thesis of Igor Shevchenko and the Bachelor theses of Buelent Sari and Alexander Mergel. I would also like to thank the proofreaders Andrea Bauer, Iason Vittorias, Michelle Karg, Tingting Xu, Michael Strolz, and my brother Daniel for improving this thesis. I am especially delighted that Prof. Thao Dang from the research center Verimag in Grenoble gave me valuable advice for my thesis.

I also have to mention the stimulus from the great atmosphere and funny moments in the office that I have shared with Marc Ueberle, Michael Strolz, and Sachit Rao in the course of time.

Finally, I would like to thank my parents for their enduring support which brought me into the position of writing this thesis, and their sympathy and understanding when I had hard times.

Munich, February 2010 Matthias Althoff

Contents

Notations

Conventions

The following fonts are used for different types of symbols:

- Sets: Number sets are denoted by blackboard bold letters ($\mathbb{N}$, $\mathbb{R}$, $\dots$), interval sets are denoted by Zapf Chancery letters ($\mathpzc{P}$, $\mathpzc{X}$, $\dots$), and other sets in Euclidean space are denoted by standard calligraphic letters ($\mathcal{P}$, $\mathcal{X}$, $\dots$).
- Sets of matrices: Interval matrices are denoted analogously to interval sets by Zapf Chancery letters ($\mathpzc{A}$, $\mathpzc{B}$, $\dots$), zonotope matrices by standard calligraphic letters with raised $\mathcal{Z}$ ($\mathcal{A}^{\mathcal{Z}}$, $\mathcal{B}^{\mathcal{Z}}$, $\dots$), and general sets of matrices by standard calligraphic letters ($\mathcal{A}$, $\mathcal{B}$, $\dots$).
- Left and right limits of intervals or interval matrices are denoted by over- and underlines, respectively, e.g. $\mathpzc{A} \in [\underline{A}, \overline{A}]$.
- Scalars and vectors are denoted by lowercase letters. Matrices and variables indexed by more than two indices (tensors) are denoted by uppercase letters.
- Random variables are denoted by bold letters.
- Enclosing hull variables are denoted by Ralph Smith's Formal Script Font ($\mathscr{H}$, $\mathscr{R}$, $\dots$).
- Functions and operators are denoted by typewriter letters ($\texttt{inv}()$, $\texttt{CH}()$, $\dots$).

Elements of a vector c are written as c_i, elements of a matrix A as A_{ij}, and elements of a tensor of third order Φ as Φ_{ijk}, and so on. Note that in this thesis, it does not make a difference if the indices are superscripts or subscripts, such that e.g. $\Phi_{ijk} = \Phi_{ij}^{k}$. A matrix with a single index such as A_i refers to the column or row of A. A tensor of third order Φ with only a single index (e.g. Φ^k) refers to a matrix within the tensor, and to a vector when two indices are used (e.g. Φ_{ij}). The presented indexing of variables also applies to sets of matrices and random variables.

In formulas, a dot is sometimes used to increase the readability when variables are multiplied (e.g. $a \cdot t$). This should not be confused with the scalar product when two vectors are multiplied, which is denoted as $c^T x$ ($c, x \in \mathbb{R}^n$).

Abbreviations

EPH	enclosing probabilistic hull

LTI	linear time invariant
LTV	linear time variant
PCA	principal component analysis
PDF	probability density function
SUE	single-use expression

Subscripts and Superscripts

$\square^0$	initial value
$\square^{end}$	value at the end
$\square_{\mathtt{g}}$	guard set of a hybrid system
$\square^{\text{inter}}$	intersection crossing
$\square_{\mathtt{inv}}$	invariant of a hybrid system
$\square^{\max}$	maximum value
$\square^{\text{road}}$	road following
$\square^{start}$	value at the start
$\square^{TP}$	traffic participant
$\square^{\text{vehicle}}$	vehicle following

Sets

Number Sets

$\mathbb{N}$	set of natural numbers
$\mathbb{N}^+$	set of positive natural numbers
$\mathbb{Q}^+$	set of positive rational numbers
$\mathbb{R}$	set of real valued numbers
$\mathbb{R}^+$	set of positive real valued numbers

Set of Intervals and Events

$\mathcal{I}$	set of real valued intervals
Ω	set of elementary events

Discrete Sets

Y	$\subset \mathbb{N}^+$	set of locations of an automaton
$\mathcal{R}^z$	$\subset \mathbb{N}^+$	discrete reachable set
$\mathtt{T}$	$\subset \mathbb{N}^+ \times \mathbb{N}^+$	set of discrete transition of a hybrid automaton

Sets in Euclidean space

$\mathcal{B}$	$\subset \mathbb{R}^2$	set of vehicle bodies
$\mathcal{C}$	$\subset \mathbb{R}^2$	set of vehicle centers
$\mathcal{D}$	$\subset \mathbb{R}^n$	linearly transformed input set
$\mathscr{d}$	$\subset \mathcal{I}$	deviation interval of traffic participants
$\mathcal{H}$	$\subset \mathbb{R}^n$	halfspace
$\mathcal{H}^{\mathcal{R}}$	$\subset \mathbb{R}^n$	reachable set of the homogeneous solution
I	$\subset \mathcal{I}^n$	general multidimensional interval
$\mathscr{L}$	$\subset \mathbb{R}^n$	set of linearization errors
$\bar{\mathscr{L}}$	$\subset \mathbb{R}^n$	set of admissible linearization errors
$\mathcal{P}$	$\subset \mathbb{R}^n$	polytope
$\mathcal{P}^{\mathcal{R}}$	$\subset \mathbb{R}^n$	reachable set of the inhomogeneous solution
$\tilde{\mathcal{P}}$	$\subset \mathbb{R}^n$	reachable set due to the constant input $\tilde{u}$
$\mathscr{P}$	$\subset \mathcal{I}^p$	multidimensional interval of parameters
$\mathcal{R}$	$\subset \mathbb{R}^n$	reachable set
$\tilde{\mathcal{R}}$	$\subset \mathbb{R}^n$	reachable set defined as $\mathcal{H}^{\mathcal{R}} + \tilde{\mathcal{P}}$
$\bar{\mathcal{R}}$	$\subset \mathbb{R}^n$	admissible reachable set used for the linearization procedure
$\mathcal{R}^e$	$\subset \mathbb{R}^n$	exact reachable set
$\mathcal{R}^{err}$	$\subset \mathbb{R}^n$	reachable set due to linearization errors
$\bar{\mathcal{R}}^{err}$	$\subset \mathbb{R}^n$	admissible reachable set due to linearization errors
$\mathcal{R}^H$	$\subset \mathbb{R}^n$	reachable set in halfspace representation
$\mathcal{R}^{lin}$	$\subset \mathbb{R}^n$	reachable set of the linearized system
$\mathcal{S}$	$\subset \mathbb{R}^n$	hyperplane
$\mathscr{s}$	$\subset \mathcal{I}$	position interval of traffic participants
$\mathcal{U}$	$\subset \mathbb{R}^m$	input set
$\tilde{\mathcal{U}}$	$\subset \mathbb{R}^m$	translated input set
$\mathscr{U}$	$\subset \mathcal{I}^m$	multidimensional interval of inputs
$\mathscr{v}$	$\subset \mathcal{I}$	velocity interval of traffic participants
$\mathcal{X}$	$\subset \mathbb{R}^n$	general set of continuous states
$\mathcal{X}^{\text{unsafe}}$	$\subset \mathbb{R}^n$	set of unsafe states
$\mathscr{X}$	$\subset \mathcal{I}^n$	multidimensional interval of continuous states
$\mathcal{Y}$	$\subset \mathbb{R}^n$	set outside the gamma confidence set
$\mathcal{Z}$	$\subset \mathbb{R}^n$	zonotope
$\tilde{\mathcal{Z}}$	$\subset \mathbb{R}^n$	reduced part of zonotope
$\check{\mathcal{Z}}$	$\subset \mathbb{R}^n$	unreduced part of zonotope
$\mathcal{Z}^{encl}$	$\subset \mathbb{R}^n$	enclosing zonotope
$\mathcal{Z}^{red}$	$\subset \mathbb{R}^n$	reduced zonotope
$\hat{\mathcal{Z}}$	$\subset \mathbb{R}^{n+m}$	combined set of reachable states and inputs $\mathcal{R} \times \mathcal{U}$
Ψ	$\subset \mathbb{R}^n$	parallelotope

Sets of Matrices

$\mathcal{A}$	$\subset \mathbb{R}^{n\times n}$	set of system matrices
$\mathscr{A}$	$\subset \mathcal{I}^{n\times n}$	interval matrix of system matrices
$\mathcal{A}^{\mathcal{Z}}$	$\subset \mathcal{I}^{n\times n}$	matrix zonotope of system matrices
$\mathcal{B}$	$\subset \mathbb{R}^{n\times m}$	set of input matrices

$\mathcal{B}$	$\subset \mathcal{I}^{n\times m}$	interval matrix of input matrices
$\mathcal{B}^{\mathcal{Z}}$	$\subset \mathcal{I}^{n\times m}$	matrix zonotope of input matrices
$\mathcal{C}$	$\subset \mathcal{I}^{n\times n}$	integral of exponential interval matrix $e^{\mathcal{A}\cdot t}$
$\mathcal{D}$	$\subset \mathcal{I}^{n\times n}$	symmetric part of interval exponential matrix $e^{\mathcal{A}\cdot r}$
$\mathcal{E}$	$\subset \mathcal{I}^{n\times n}$	interval matrix of Taylor series remainders
$\mathcal{F}$	$\subset \mathcal{I}^{n\times n}$	interval matrix for errors of time interval solutions
$\tilde{\mathcal{F}}$	$\subset \mathcal{I}^{n\times n}$	same as $\mathcal{F}$, but for solutions due to constant input $\tilde{u}$
$\mathcal{L}$	$\subset \mathcal{I}^{n\times n}$	general interval matrix for linear transformations
$\mathcal{L}^{\mathcal{Z}}$	$\subset \mathbb{R}^{n\times n}$	linear part of the zonotope matrix exponential
$\hat{\mathcal{L}}^{\mathcal{Z}}$	$\subset \mathbb{R}^{n\times n}$	preliminary linear part of the zonotope matrix exponential
$\mathcal{N}^{\mathcal{Z}}$	$\subset \mathbb{R}^{n\times n}$	nonlinear part of zonotope matrix exponential
$\mathcal{S}$	$\subset \mathcal{I}^{n\times n}$	symmetric interval matrix
$\mathcal{W}$	$\subset \mathcal{I}^{n\times n}$	exact interval matrix of the second order Taylor expansion

Variables

$\mathsf{0}$	$\in \mathbb{R}^{n}$	origin
$\mathbf{0}$	$\in \mathbb{R}^{n\times n}$	matrix of zeros
$\mathbf{1}$	$\in \mathbb{R}^{n\times n}$	matrix of ones
A	$\in \mathbb{R}^{n\times n}$	system matrix
$\hat{A}$	$\in \mathbb{R}^{n\times n}$	center/generator matrix of an uncertain system matrix
$\tilde{A}$	$\in \mathbb{R}^{n\times n}$	vertex matrix of an uncertain system matrix
$\breve{A}$	$\in \mathbb{R}^{n\times n}$	sample matrix which is element of $\mathcal{A}^{\mathcal{Z}}$
a	$\in \mathbb{R}^{+}$	absolute acceleration of a traffic participant
$\bar{a}_d$	$\in \mathbb{R}^{+}$	d-th absolute acceleration which is between 0 and $a^{\max}$
a_T	$\in \mathbb{R}$	tangential acceleration of a traffic participant
a_N	$\in \mathbb{R}$	normal acceleration of a traffic participant
$\hat{a}$	$\in \mathbb{R}^{n^2}$	vertex vector representing a vertex matrix of $\mathcal{A}$
B	$\in \mathbb{R}^{n\times m}$	input matrix
C	$\in \mathbb{R}^{q\times n}$	C-matrix of polytopes
c	$\in \mathbb{R}^{n}$	center of a zonotope
d	$\in \mathbb{R}^{q}$	d-vector of polytopes
d^{lin}	$\in \mathbb{R}^{n}$	constant vector obtained after system linearization
e	$\in \mathbb{R}^{n}$	unit vector
f	$\in \mathbb{R}^{n}$	alternative variable for a generator of a zonotope
$\mathscr{f}$	$\in \mathbb{R}^{n}$	alternative variable for a generator of a Gaussian zonotopes
G	$\in \mathbb{R}^{n\times e}$	matrix of zonotope generators
$\mathcal{G}$	$\in \mathbb{R}^{n\times o}$	matrix of Gaussian zonotope generators
g	$\in \mathbb{R}^{n}$	generator of a zonotope
$\mathscr{g}$	$\in \mathbb{R}^{n}$	generator of a Gaussian zonotope
H	$\in \mathbb{R}^{n\times(n-1)}$	matrix of zonotope generators spanning a facet
h	$\in \mathbb{R}^{n}$	h-vector to compute the n-dimensional cross product
I	$\in \mathbb{R}^{n\times n}$	identity matrix
J	$\in \mathbb{R}^{n\times n\times n}$	second derivative of flow function

K	$\in \mathbb{R}^{n\times n}$	matrix of the jump function $\mathtt{h}$
k	$\in \mathbb{R}^{n}$	vector of the jump function $\mathtt{h}$
L	$\in \mathbb{R}^{n\times n}$	matrix for linear part of the zonotope matrix exponential
l	$\in \mathbb{R}^{+}$	length (2-norm) of a generator
$\hat{l}$	$\in \mathbb{R}^{n}$	line segment for the definition and construction of zonotopes
$\bar{\ell}$	$\in \mathbb{R}^{n}$	bounding vector for the set of admissible linearization errors
ℓ	$\in \mathbb{R}^{n}$	bounding vector for the set of absolute linearization errors
N	$\in \mathbb{R}^{n\times n}$	center of the nonlinear part of the zonotope matrix exponential
P	$\in \mathbb{R}^{2\times n}$	projection matrix for plotting of reachable sets
p	$\in [0,1]^{d\times c}$	probability of discrete states and inputs
p^{e}	$\in [0,1]^{d\times c}$	exact probability of discrete states and inputs
$\hat{p}$	$\in [0,1]^{d}$	probability of discrete states
$\tilde{p}$	$\in [0,1]^{d\cdot c}$	combined probability vector of state and input
p^{lc}	$\in [0,1]$	probability for a lane change
$p^{\max}$	$\in [0,1]$	maximum probability for computational time reduction
p^{crash}	$\in [0,1]$	probability of the autonomous vehicle for a crash
p^{path}	$\in [0,1]^{a}$	path segment probability of a traffic participant
p^{dev}	$\in [0,1]^{b}$	deviation segment probability of a traffic participant
p^{pos}	$\in [0,1]^{a\times b}$	position trapezoid probability of a traffic participant
p^{int}	$\in [0,1]^{a\times b\times a\times b}$	intersection probability of two position trapezoids
$\bar{p}$	$\in [0,1]$	over-approximated probability of hitting an unsafe set
Q	$\in \mathbb{R}^{n\times n}$	matrix of eigenvectors
q	$\in [0,1]^{d\times c}$	conditional probability of inputs
r	$\in \mathbb{R}^{+}$	time increment for reachable set computations
s	$\in \mathbb{R}$	path variable of a traffic participant
t	$\in \mathbb{R}^{+}$	time
t^{comp}	$\in \mathbb{R}^{+}$	computational time for an algorithm
t_f	$\in \mathbb{R}^{+}$	final time, time horizon
$\tilde{t}$	$\in \mathbb{R}^{+}$	estimated time
u	$\in \mathbb{R}^{m}$	input vector
$\tilde{u}$	$\in \mathbb{R}^{m}$	input vector that translates the input set $\mathcal{U}$
$\mathtt{u}$	$\in \mathbb{R}^{m}$	alternative input vector
$\hat{u}$	$\in \mathbb{R}^{m}$	alternative input vector for Markov Chain abstraction
u^*	$\in \mathbb{R}^{m}$	linearization point of the input u
$\mathtt{V}$	$\in \mathbb{R}^{n\times r}$	matrix of vertices
$\mathtt{v}$	$\in \mathbb{R}^{n}$	vertex of a polytope
v	$\in \mathbb{R}$	velocity of a traffic participant
v^{sw}	$\in \mathbb{R}^{+}$	velocity when the acceleration model is switched
W	$\in \mathbb{R}^{n\times n}$	weighting matrix
w	$\in \mathbb{R}^{n}$	diagonal vector of the weighting matrix W
x	$\in \mathbb{R}^{n}$	state vector
x^h	$\in \mathbb{R}^{n}$	state vector of the homogeneous solution
x^p	$\in \mathbb{R}^{n}$	state vector of the inhomogeneous solution
$\tilde{x}^p$	$\in \mathbb{R}^{n}$	state vector due to the constant input $\tilde{u}$
x^*	$\in \mathbb{R}^{n}$	linearization point of the state x
Y	$\in \mathbb{R}^{n}$	auxiliary matrix for an under-approximation of $e^{\mathcal{A}t}$

y	$\in \mathbb{N}^+$	discrete state of hybrid automaton
Z	$\in \mathbb{R}^n$	auxiliary matrix for an under-approximation of $e^{\mathcal{A}t}$
z	$\in \mathbb{R}^{n+m}$	combined state and input vector
z^*	$\in \mathbb{R}^{n+m}$	linearization point of the combined state and input vector z
d	$\in \mathbb{R}$	deviation variable of a traffic participant
ϵ	$\in [0,1]$	probability for driver inattentiveness
η	$\in [0,1]^{d\times c}$	constraint values
$\tilde{\eta}$	$\in [0,1]^{d\cdot c}$	constraint vector
Γ	$\in [0,1]^{c\times c\times d}$	input transition values
$\tilde{\Gamma}$	$\in [0,1]^{d\cdot c\times d\cdot c}$	input transition matrix
Λ	$\in \mathbb{R}^{n\times n}$	generator matrix for computing enclosing parallelotopes
λ	$\in [0,1]^{d\times c}$	priority values
$\hat{\lambda}$	$\in \mathbb{R}^n$	vector of eigenvalues
μ	$\in [0,1]^c$	motivation vector
Φ	$\in [0,1]^{d\times d\times c}$	state transition values
$\tilde{\Phi}$	$\in [0,1]^{d\cdot c\times d\cdot c}$	state transition matrix
Π	$\in \mathbb{R}^{n\times n}$	generator matrix for zonotope reduction
Ψ	$\in [0,1]^{c\times c}$	intrinsic input transition matrix
ρ	$\in \mathbb{R}^p$	parameter vector
$\hat{\rho}$	$\in \mathbb{R}^n$	direction vector of a vector field
ϱ	$\in \mathbb{R}^+$	performance measure for the splitting of zonotopes
$\hat{\varrho}$	$\in \mathbb{Q}^+$	order of a zonotope
σ^{cl}	$\in [0,1]$	motivation for driving on current lane
σ^{nl}	$\in [0,1]$	motivation for driving on neighboring lane
σ^{conv}	$\in [0,1]$	convenience of following driver
τ	$\in \mathbb{R}^+$	time increment of Markov chains
Θ	$\in [0,1]^{d\times d\times c\times c}$	interaction values
$\tilde{\Theta}$	$\in [0,1]^{d\cdot c\times d\cdot c}$	interaction matrix
θ	$\in \mathbb{R}^n$	expansion vector for the admissible linearization error
υ	$\in \mathbb{R}^+$	over-appr. measure for the order reduction of zonotopes
ξ	$\in \mathbb{R}^{n+m}$	auxiliary variable to determine linearization errors

Random Variables

$\mathbf{a}$	$\Omega \to \mathbb{R}$	random acceleration of a traffic participant
$\mathbf{N}$	$\Omega \to \mathbb{R}^n$	random vector with normal (Gaussian) distribution
$\mathbf{p}$	$\Omega \to \mathbb{R}^p$	random parameter
$\mathbf{t}$	$\Omega \to \mathbb{R}^+$	random time
$\mathbf{u}$	$\Omega \to \mathbb{R}^n$	random input variable
$\mathbf{W}$	$\Omega \to \mathbb{R}^m$	Wiener process
$\mathbf{X}$	$\Omega \to \mathbb{R}^n$	random variable of the spectral density
$\mathbf{x}$	$\Omega \to \mathbb{R}^n$	random state variable
$\mathbf{x}^h$	$\Omega \to \mathbb{R}^n$	random state variable of the homogeneous solution
$\mathbf{x}^p$	$\Omega \to \mathbb{R}^n$	random state variable of the inhomogeneous solution

$\mathbf{y}$	$\Omega \to \mathbb{N}^+$	random variable of the discrete input
$\mathbf{Z}$	$\Omega \to \mathbb{R}^n$	Gaussian zonotope
$\mathbf{z}$	$\Omega \to \mathbb{N}^+$	random variable of the discrete state
$\boldsymbol{\xi}$	$\Omega \to \mathbb{R}^m$	white noise

Enclosing Hull Variables

$\mathscr{H}$	enclosing hull variable of the homogeneous solution
$\mathscr{P}$	enclosing hull variable of the inhomogeneous solution
$\mathscr{R}$	enclosing hull variable of the state
$\tilde{\mathscr{R}}$	enclosing hull variable defined as $\mathscr{H} + \tilde{\mathcal{P}}$.
$\mathscr{X}^0$	enclosing hull variable of initial states
$\mathscr{Z}$	enclosing hull zonotope

Functions

erf()	$\mathbb{R} \to \mathbb{R}$	error function
f()	$\mathbb{N}^+ \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$	flow function of a hybrid automaton
$\bar{f}()$	$\mathbb{R}^n \to [0, 1]$	enclosing hull of a probability density function
g()	$\mathbb{N}^+ \times \mathbb{N}^+ \to 2^{\mathbb{R}^n}$	guard function of a hybrid automaton
h()	$\mathbb{R}^n \to \mathbb{R}^n$	jump function of a hybrid automaton
ind()	$\mathbb{R}^n \to \{0, 1\}$	indicator function
inv()	$\mathbb{N}^+ \to 2^{\mathbb{R}^n}$	invariant function of a hybrid automaton
rect()	$\mathbb{R} \to \{0, 1\}$	rectangular function
si()	$\mathbb{R} \to \mathbb{R}$	sinc function

Operators

box()	returns the enclosing axis-aligned box of a set
center()	returns the center of a set
CH()	returns the convex hull
conf()	returns the confidence set of a G- or EH-zonotope
cov	returns the covariance of a matrix
det()	returns the determinant of a matrix
diag()	returns the diagonal matrix given a vector
generatorConv()	returns over-approximative generator representation
halfspaceConv()	returns over-approximative halfspace representation
inputReset()	performs reset of an input probability vector
intersectionTimes()	returns over-appr. time interval for guard set intersection
max()	returns the maximum value

`min`()	returns the minimum value
`norm`()	normalizes the column sum of matrices to 1
`nX`()	returns the n-dimensional cross product
`reach`()	returns over-approximative continuous reachable set
`reduce`()	returns zonotope with reduced order
`sign`()	returns the sign of a scalar
`sup`()	returns the supremum
`tr`()	returns the trace of a matrix
`V`()	returns the volume of set
`Var`()	returns the variance of a random variable

Others

Constants

`const`	arbitrary constant value
g	gravity constant

Quantities

a	number of path segments
b	number of deviation segments
c	number of discrete inputs
d	number of locations ($\hat{=}$ discrete state values)
e	number of generators of a zonotope
m	number of continuous input variables
N^{crash}	number of Monte Carlo simulations resulting in a crash
N_s	number of Monte Carlo simulations
n	number of continuous state variables/ general number of dimensions
n^{sim}	number of simulation runs for the Markov chain abstraction
$\tilde{n}$	number of intermediate time steps for the Markov chain abstraction
$\boldsymbol{o}$	number of generators of a Gaussian zonotope
p	number of parameters
q	number of halfspaces of a polytope
r	number of vertices of a polytope
u	alternative number of generators of a zonotope
$\boldsymbol{u}$	alternative number of generators of a Gaussian zonotope
$\tilde{u}$	selected number of generators of a zonotope that are unreduced
η	number of terms for the Taylor expansion
ϑ	number of terms for the Taylor expansion up to numerical precision
κ	number of matrices specifying a matrix zonotope or matrix polytope
κ	selected number of generators for the zonotope reduction process
λ	number of best combinations of generators for zonotope reduction
ζ	number of best parallelotopes for zonotope conversion

Abstract

This thesis is about the safety verification of dynamical systems using reachability analysis. Novel solutions have been developed for three major problem classes: Classical reachability analysis, stochastic reachability analysis, and their application to the safety assessment of autonomous cars.

Classical reachability analysis aims to compute the exact or over-approximative set of states that can be reached by a system for a given set of initial states, inputs, and parameters. If the reachable set does not intersect any set of unsafe states, the safety of the system is guaranteed – this cannot be achieved by simulation techniques since only single solutions out of infinitely many can be checked. Specialized algorithms have been developed for linear systems, nonlinear systems, and hybrid systems.

The concept of reachability analysis is extended to stochastic reachability analysis which measures the probability of reaching an unsafe set. One pursued approach computes over-approximative enclosures of stochastic reachable sets for linear systems. Another developed approach generates a Markov chain which approximately reproduces arbitrary dynamics, allowing the computation of the stochastic reachable set on the simplified dynamics.

Finally, stochastic reachable sets are applied to the safety assessment of autonomous cars. Autonomous cars are driverless, i.e. they drive automated according to a planned trajectory which is computed based on detected roads, obstacles, and traffic participants. The safety assessment relies on a prediction of traffic scenes, where the stochastic reachable set of each relevant traffic participant is computed by the developed abstraction by Markov chains. This allows the crash probability of the autonomous car to be predicted for a planned trajectory.

Zusammenfassung

Die vorliegende Dissertation behandelt die Sicherheitsverifikation von dynamischen Systemen mittels Erreichbarkeitsanalyse. Neuartige Lösungen wurden für drei wesentliche Problemklassen erarbeitet: klassische Erreichbarkeitsanalyse, stochastische Erreichbarkeitsanalyse und deren Anwendung auf die Sicherheitsbewertung von autonomen Autos.

Klassische Erreichbarkeitsanalyse zielt darauf ab, die exakte oder überapproximierte Menge an Zuständen zu berechnen, die von einer gegebenen Menge an Anfangszuständen, Eingangswerten und Parametern erreicht werden kann. Falls die Erreichbarkeitsmenge keine Menge mit unsicheren Zuständen schneidet, ist die Sicherheit des Systems garantiert – dies kann allerdings nicht mit simulativen Methoden bewerkstelligt werden, da man damit immer nur einzelne Lösungen aus einer unendlichen Anzahl möglicher Lösungen überprüfen kann. Spezialisierte Algorithmen wurden für lineare Systeme, nichtlineare Systeme und hybride Systeme entwickelt.

Eine Erweiterung des Konzepts der Erreichbarkeitsanalyse stellt die stochastische Erreichbarkeitsanalyse dar, die abschätzt mit welcher Wahrscheinlichkeit eine unsichere Menge erreicht wird. Ein verfolgter Ansatz berechnet einen überapproximierten Einschluss von stochastischen Erreichbarkeitsmengen für lineare Systeme. Ein anderer entwickelter Ansatz erzeugt eine Markov Kette, die eine beliebige Dynamik näherungsweise wiedergibt, so dass die stochastische Erreichbarkeitsanalyse mit der vereinfachten Dynamik berechnet werden kann.

Zuletzt werden stochastische Erreichbarkeitsmengen zur Sicherheitsbewertung von autonomen Autos angewandt. Autonome Autos sind fahrerlos, d.h. sie fahren automatisiert mit Hilfe einer geplanten Trajektorie, die basierend auf detektierten Straßen, Hindernissen und Verkehrsteilnehmern berechnet wird. Die Sicherheitsbewertung beruht auf der Prädiktion von Verkehrsszenen, wobei die stochastischen Erreichbarkeitsmengen anderer relevanter Verkehrsteilnehmer mit der entwickelten Abstraktion durch Markov Ketten berechnet werden. Damit kann die Unfallwahrscheinlichkeit des autonomen Fahrzeugs für eine geplante Trajektorie prädiziert werden.

1. Introduction

One of the biggest boosts for innovation in engineering has been the ongoing improvement of digital processor technology. Connections between physical systems and computing elements are becoming more and more intense in many areas, such as transportation, energy, healthcare, manufacturing, chemical processes, and consumer appliances, to name only a few. Systems with strong interconnected physical and computing elements are referred to as embedded systems, mechatronic systems, or cyber-physical systems. The latter expression has recently become more popular when the interconnection between the computational and physical elements is strong.

Besides an improvement in e.g. the efficiency or functionality, cyber-physical systems provide a better reliability and safety compared to conventional solutions. Examples are driving assistant systems in passenger cars or search and rescue robots. The drawback of cyber-physical systems is that they are complex and thus difficult to analyze. This contradicts the objective of improving the safety of such systems, since the correct operation of these systems has to be ensured first. In order to cope with the growing complexity, one has to design improved analysis tools to better understand the behavior of a system or even verify its safety. In short, the demand for automatic verification tools that can analyze complex systems is constantly increasing.

Powerful verification methods already exist to prove the safe operation of complex discrete systems [41]. Starting from discrete automata, verification methods have been extended to timed automata which contain clocks, where each clock represents a continuous variable [7]. Verification methods have also been extended to hybrid automata which allow discrete dynamics with general continuous dynamics to be combined. Verification methods that have been developed in the hybrid systems community have also become popular tools for the analysis of purely continuous dynamics. Additionally, a large variety of verification techniques have been suggested for stochastic systems with continuous and hybrid dynamics.

Methods that allow the verification of deterministic and stochastic systems with continuous and hybrid dynamics are briefly surveyed below. Note that a more detailed survey is given at the beginning of each chapter.

1.1. Safety Verification

Before a system can be verified, one has to specify the properties to be checked. There exists a large number of safety properties for discrete systems, which can be extended to hybrid systems [41]. However, for continuous and hybrid systems, most research activities concentrate on the problem of proving that there exists no trajectory entering a set of forbidden or also called unsafe states. For instance, a set of unsafe states in a chemical plant may contain all temperatures above the boiling point of a liquid. The verification task would be to guarantee that it is impossible to reach the unsafe set containing temperatures above the boiling point. The challenge of guaranteeing the avoidance of unsafe sets lies in the infinitely many possible trajectories that a system can evolve with when the initial state, the input, or the parameters may take values within a continuous set.

An obvious technique to test the correct behavior of a system, is by simulation. The big advantage of a simulation is that it might produce a counter-example, i.e. a trajectory that hits a set of unsafe states. In this case, one can show that a system is unsafe. However, one cannot prove that the system is safe if no counter-example is produced, since there exist infinitely many possible trajectories due to uncertain initial states, inputs, and parameters. Thus, testing exemplary trajectories is not sufficient since the trajectory that hits the unsafe set may have been missed; see Fig. 1.1. A possibility to increase the chance of finding a counter-example is to strategically explore the state space. This has been done using Rapidly-Exploring Random Trees (RRTs) as shown in [30, 61] and robust test case generation in [90].

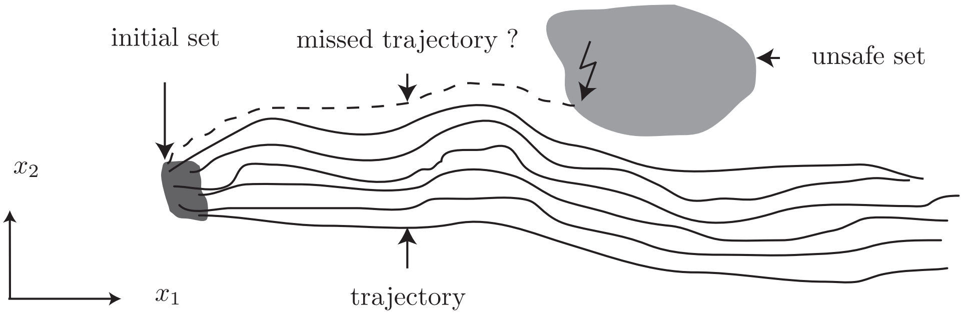


Fig. 1.1.: Searching for counter-examples by simulation.

Simulation techniques can be used for the safety verification of hybrid systems, if it can be guaranteed that sets of trajectories stay within certain regions around exemplary trajectories [56, 57, 73, 94]. This allows to cover all possible behaviors by a finite number of simulations as visualized in Fig. 1.2. A work that allows the safety of a hybrid system to be proven using RRTs can be found in [24].

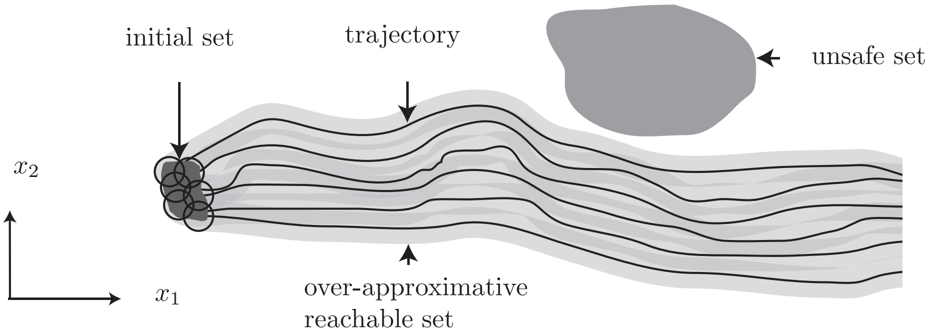


Fig. 1.2.: Verification using simulation.

A further possibility to verify that there exists no trajectory from a set of initial states to a set of unsafe states is the use of barrier certificates [137, 138]. This approach is based on the idea of finding a barrier which cannot be crossed by a system trajectory. If this barrier lies between the set of initial states and the set of unsafe states, it is proven that the system is safe, as depicted in Fig. 1.3. A barrier certificate is defined as a function which is similar to a Lyapunov function. Suppose there exists a barrier certificate which is negative in the initial set and positive in the set of unsafe states. In addition, the derivative of the barrier certificate is negative or zero along the system trajectories. Under these conditions, it is not possible that a system trajectory crosses the zero level set of the barrier certificate such that this level set serves as a barrier. The difficulty of this approach is finding a proper barrier certificate for a given problem.

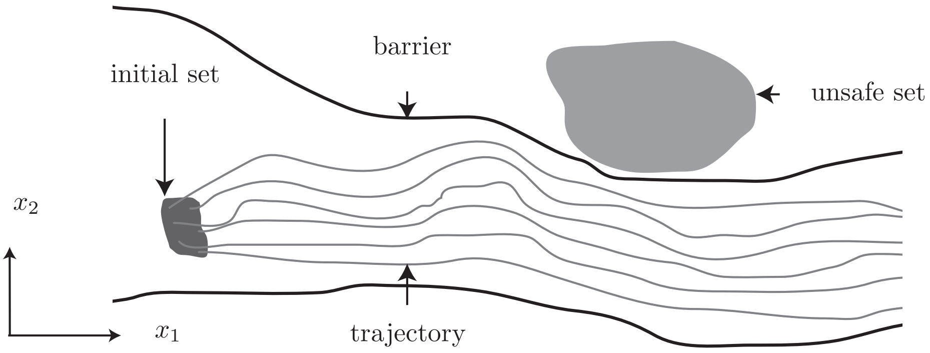


Fig. 1.3.: Verification using barrier certificates.

In this thesis, the safety verification is conducted via reachability analysis. Loosely speaking, reachability analysis determines the set of states that a system can possibly visit. A more precise description of a reachable set is the union of all possible trajectories that a system can evolve within finite or infinite time, when starting from a bounded set of initial states, subject to a set of possible input and parameter values. An example of a reachable set is presented in Fig. 1.4. If the reachable set does not intersect any set of unsafe states, one can guarantee the safety of the system.

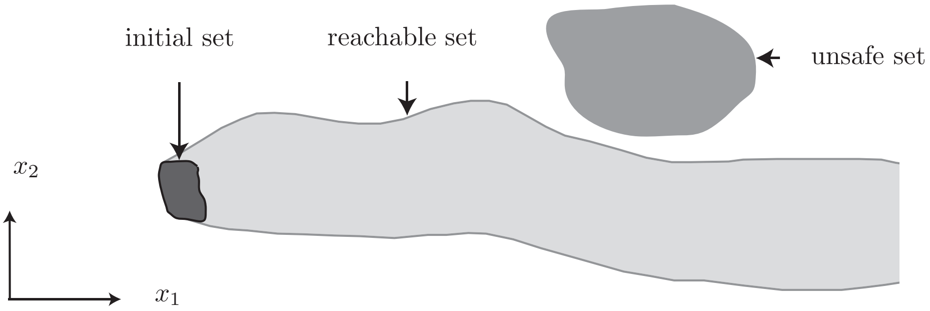


Fig. 1.4.: Verification using reachable sets.

However, one can only compute the exact reachable set for special cases [7, 81, 107]. A possibility to still prove the safety of a system is to over-approximate the set of reachable states, as shown in Fig. 1.5. Clearly, if the over-approximated set of reachable states does not intersect the set of unsafe states, the original system is safe, too. The downside is that if the over-approximation intersects the unsafe set, one cannot decide if the system is unsafe since the exact reachable set might not intersect the unsafe set. Thus, the goal is to minimize the over-approximation of reachable sets along with a moderate increase in computational costs. If this goal is accomplished, there is much hope that reachability analysis will become a tool that is frequently used by a huge variety of engineers – much the same as today's use of simulations. Besides over-approximative techniques, approximation techniques which compute reachable sets converging to the exact solution when refining the computation also exist; see e.g. [121, 163].

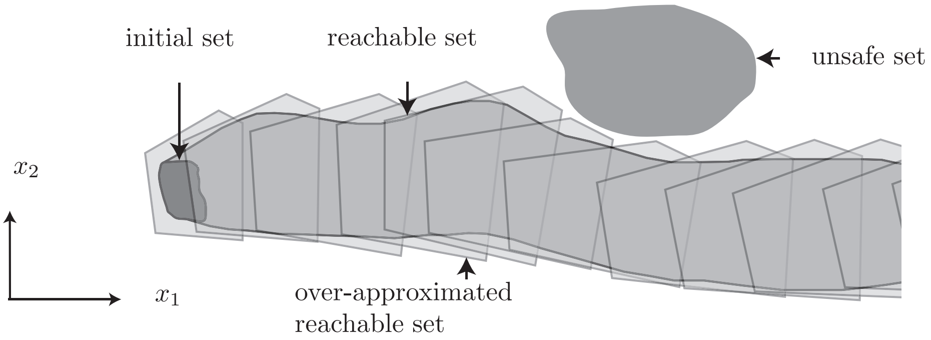


Fig. 1.5.: Verification using over-approximated reachable sets.

A different line of research, which is not based on state space exploration, is automated theorem proving as presented in e.g. [8, 135]. Thereto, special logics are developed that allow to specify correctness properties of hybrid systems which are verified using dedicated logical calculus.

Below, the concept of safety verification is extended to the concept of stochastic verification.

1.2. Stochastic Safety Verification

Often, it is not sufficient to know that a system is not safe. Especially for stochastic systems, it is important to know the probability that the system enters an unsafe state. Further, the probability of an unsafe operation allows the search for control strategies that minimize the risk of an accident or failure. Of course, this is not possible if the safety verification always returns that the system is unsafe without any additional probabilistic information. For instance, one has to use stochastic safety verification methods for the safety analysis of automated road vehicles, since road traffic with human drivers is inherently unsafe, as shown later.

The most prominent method for the safety analysis of stochastic systems is Monte Carlo simulation [147]. The term Monte Carlo simulation or Monte Carlo method refers to methods that are based on random sampling and numerical simulation. Monte Carlo methods are especially popular for complex and highly coupled problems where other probabilistic approaches fail. When applying Monte Carlo simulation in the context of safety verification, initial states and input trajectories are randomly created according to specified probability distributions and simulated as shown in Fig. 1.1. The number and weight of the trajectories hitting unsafe sets determines the probability that the system is unsafe.

Besides the simulative analysis of stochastic systems, the previously introduced concept of barrier certificates can also be extended to stochastic systems using supermartingales

[139]. The special property of stochastic barrier certificates is that an upper bound for the probability of entering an unsafe set can be computed. Most other approaches, such as Monte Carlo simulation, can only approximate this probability.

The concept of reachability analysis can also be extended to a stochastic setting by additionally computing the probability distribution within the reachable set as indicated in Fig. 1.6. For some special classes, such as linear continuous systems with Gaussian white noise, the probability distribution can be computed exactly. However, for most other systems, one has to approximate or over-approximate the probability of hitting an unsafe set.

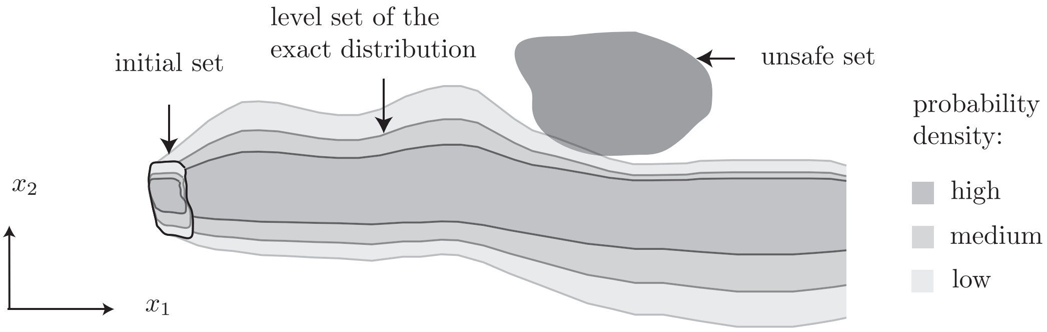


Fig. 1.6.: Verification using stochastic reachable sets.

In this thesis, two approaches are presented for computing stochastic reachable sets. The first concept over-approximates the probability of hitting a set of unsafe states for linear continuous systems. This is achieved by computing a function enclosing the exact probability distribution of the system, i.e. the probability values of the exact distribution are always equal or lower than the values of the enclosing function. Such a function is later called enclosing hull and is indicated in Fig. 1.7 by pushing the level sets of the exact distribution outwards.

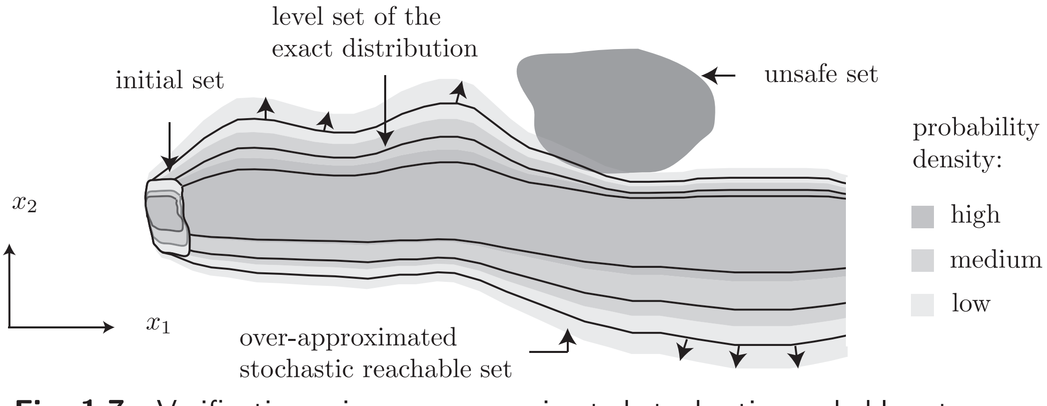


Fig. 1.7.: Verification using over-approximated stochastic reachable sets.

The other method presented in this thesis can be applied to general hybrid systems. The basic idea of this approach is to abstract the original hybrid dynamics to a Markov chain, which is a discrete stochastic system. The computation of the stochastic reachable set is then performed by the Markov chain via matrix multiplications, which is much easier to compute than the stochastic reachable set of the original dynamics. The abstraction to Markov chains is accomplished by discretization of the continuous state space, resulting in state space regions which are defined as the discrete states of the Markov chain. The drawback of this method is that the number of discrete states grows exponentially with the number of continuous state variables. Thus, this method can only be applied to low dimensional systems.

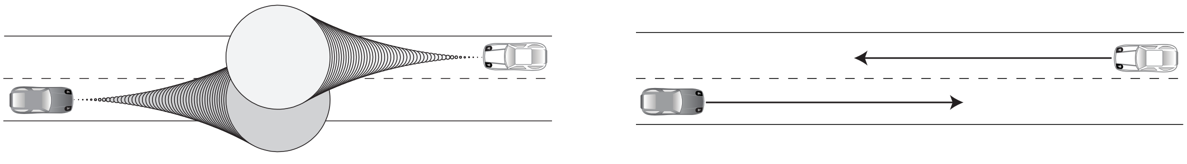
The concept of stochastic verification is applied to autonomous cars as described next.

1.3. Safety Assessment of Autonomous Cars

An important and representative application scenario for stochastic verification methods is the safety assessment of autonomous road vehicles, i.e. vehicles that drive without a human driver. One of the hopes for the development of autonomous vehicles is to drastically reduce the number of accidents. Worldwide, the number of people killed in road traffic each year is estimated at almost 1.2 million, while the number of injured is estimated at 50 million [134, chap. 1]. The developed methods for the safety assessment of autonomous vehicles aim at improving these statistics and may also contribute to the improvement of future driving assistance systems.

Due to the inherently unsafe nature of road traffic, deterministic verification methods are not applied. This is obvious as traffic participants have the possibility to crash into another vehicle on purpose, which would classify a situation as unsafe. However, since most traffic participants are cooperative, a pure deterministic analysis would be too conservative. Additionally, it is important to modify the planned action of the autonomous vehicle based on the current threat level, which can only be provided by stochastic methods.

A reasonable application of deterministic safety verification is given when all traffic participants are automated and broadcast their planned trajectories to other traffic participants. In such a scenario, each vehicle would know the planned trajectories of other vehicles and could check deterministically if a crash is possible. This is illustrated by a simple example, where two cars pass each other on a straight road as shown in Fig. 1.8. In Fig. 1.8(a), the trajectory of the other vehicle is unknown, such that the physically possible set of the centers of gravity of both vehicles has to be computed according to [149]. Because the obtained sets intersect, this everyday situation has to be classified as unsafe. When the planned trajectories of both vehicles are known, the situation can be classified as safe; see Fig. 1.8(b). Thus, deterministic methods are only suitable in situations in which the plans of other vehicles are known.



(a) Unknown behavior.

(b) Known behavior.

Fig. 1.8.: Reachable sets of the centers of gravity of two passing cars.

In general, the surrounding traffic situation of autonomous vehicles is sensed by environment sensors, such as optical cameras, laser scanners, and infrared cameras. The measured raw data allow to probabilistically estimate the initial states of the detected traffic participants. Based on the uncertain initial states, the stochastic reachable set of each traffic participant is computed for a predefined time horizon. The computations are based on the previously mentioned Markov chain abstraction. The offline generated Markov chains are applied to the online computation of the stochastic reachable sets. Ultimately, the crash probability of the autonomous vehicle is approximated based on the stochastic reachable sets of other traffic participants and the planned trajectory of the autonomous vehicle. This is illustrated in Fig. 1.9, where the autonomous car has to pass another car before turning left. In this situation, the autonomous car might cause a crash since the stochastic reachable sets intersect – the risk would be minimized by reducing the velocity of the autonomous vehicle.

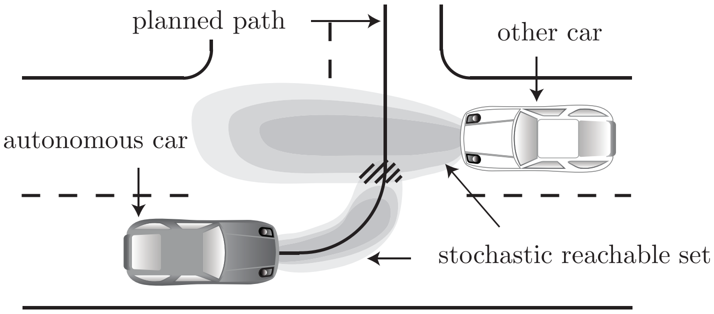


Fig. 1.9.: Safety assessment of a traffic scene using stochastic reachable sets.

The developed framework for the safety assessment of autonomous vehicles can in principle be applied to other scenarios than road traffic, too. Examples are the safety analysis of air traffic or mobile robots. For the autonomous robot ACE (Autonomous City Explorer), the prediction of humans was computed by the Markov chain abstraction developed in this work [195].

1.4. Outline of the Thesis

This thesis covers three major aspects of the verification of continuous and hybrid systems: Reachability analysis, stochastic reachability analysis, and the application of stochastic reachability analysis to autonomous vehicles.

In contrast to discrete systems, the set of reachable states in continuous spaces is not countable, which requires a proper set representation to be found. One of the challenges here, is the curse of dimensionality. For instance, the number of vertices required to represent a cube in n dimensions is 2^n, where a cube is one of the simplest geometries. In order to efficiently represent continuous sets in high dimensional spaces, zonotopes are used since they scale well with the dimension of the state space. In order to accelerate or allow certain operations on zonotopes, their enclosure by simpler representations is also presented. Different set representations, their transformation to other representations, their over-approximation, and operations on them are discussed in detail in Chap. 2. The results of this chapter are fundamental to reachability analysis in Chap. 3 and to enclosing probabilistic hulls, which are introduced in Chap. 4.

In Chap. 3, the efficient computation of over-approximations of reachable sets for different classes of dynamic systems with uncertain inputs is presented. First, existing algorithms for reachability analysis of linear systems are presented. The existing methods are extended to linear systems with uncertain, but constant parameters. A further extension is introduced in order to compute reachable sets for nonlinear systems. The procedure is based on linearization so that algorithms for linear systems can be used as underlying algorithms. The over-approximative computation is ensured by adding the set of possible linearization errors as an additional uncertain input. The approach for linear and nonlinear continuous systems scales well with the number of continuous state variables: Examples with up to 100 continuous state variables have been computed. Finally, the extension to hybrid systems is presented, which considers the switching of the continuous dynamics and possible jumps in the continuous states.

Stochastic reachability analysis is discussed in Chap. 4. Two approaches are presented. One approach computes enclosing probabilistic hulls of linear systems. An enclosing probabilistic hull is later defined as a function that encloses the exact probability distribution, i.e. the values of the enclosing probabilistic hull are greater than the values of the exact probability distribution. Thus, enclosing probabilistic hulls are no longer probability distributions since their integral is greater than one. This approach allows the probability of hitting an unsafe set to be over-approximated and can be applied to high dimensional linear systems. The other approach is based on the abstraction of continuous or hybrid systems to Markov chains. This simplifies the computation of stochastic reachable sets to matrix multiplications. The abstraction to Markov chains is based on a discretization of the continuous state space, where each state space region is represented by a discrete state. The probabilities of entering certain state space regions starting from certain state space regions are stored as transition probabilities – these probabilities uniquely specify the Markov chain. In contrast to the approach computing with enclosing hulls, the Markov chain abstraction can also be applied to nonlinear and hybrid systems. The drawback is that due to the discretization of the continuous state space, only systems with a few (up to $3-5$) continuous state variables can be handled.

The Markov chain abstraction is applied to the safety analysis of autonomous vehicles in Chap. 5. The stochastic reachable sets of each traffic participant are computed for a specified time horizon. A major requirement is that the prediction is computationally efficient, since it has to be executed during the operation of the autonomous vehicle. This is ensured by the Markov chain abstraction, because most computations can be performed offline when generating the Markov chain. The execution of the Markov chains is efficient since only matrix multiplications are performed. One of the challenges in the prediction of other traffic participants is the stochastic modeling of their driving behavior. The proposed algorithms consider physical constraints and constraints due to the interactions with other traffic participants. Another focus is on the efficient computation of crash probabilities given the stochastic reachable sets. In order to increase the confidence in the results obtained from Markov chains, the results have been compared in terms of accuracy and computational efficiency to the ones obtained from Monte Carlo simulation. The safety assessment of autonomous vehicles is concluded by a description of a driving experiment conducted with the experimental vehicle *MUCCI*.

The thesis concludes with possible future research directions in the field of (stochastic) reachability analysis and its application to autonomous vehicles in Chap. 6.

2. Set Representations and Set Operations

One of the biggest challenges in reachability analysis is the curse of dimensionality. It is crucial to use a representation of reachable sets that scales well with the dimension of the state space. Since this thesis deals with the over-approximative computation of reachable sets, it is possible to over-approximate complicated set representations by a simpler one in order to save computational time. This can be done by either enclosing a set by a different representation or by the same representation but with fewer parameters. Another important property of the set representation to be chosen is that the most important operations can be computed efficiently. Given two general sets $\mathcal{V}_1, \mathcal{V}_2 \subset \mathbb{R}^n$, the most important operations for reachability analysis are:

- Linear transformation: $A \cdot \mathcal{V}_1 = \{A \cdot s_1 | s_1 \in \mathcal{V}_1\}$, $A \in \mathbb{R}^{n\times n}$.
- Minkowski sum[1]: $\mathcal{V}_1 + \mathcal{V}_2 = \{s_1 + s_2 | s_1 \in \mathcal{V}_1, s_2 \in \mathcal{V}_2\}$.
- Convex hull: $\mathtt{CH}(\mathcal{V}_1, \mathcal{V}_2) = \{\alpha_1 \cdot s_1 + \alpha_2 \cdot s_2 | s_1 \in \mathcal{V}_1, s_2 \in \mathcal{V}_2, \alpha_{1,2} \geq 0, \alpha_1 + \alpha_2 = 1\}$.
- Intersection: $\mathcal{V}_1 \cap \mathcal{V}_2 = \{s | s \in \mathcal{V}_1, s \in \mathcal{V}_2\}$.

In this thesis, zonotopes are used as a representation of reachable sets due to the efficient computation of linear transformations and Minkowski sums. In general, it is not possible to represent the convex hull of two zonotopes by a zonotope. However, there exists a simple algorithm that over-approximates the convex hull by a zonotope. It is also not possible to generally represent the intersection of two zonotopes by a zonotope. For this reason, zonotopes have to be converted to a halfspace representation which allows the intersection of two zonotopes to be represented by a polytope. Another auxiliary representation that is used are multidimensional intervals.

This chapter is organized as follows: In Sec. 2.1, the representations of polytopes, zonotopes and multidimensional intervals are introduced. Next, it is shown in Sec. 2.2 how to exactly convert less general representations to more general ones. The contrary problem of over-approximating more general representations by less general ones is addressed in Sec. 2.3. The required operations on zonotopes (linear transformation, Minkowski sum, convex hull) are presented in Sec. 2.4. In addition, several algorithms are presented in Sec. 2.5 that allow the number of parameters for a zonotope to be reduced while causing only a small over-approximation. Finally, interval arithmetics is briefly introduced in Sec. 2.6 which allows multidimensional intervals to be computed with. Many results and aspects of this chapter have been published by the author in [187, 191].

[1] In other works, the Minkowski sum is often denoted by $\oplus$.

2.1. Set Representations

The most general sets considered in this thesis are convex polytopes, for which two representations exist: The halfspace representation (H-representation) and the vertex representation (V-representation). The halfspace representation specifies a convex polytope $\mathcal{P}$ by the intersection of q halfspaces $\mathcal{H}^{(i)}$: $\mathcal{P} = \mathcal{H}^{(1)} \cap \mathcal{H}^{(i)} \cap \ldots \cap \mathcal{H}^{(q)}$. A halfspace is either one of the two parts that is obtained by dividing the n-dimensional Euclidian space with a hyperplane $\mathcal{S}$, which is given by $\mathcal{S} := \{x|c \cdot x = d\}, c \in \mathbb{R}^{1\times n}, d \in \mathbb{R}$. The vector c is the normal vector of the hyperplane and d the scalar product of any point on the hyperplane with the normal vector. From this follows that the corresponding halfspace is $\mathcal{H} := \{x|c \cdot x \leq d\}$. As the convex polytope $\mathcal{P}$ is the nonempty intersection of q halfspaces, q inequalities have to be fulfilled simultaneously:

Definition 2.1 (H-Representation of a Polytope): For q halfspaces, a convex polytope $\mathcal{P}$ is the set

$$\mathcal{P} = \Big\{x \in \mathbb{R}^n \Big| C \cdot x \leq d, \quad C \in \mathbb{R}^{q\times n}, d \in \mathbb{R}^{q\times 1}\Big\}. \qquad \square$$

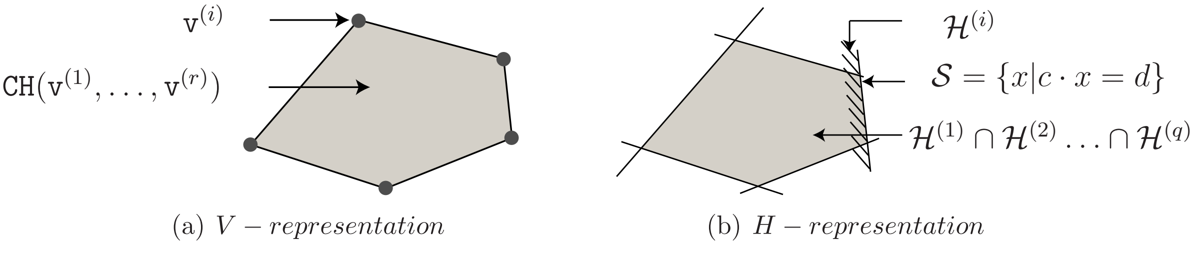
A polytope with vertex representation is defined as the convex hull of a finite set of points in the n-dimensional Euclidian space. The points are also referred to as vertices and are denoted by $\mathtt{v}^{(i)} \in \mathbb{R}^n$. A convex hull of a finite set of r points is obtained from their linear combination:

$$\mathtt{CH}(\mathtt{v}^{(1)}, \ldots, \mathtt{v}^{(r)}) := \Big\{ \sum_{i=1}^{r} \alpha_i \mathtt{v}^{(i)} \Big| \mathtt{v}^{(i)} \in \mathbb{R}^n,\ \alpha_i \in \mathbb{R},\ \alpha_i \geq 0,\ \sum_{i=1}^{r} \alpha_i = 1 \Big\}.$$

Given the convex hull operator $\mathtt{CH}()$, a convex and bounded polytope can be defined in vertex representation as follows:

Definition 2.2 (V-Representation of a Polytope): For r vertices $\mathtt{v}^{(i)} \in \mathbb{R}^n$, a convex polytope $\mathcal{P}$ is the set $\mathcal{P} = \mathtt{CH}(\mathtt{v}^{(1)}, \ldots, \mathtt{v}^{(r)})$. □

The halfspace and the vertex representation are illustrated in Fig. 2.1. Algorithms that convert from H- to V-representation and vice versa are presented in [91].


$\mathtt{v}^{(i)}$
$\mathtt{CH}(\mathtt{v}^{(1)}, \ldots, \mathtt{v}^{(r)})$
$\mathcal{H}^{(i)}$
$\mathcal{S} = \{x|c \cdot x = d\}$
$\mathcal{H}^{(1)} \cap \mathcal{H}^{(2)} \ldots \cap \mathcal{H}^{(q)}$

(a) $V - representation$ (b) $H - representation$

Fig. 2.1.: Possible representations of a polytope.

Next, zonotopes are introduced which are predominantly used for reachable set computations in this work. Since zonotopes are a special case of polytopes, they can also be represented by the halfspace or the vertex representation. A further possibility for the representation of zonotopes is the use of so-called generators (G-representation). After introducing the center of a zonotope as $c \in \mathbb{R}^n$ and the e generators of a zonotope as $g^{(i)} \in \mathbb{R}^n$, $i = 1, \dots, e$, the generator representation can be defined as (see e.g. [176]):

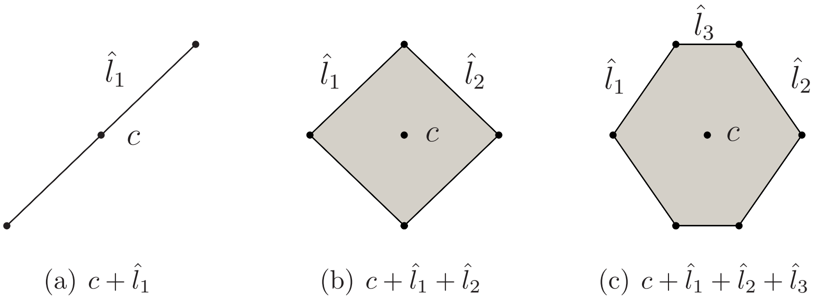
Definition 2.3 (G-Representation of a Zonotope): A zonotope is a set

$$\mathcal{Z} = \Big\{ x \in \mathbb{R}^n \Big| x = c + \sum_{i=1}^{e} \beta_i \cdot g^{(i)}, \quad -1 \leq \beta_i \leq 1 \Big\}$$

with $c, g^{(1)}, \dots, g^{(e)} \in \mathbb{R}^n$. □

The vector c is the center of the zonotope, to which the latter is centrally symmetric. The definition can be interpreted as the Minkowski sum of a finite set of line segments $\hat{l}_i = [-1, 1] \cdot g^{(i)}$ which illustrates how a zonotope is built step-by-step. This is shown in Fig. 2.2, where from left to right further two-dimensional generators are added. Another definition is the linear map of an e-dimensional hypercube.

The order of a zonotope is defined as $\hat{\varrho} = \frac{e}{n}$. If the order is less than one, the zonotope represents a set of lower dimension than n (see Fig. 2.2(a)) and a zonotope of order one is a parallelotope (see Fig. 2.2(b)). Zonotopes of order greater than one create sets with an increasing number of facets and vertices (see Fig. 2.2(c)). In order to receive a concise notation, the center and generators of a zonotope are written in short form as $\mathcal{Z} = (c, g^{(1)}, \dots, g^{(e)})$.



(a) $c + \hat{l}_1$ (b) $c + \hat{l}_1 + \hat{l}_2$ (c) $c + \hat{l}_1 + \hat{l}_2 + \hat{l}_3$

Fig. 2.2.: Construction of a zonotope.

Another type of sets that are frequently used in this thesis are multidimensional intervals $\mathcal{I}$, which are also called interval-hulls or hyperrectangles in the literature. Multidimensional intervals are a special case of zonotopes and, thus, they can be described in H-,V-, and G-representation. A further possibility is the interval representation (I-representation):

Definition 2.4 (I-Representation of a Multidimensional Interval):
A multidimensional interval is a set:

$$\mathcal{I} := [\underline{a}, \overline{a}], \quad \underline{a} \in \mathbb{R}^n, \, \overline{a} \in \mathbb{R}^n, \, \underline{a} \leq \overline{a}.$$ □

Another useful definition is the set of real valued intervals: $\mathcal{I} = \{[\underline{d}, \overline{d}] \,|\, \underline{d} \in \mathbb{R}, \, \overline{d} \in \mathbb{R}, \, \underline{d} \leq \overline{d}\}$, such that $\mathcal{I} \in \mathcal{I}^n$. Next, it is shown how less general representations of continuous sets can be converted to more general ones.

2.2. Exact Conversion of Set Representations

This subsection deals with the conversion of the H-,V-,G-, and I-representation. As the representations are not equally rich, exact conversions of representations are only possible in the following directions: I → G → H ↔ V . Conversions that are not possible in an exact way are performed in an over-approximative way in this thesis, i.e. the newly obtained set representations enclose the previous ones.

The problem of finding a V-representation of a polytope given in H-representation is well known as the *vertex enumeration problem* and the inverse problem of finding a V-representation given the H-representation is known as the *facet enumeration problem*. Both transformation directions are well-studied and there exist algorithms that are polynomial with respect to time [91].

2.2.1. Conversion of Zonotopes

A much less studied problem is finding a V- or H-representation for a G-representation. This problem is strongly related to the problem of computing the Minkowski sum of two polytopes. This is because a G-representation can be represented as the Minkowski sum of line segments and each line segment can be represented in V- or H-representation. It is much easier to compute the Minkowski sum for a V-representation than for a H-representation; see [170, chap. 8]. For the V-representation, there exists an algorithm with polynomial time with respect to the input and output size [66]. An algorithm that directly computes the H-representation without use of the Minkowski addition is proposed by the author in [191]. This algorithm is linear in the number of facets and can be implemented in a simple way, as shown below. However, the problem is that a zonotope with e generators in dimension n might have up to $2\binom{e}{n-1}$ facets, which will become clear later.

The proposed algorithm makes use of the n-dimensional cross-product, which computes the vector that is orthogonal to $n-1$ linearly independent vectors. These vectors are stored in the matrix $H = [h^{(1)}, \dots, h^{(n-1)}] \in \mathbb{R}^{n\times(n-1)}$. For convenience, the matrix $H^{[i]} \in \mathbb{R}^{(n-1)\times(n-1)}$, which is the H matrix where the i-th row is removed, is introduced. The cross product operator $\mathtt{nX}()$ can then be defined as (see e.g. [124]):

$$y = \mathtt{nX}(H) := \big[\dots,\ \ (-1)^{i+1}\det(H^{[i]}),\ \ \dots\big]^T .$$

Further, the matrix of generators $G = [g^{(1)}, \dots, g^{(n)}]$ and $G^{\langle i\rangle} \in \mathbb{R}^{n\times(n-1)}$, which is defined as the matrix G in which the i-th generator is removed, are introduced[2].

For an intuitive understanding of the algorithm transforming the G-representation of a zonotope into H-representation, the transformation is performed for the special case of a parallelotope (zonotope of order 1) first.

Lemma 2.1 (Halfspace Conversion of Parallelotopes): The halfspace representation $C \cdot x \le d$ of a parallelotope $\mathcal{Z} = (c, g^{(1)}, \dots, g^{(n)})$ with n independent generators

[2] Here, the removal is denoted by $\langle\,\rangle$ and not by $[\,]$ because a column instead of a row is removed.

is

$$C = \begin{bmatrix} C^+ \\ -C^+ \end{bmatrix}, \quad d = \begin{bmatrix} d^+ \\ d^- \end{bmatrix}$$

with:

$$C_i^+ = \mathtt{nX}(G^{\langle i \rangle})^T / \|\mathtt{nX}(G^{\langle i \rangle})\|_2,$$
$$d_i^+ = C_i^+ \cdot c + \Delta d_i, \quad d_i^- = -C_i^+ \cdot c + \Delta d_i, \quad \Delta d_i = |C_i^+ \cdot g^{(i)}|.$$

C_i^+ denotes the i-th row of C^+ and d_i the i-th element of d. The computational complexity of computing the H-representation for a given G is $\mathcal{O}(n^5)$. □

Proof: The i-th separating hyperplane $\mathcal{S}^{(i)}$ of a parallelotope can be reached from the center c by translation of the generator $g^{(i)}$: $c + g^{(i)} \in \mathcal{S}^{(i)}$ (see Fig. 2.3). As there are only n generators, the facet must be spanned by the matrix of remaining generators $G^{\langle i \rangle}$. Thus, the normal vectors are computed as $C_i^+ = \mathtt{nX}(G^{\langle i \rangle})^T / \|\mathtt{nX}(G^{\langle i \rangle})\|_2$. It is sufficient to compute n halfspaces denoted by a superscript '+', as the remaining n halfspaces denoted by a superscript '$-$' differ only in sign due to the central symmetry of zonotopes.

The elements d_i^+ are the scalar products of any point on the i-th halfspace with its normal vector C_i^+. A possible point on the i-th halfspace is $c + g^{(i)}$. Thus, the values of d_i^+ can be computed as $d_i^+ = C_i^+ \cdot c + \Delta d_i$ and $d_i^- = -C_i^+ \cdot c + \Delta d_i$ with $\Delta d_i = |C_i^+ \cdot g^{(i)}|$.

The computational complexity is derived as follows. The computation of the determinants is $\mathcal{O}((n-1)^3) = \mathcal{O}(n^3)$ when applying LU decomposition, and thus the computation of the n-dimensional cross product is $n \cdot \mathcal{O}(n^3) = \mathcal{O}(n^4)$. As n non-parallel hyperplanes have to be computed, the overall complexity is $\mathcal{O}(n^5)$. □

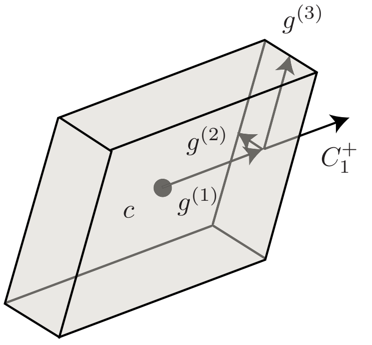


Fig. 2.3.: Generators and normal vector of a parallelotope.

The extension for the conversion of zonotopes from G- to H- representation is straightforward. For a general zonotope, the generator matrix is of dimension $G \in \mathbb{R}^{n \times e}$. Because $n-1$ generators have to be selected from e generators for each non-parallel facet, the result are $2\binom{e}{n-1}$ facets if the generators in G are linearly independent, i.e. G has full rank. The generators that span a facet are obtained by canceling $e-n+1$ generators from the G-matrix. This is denoted by $G^{\langle \gamma, \dots, \eta \rangle}$, where $\gamma, \dots, \eta$ are the $e-n+1$ indices of the generators that are taken out of G. The extended computation for the halfspace representation is:

Theorem 2.1 (Halfspace Conversion of Zonotopes): The halfspace representation $C \cdot x \leq d$ of a zonotope $\mathcal{Z} = (c, g^{(1)}, \dots, g^{(e)})$ with e independent generators is

$$C = \begin{bmatrix} C^+ \\ -C^+ \end{bmatrix}, \quad d = \begin{bmatrix} d^+ \\ d^- \end{bmatrix}$$

with:

$$C_i^+ = \mathtt{nX}(G^{\langle\gamma,\dots,\eta\rangle})^T / \|\mathtt{nX}(G^{\langle\gamma,\dots,\eta\rangle})\|_2,$$

$$d_i^+ = C_i^+ \cdot c + \Delta d_i, \quad d_i^- = -C_i^+ \cdot c + \Delta d_i, \quad \Delta d_i = \sum_{\upsilon=1}^{e} |C_i^+ \cdot g^{(\upsilon)}|.$$

The index i varies from 1 to $\binom{e}{n-1}$ and the indices $\gamma, \dots, \eta$ are obtained by picking $n-1$ out of e elements. The complexity of the computation with respect to the number e of generators is $\mathcal{O}(\binom{e}{n-1} \cdot e)$, which is linear in the number of facets. □

Proof: The computation of C_i^+, d_i^+, d_i^- is analog to lemma 2.1. The difference for the computation of Δd_i is that e generators contribute to pushing the facets outwards.

The complexity result is obtained as follows: The computational complexity for computing the normal vectors is $\mathcal{O}(\binom{e}{n-1} \cdot n^4)$ since the complexity for a single facet is $\mathcal{O}(n^4)$ (see lemma 2.1). The computational complexity for the computation of Δd is $\mathcal{O}(\binom{e}{n-1} \cdot e)$ as e generators have to be considered for each element Δd_i. The overall complexity with respect to the number of generators is $\mathcal{O}(\binom{e}{n-1} \cdot n^4) + \mathcal{O}(\binom{e}{n-1} \cdot e) = \mathcal{O}(\binom{e}{n-1} \cdot e)$. □

It is noted that the author does not know any reference where a similar algorithm has been proposed.

2.2.2. Conversion of Multidimensional Intervals

In the course of this thesis, it is useful to transform multidimensional intervals from I- into G-representation. In contrast to the previous transformation from G- into H-representation, the rewriting of multidimensional intervals into G-representation is straightforward.

Proposition 2.1 (Generator Representation of Multidimensional Intervals): The generator representation $(c, g^{(1)}, \dots, g^{(n)})$ of a multidimensional interval $I = [\underline{a}, \overline{a}]$ is

$$c = \frac{1}{2}(\underline{a} + \overline{a}), \quad g_i^{(j)} = \begin{cases} \frac{1}{2}(\overline{a}_i - \underline{a}_i), \text{ if } i = j \\ 0, \text{ otherwise} \end{cases},$$

where $g_i^{(j)}$ is the i-th element of the j-th generator. Note that the matrix of generators G is diagonal. □

Due to the obvious computation, the proof is skipped.

2.3. Over-Approximative Conversion of Set Representations

As mentioned in the previous subsection, there is no G-representation for polytopes and no I-representation for zonotopes in general. However, those conversions are necessary in reachability analysis, requiring that those transformations are performed in an over-approximative way. The simplest over-approximation is the enclosure of zonotopes by multidimensional intervals:

Proposition 2.2 (Interval-Enclosure of Zonotopes):
A zonotope $\mathcal{Z} = (c, g^{(1)}, \dots, g^{(e)})$ is over-approximated by a multidimensional interval by:

$$I = \texttt{box}(\mathcal{Z}) := [c - \Delta g, c + \Delta g], \quad \Delta g = \sum_{i=1}^{e} |g^{(i)}|,$$

where the absolute value is taken elementwise and $\texttt{box}()$ denotes the operator returning a multidimensional interval. □

Proof: As multidimensional intervals are axis-aligned sets, one can compute an over-approximation by considering each dimension separately. As each generator is multiplied by a factor $\beta_i \in [-1, 1]$, the possible interval of the j-th coordinate of a zonotope $\mathcal{Z} = (c, g^{(1)}, \dots, g^{(e)})$ is: $[c_j - \Delta g_j, c_j + \Delta g_j]$ and $\Delta g_j = \sum_{i=1}^{e} |g_j^{(i)}|$. This can be directly extended to the n-dimensional case. □

Similarly, the $\texttt{box}()$-operator can be formulated for a polytope:

Proposition 2.3 (Interval-Enclosure of Polytopes):
A polytope $\mathcal{P} = \texttt{CH}(\texttt{v}^{(1)}, \dots, \texttt{v}^{(r)})$ with vertices $\texttt{v}^{(1)}, \dots, \texttt{v}^{(r)}$ is over-approximated by a multidimensional interval as:

$$I = \texttt{box}(\mathcal{P}) := [\underline{m}, \overline{m}], \quad \underline{m}_j = \min(\texttt{v}_j^{(1)}, \dots, \texttt{v}_j^{(r)}), \, \overline{m}_j = \max(\texttt{v}_j^{(1)}, \dots, \texttt{v}_j^{(r)}). \qquad □$$

The proof is omitted, as it is obvious that the minimum and maximum values of the vertices of each coordinate return the intervals of the corresponding coordinate.

The problem of finding a zonotope enclosing a general polytope is more complicated. Even the problem of finding an optimal parallelotope enclosing a polytope is complicated and unsolved. On that account, polytopes are enclosed by parallelotopes (zonotopes of order one) in this work. There is much literature on finding enclosing parallelotopes of polytopes (or sets of points) in two-dimensional and three-dimensional space [17, 133, 169]. In n dimensions, there is no known work which computes a parallelotope with minimum volume. However, enclosing zonotopes which are optimal in the sum of the total length of the generators are presented in [79]. In [158], enclosing parallelotopes in n dimensions have been computed based on a principal component analysis (PCA) of the point set to be enclosed. The problem of finding an enclosing parallelotope of a set of points can be reformulated using a linear transformation and the $\texttt{box}()$-operator:

Proposition 2.4 (Parallelotope-Enclosure of Polytopes): An over-approximating parallelotope Ψ of a polytope $\mathcal{P}$ is obtained as:

$$\Psi = \Lambda \cdot \mathtt{box}(\Lambda^{-1}\mathcal{P}),$$

where $\Lambda \in \mathbb{R}^{n\times n}$ is of full rank. □

Proof: First, the coordinates of $\mathcal{P}$ are transformed by the linear map Λ^{-1}, where the new coordinate axes are the column vectors within Λ. Note that this coordinate system is not orthogonal in general. Within the transformed coordinate system, the zonotope is over-approximated by a multidimensional interval ($\mathtt{box}(\Lambda^{-1} \cdot \mathcal{P})$). As a final step, the multidimensional interval is transformed back to the original coordinate system, which returns a parallelotope. The over-approximation is guaranteed by the fact that the parallelotope is over-approximated in the transformed coordinate system by the $\mathtt{box}()$ operator, such that it is also over-approximated after the transformation to the original coordinate system. This is also illustrated in Fig. 2.4. □

The challenge is to find a linear transformation matrix Λ, which over-approximates the polytopes in a good way. Note that the column vectors of Λ determine the direction of the generators of the over-approximating parallelotope. The choice of the Λ matrix, and hence the selection of generators for the over-approximating parallelotope, is addressed later in Sec. 3.5.4 when dealing with reachable sets of hybrid systems.

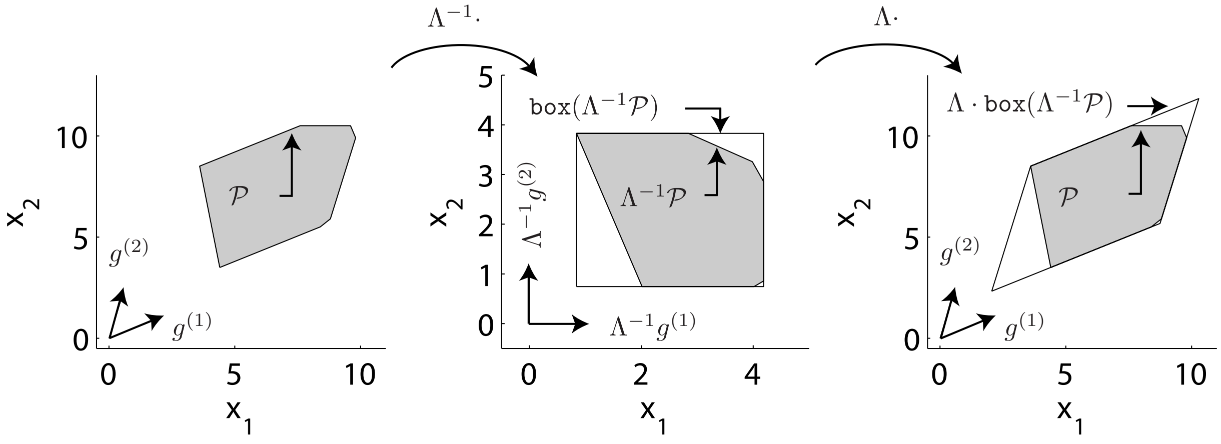


Fig. 2.4.: Parallelotope-enclosure of a polytope where $\Lambda = [g^{(1)}, g^{(2)}]$.

2.4. Operations on Zonotopes

The three most basic operations on sets for the reachability algorithms to be presented are: linear transformation, addition, and convex hull computation. In this work, reachable sets are represented by zonotopes and, thus, the three mentioned set operations are introduced for zonotopes only.

The linear map of a zonotope and the addition of two zonotopes are computationally cheap, which makes zonotopes attractive for reachability analysis. For two given zonotopes

$\mathcal{Z}_1 = (c^{(1)}, g^{(1)}, \dots, g^{(e)})$ and $\mathcal{Z}_2 = (c^{(2)}, f^{(1)}, \dots, f^{(u)})$, the linear map and addition are obtained as follows (see e.g. [100]):

$$\begin{aligned} L \cdot \mathcal{Z}_1 &= (Lc^{(1)}, Lg^{(1)}, \dots, Lg^{(e)}), \quad L \in \mathbb{R}^{n\times n} \\ \mathcal{Z}_1 + \mathcal{Z}_2 &= (c^{(1)} + c^{(2)}, g^{(1)}, \dots, g^{(e)}, f^{(1)}, \dots, f^{(u)}). \end{aligned} \tag{2.1}$$

Note that the Minkowski sum of two zonotopes is simply an addition of their centers and a concatenation of their generators, which directly follows from their definition as the Minkowski sum of generators.

Another important operation is the computation of the convex hull of two zonotopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$, which is defined as $\mathtt{CH}(\mathcal{Z}_1, \mathcal{Z}_2) := \{\alpha_1 \cdot a^{(1)} + \alpha_2 \cdot a^{(2)} | a^{(1)} \in \mathcal{Z}_1, a^{(2)} \in \mathcal{Z}_2, \alpha_1 + \alpha_2 = 1\}$. In general, the convex hull of two zonotopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$ is not a zonotope anymore, such that the tightest zonotope enclosing the convex hull has to be found: $\overline{\mathtt{CH}}(\mathcal{Z}_1, \mathcal{Z}_2) \supseteq \mathtt{CH}(\mathcal{Z}_1, \mathcal{Z}_2)$. This problem is complex [79], such that a rougher over-approximation is used which is proposed in [69] for two zonotopes of equal order and equal dimension:

$$\begin{aligned} \overline{\mathtt{CH}}(\mathcal{Z}_1, \mathcal{Z}_2) = \frac{1}{2}(&c^{(1)} + c^{(2)}, g^{(1)} + f^{(1)}, \dots, g^{(e)} + f^{e}, \\ &c^{(1)} - c^{(2)}, g^{(1)} - f^{(1)}, \dots, g^{(e)} - f^{(e)}). \end{aligned} \tag{2.2}$$

The generalization for zonotopes of different order is straightforward. Without loss of generality, the first zonotope $\tilde{\mathcal{Z}}_1$ has $\tilde{e}$ generators which is more than the u generators of the second zonotope $\mathcal{Z}_2$. For this reason, the first zonotope is split up into a zonotope with u and $\tilde{e} - u$ generators: $\tilde{\mathcal{Z}}_1 = \mathcal{Z}_{1,a} + \mathcal{Z}_{1,b}$. This allows the over-approximating convex hull to be computed as: $\overline{\mathtt{CH}}(\tilde{\mathcal{Z}}_1, \mathcal{Z}_2) = \overline{\mathtt{CH}}(\mathcal{Z}_{1,a}, \mathcal{Z}_2) + \mathcal{Z}_{1,b}$.

2.5. Order Reduction of Zonotopes

The order reduction of zonotopes is a further operation on zonotopes which is discussed in more detail. The order of a zonotope has been introduced as $\hat{\varrho} = \frac{e}{n}$, where e is the number of generators and n is the dimension of the Euclidean space. Order reduction techniques are very important as they limit the computational complexity of operations on zonotopes. Especially the conversion of zonotopes from G- to H-representation is computationally expensive for zonotopes of high order, as shown in Theorem 2.1. In consistency to all other operations, the order reduction is performed in an over-approximative way, such that the reduced zonotope $\mathcal{Z}^{red}$ has fewer generators and encloses the original zonotope $\mathcal{Z}^{orig}$ ($\mathcal{Z}^{red} \supseteq \mathcal{Z}^{orig}$).

In order to measure the performance of different order reduction techniques, an over-approximation measure for the reduced zonotopes $\mathcal{Z}^{red}$ is introduced first. Next, the general procedure for the order reduction is introduced. It is based on the selection of generators that are stretched in order to compensate for the generators that have been discarded. The selection process is firstly discussed for the reduction to parallelotopes, which are zonotopes of order $\hat{\varrho} = 1$. In a second step, the generator selection is extended to the case when zonotopes are reduced to zonotopes of order greater than one. To assess the quality of the over-approximations, randomly generated zonotopes are reduced and assessed according to the introduced over-approximation measure. The general scheme for the order reduction of zonotopes is as follows:

Proposition 2.5 (Order Reduction of a Zonotope): An over-approximating zonotope with reduced number of generators $\mathcal{Z}^{red}$ is obtained by splitting the zonotope $\mathcal{Z} = \check{\mathcal{Z}} + \tilde{\mathcal{Z}}$ into $\check{\mathcal{Z}}$, $\tilde{\mathcal{Z}}$ and applying an order reduction scheme so that $\tilde{\mathcal{Z}}$ is enclosed by a parallelotope Ψ:

$$\mathcal{Z}^{red} = \check{\mathcal{Z}} + \Psi, \quad \Psi = \Pi \cdot \mathtt{box}(\Pi^{-1}\tilde{\mathcal{Z}}),$$

where $\Pi \in \mathbb{R}^{n\times n}$ is of full rank. □

The proof is omitted, as it has already been shown in Prop. 2.4 that $\Pi \cdot \mathtt{box}(\Pi^{-1}\tilde{\mathcal{Z}})$ returns an enclosing parallelotope Ψ and it is obvious that its addition with the unreduced part $\check{\mathcal{Z}}$ returns an over-approximation of $\mathcal{Z}$. In order to decide how to best split $\mathcal{Z}$ and how to select Π, a measure for the performance of the order reduction is proposed.

2.5.1. Over-Approximation Measure

The proposed over-approximation measure υ is based on the ratio of the volume of the over-approximating set $\mathcal{V}^{red}$ compared to the original set $\mathcal{V}^{orig}$: $\mathtt{V}(\mathcal{V}^{red})/\mathtt{V}(\mathcal{V}^{orig})$ and $\mathtt{V}()$ is the operator returning the volume of a set. The set $\mathcal{V}$ denotes a general set (polytope, zonotope, etc.) as the over-approximation measure is defined such that it is applicable for general sets. It should also be considered that each dimension represents a variable whose values may differ in scale, such that they have to be normalized. For this reason, the considered sets are weighted by some weighting matrix $W = \mathtt{diag}(w)$, $w \in \mathbb{R}^n$, resulting in the ratio $\mathtt{V}(W \cdot \mathcal{V}^{red})/\mathtt{V}(W \cdot \mathcal{V}^{orig})$. Another aspect of the over-approximation measure is that the measure should allow the results of over-approximations obtained in spaces of different dimension n to be compared. For this reason, the n-th root ($n \hat{=}$ dimension) of the volume ratio is taken, so that the over-approximation measure is equivalent to the ratio of the edge length of n-dimensional cubes containing the corresponding volumes. Combining the mentioned aspects, the over-approximation measure is computed as:

Definition 2.5 (Over-approximation Measure of Reduced Sets): The over-approximation for the reduction of a set $\mathcal{V}^{orig}$ to a set $\mathcal{V}^{red}$ is measured with a scaling vector w according to the following formula:

$$\upsilon = \left(\frac{\mathtt{V}(W \cdot \mathcal{V}^{red})}{\mathtt{V}(W \cdot \mathcal{V}^{orig})}\right)^{\frac{1}{n}}, \quad W = \mathtt{diag}(w),\, w \in \mathbb{R}^n. \qquad □$$

In order to simplify the notation, it is assumed from now on that all sets $\mathcal{V}$ are represented in a normalized space, where the original sets have been mapped to $\mathcal{V}^{orig} := W \cdot \mathcal{V}^{orig}$ beforehand, such that the weighting matrix is now the identity matrix ($W = I$).

2.5.2. Generator Selection for Parallelotopes

The important question of how to select the generators in Π spanning the parallelotope Ψ in Prop. 2.5 is addressed next. In this work, the generators in Π are selected out of the e generators of the original zonotope $\mathcal{Z}$ – this has the advantage that the over-approximating parallelotope Ψ touches facets of the original zonotope. The separation of the original zonotope $\mathcal{Z}$ into $\check{\mathcal{Z}}$ and $\tilde{\mathcal{Z}}$ is discussed later.

The selected generators should maximize the over-approximation index υ of the resulting parallelotope Ψ. A straightforward approach is an exhaustive search by computing the over-approximation value υ for all possible combinations of n out of e generators:

Proposition 2.6 (Exhaustive Search for the Generator Matrix Π): The exhaustive search for the best combination of n out of e generators for the over-approximation of a zonotope $\tilde{\mathcal{Z}}$ by a parallelotope Ψ (Prop. 2.5) has complexity $\mathcal{O}(\binom{e}{n} \cdot e)$ with respect to the number of generators e. Instead of searching for generators which minimize the over-approximation measure υ, one can alternatively evaluate the over-approximation measure $\upsilon^* = \mathtt{V}(\mathcal{V}^{red})$, which is computationally less expensive. □

Proof: The exhaustive search allows the generation of $\binom{e}{n}$ different parallelotopes. The complexity for the computation of the parallelotopes with respect to the number of generators according to Prop. 2.5 is $\mathcal{O}(e)$. Thus, the overall complexity of the exhaustive search is $\mathcal{O}(\binom{e}{n} \cdot e)$.

As all parallelotopes are over-approximations of the same zonotope, the volume $\mathtt{V}(\mathcal{V}^{orig})$ and the dimension n are equal for all parallelotopes. Thus, the over-approximation index $\upsilon^* = \mathtt{V}(\mathcal{V}^{red})$ is minimal if $\upsilon = \big(\mathtt{V}(\mathcal{V}^{red})/\mathtt{V}(\mathcal{V}^{orig})\big)^{\frac{1}{n}}$ is minimal. □

Clearly, the exhaustive search is infeasible in higher dimensions and for zonotopes of high order. For this reason, the number of possible combinations $\binom{e}{n}$ for the exhaustive search has to be reduced. Two approaches are discussed, where one approach reduces the number of generators $\tilde{e} < e$ for the exhaustive search. The other approach passes only a subset of reasonable combinations of n generators.

Heuristic 2.1 (Generator Selection for the Generator Matrix Π): A heuristic which has proven effective for extracting $\tilde{e}$ promising generators $g^{(i_1)}, \dots, g^{(i_{\tilde{e}})}$ is to select the longest generators (2-norm), such that

$$\underbrace{\|g^{(i_1)}\|_2 \geq \dots \geq \|g^{(i_{\tilde{e}})}\|_2}_{\text{candidates for } \Pi} \geq \|g^{(i_{\tilde{e}+1})}\|_2 \geq \dots \geq \|g^{(i_e)}\|_2. \qquad \square$$

The complexity of this selection is $\mathcal{O}(j \cdot \log(j))$ due to the sorting of the norms and one can give the following statement:

Proposition 2.7 (Order Reduction Complexity for Fixed Zonotope Order): If $\tilde{e}$ is chosen as $\tilde{e} = n + \kappa$, the complexity with respect to κ is $\mathcal{O}(\kappa^{n+1})$, i.e. polynomial in κ for fixed n. □

Proof: The complexity of the exhaustive search is according to Prop. 2.6

$$\mathcal{O}\left(\binom{n+\kappa}{n} \cdot (n+\kappa)\right) = \mathcal{O}\left(\frac{(n+\kappa)!}{n!\,\kappa!} \cdot (n+\kappa)\right) = \mathcal{O}\left((n+\kappa)^n \cdot (n+\kappa)\right) = \mathcal{O}(\kappa^{n+1}).$$

□

Besides extracting the longest generators, clustering methods like k-means clusters have also been investigated. The data points of the clusters have been defined as the end points of the generators starting in the origin. Next, the vectors from the origin to the cluster centers have been chosen as the reduced set of generators. However, this approach did not perform better than the selection based on the length of the generators.

The other mentioned possibility to accelerate the generator selection is to compute the over-approximation index υ^* on a subset of possible combinations of generators. The subset of combinations of generators is selected by an alternative over-approximation measure, computed on all $\binom{e}{n}$ combinations of generators. Thus, the complexity of this technique is the same compared to the exhaustive search. However, the computational time is decreased, as the alternative over-approximation index is less costly than the computation of υ^*. In this work, the following alternative measure is proposed:

Heuristic 2.2 (Selection of Candidates for the Generator Matrix Π): Candidates for the generator matrix Π are selected by an exhaustive search using the alternative measure

$$\tilde{\upsilon} = \left|\texttt{det}[g^{(i_1)}, \dots, g^{(i_n)}]\right|^{-1},$$

where $|\texttt{det}[g^{(i_1)}, \dots, g^{(i_n)}]|$ is the volume spanned by the n selected generators. The basic idea is that generators which already span a large volume no longer have to be stretched so much in order to enclose the original zonotope. □

It is remarked that the proposed order reduction techniques are not stable if the representation of the zonotope is changed. Consider a zonotope with the longest generator g^{max}. After replacing this generator by $\frac{1}{2}g^{max}$ and $\frac{1}{2}g^{max}$, the zonotope is unchanged (only its representation is changed). However, it is likely that the generators $\frac{1}{2}g^{max}$ will no longer be selected, such that the over-approximation is changed. This problem will be the subject of future research.

2.5.3. Generator Selection for Zonotopes

It remains to properly separate the original zonotope $\mathcal{Z}$ into $\check{\mathcal{Z}}$ and $\tilde{\mathcal{Z}}$ as presented in Prop. 2.5. The reduction of $\tilde{\mathcal{Z}}$ to a parallelotope Ψ has already been addressed. This means that it is left to select the $\tilde{u}$ generators for $\check{\mathcal{Z}}$ by the following heuristic:

Heuristic 2.3 (Generator Selection for Unreduced Zonotope $\check{\mathcal{Z}}$): The generators of the unreduced zonotope $\check{\mathcal{Z}}$ in Prop. 2.5 are chosen as the longest ones (2-norm):

$$\underbrace{\|g^{(i_1)}\|_2 \geq \ldots \geq \|g^{(i_{\tilde{u}})}\|_2}_{\text{for } \check{\mathcal{Z}}} \geq \underbrace{\|g^{(i_{\tilde{u}+1})}\|_2 \geq \ldots \geq \|g^{(i_e)}\|_2}_{\text{for } \tilde{\mathcal{Z}}}$$

The center of the original zonotope $\mathcal{Z}$ can be assigned to either $\check{\mathcal{Z}}$ or $\tilde{\mathcal{Z}}$. □

An alternative reduction method which is not based on reducing $\tilde{\mathcal{Z}}$ to a parallelotope but to a multidimensional interval has proven useful for marginal reductions of zonotopes, i.e. the ratio of the new order $\hat{\varrho}^{new}$ compared to the old order $\hat{\varrho}^{old}$ is close to one. Thus, these methods are a special case of Prop. 2.5 where $\Pi = I$. Heuristics based on this special case have been proposed in [42, 69, 99, 100]. In [99, 100], zonotopes are modeled as Minkowski additions of parallelotopes on which also the order reduction is based. However, these methods are not applied in this work, since the more general modeling of Minkowski additions of line segments is used. The work of [42] also separates the generators into $\check{\mathcal{Z}}$ and $\tilde{\mathcal{Z}}$ by choosing the generators with the largest 2-norm for $\check{\mathcal{Z}}$. Another heuristic which has shown better results for reachability problems than [42] is proposed in [69], where the generators for $\check{\mathcal{Z}}$ are selected by a combination of the 1- and the inf-norm:

Heuristic 2.4 (Generator Selection by 1- and Inf-norm): A heuristic for reducing the order of zonotopes based on Prop. 2.5 with $\Pi = I$, separates the generators for $\check{\mathcal{Z}}$ and $\tilde{\mathcal{Z}}$ as follows:

$$\underbrace{\|g^{(i_1)}\|_1 - \|g^{(i_1)}\|_\infty \geq \ldots \geq \|g^{(i_{\check{u}})}\|_1 - \|g^{(i_{\check{u}})}\|_\infty}_{\text{for } \check{\mathcal{Z}}} \geq$$
$$\underbrace{\|g^{(i_{\check{u}+1})}\|_1 - \|g^{(i_{\check{u}+1})}\|_\infty \geq \ldots \geq \|g^{(i_e)}\|_1 - \|g^{(i_e)}\|_\infty}_{\text{for } \tilde{\mathcal{Z}}}$$

The chosen generators for $\tilde{\mathcal{Z}}$ are close to vectors with only one nonzero component and are therefore well approximated by an interval hull. □

It is next presented how to randomly generate zonotopes for the evaluation of the proposed reduction methods.

2.5.4. Randomly Generated Zonotopes

For the evaluation of the over-approximating methods, zonotopes are generated by independent randomized generators $g^{(i)}$. In a first step, randomized points are obtained which are uniformly distributed on a unit hypersphere. This is achieved by computing $x/\|x\|_2$, where each element of $x \in \mathbb{R}^n$ is a Gaussian random variable; see [125]. Next, the generator is defined as the vector from the origin to a point on the hypersphere, which is stretched by the length of the generators $l^{(i)} = \|g^{(i)}\|_2$ and $l^{(i)}$ has a uniform distribution within the interval $0 < l^{(i)} \leq l^{max}$. The randomly generated zonotopes are used for the following evaluations.

2.5.5. Numerical Evaluation

The numerical results from the evaluation of the direct conversion are firstly discussed for the case when the zonotopes are reduced to parallelotopes. The method which best performs for the reduction to parallelotopes is then used for the order reduction to zonotopes of order greater than one. For the reduction to parallelotopes, three methods are evaluated:

- *Method A* This method is equal to the exhaustive search.
- *Method B* The exhaustive search is performed on the $\tilde{e} = n + \kappa$ longest generators according to Heuristic 2.1, where n is the system dimension and κ is chosen to $\kappa = 8$.

- *Method C* Method C is based on Method B, which provides the $\tilde{e} = n + \kappa$ longest generators. Next, Heuristic 2.2 is applied to the remaining $\tilde{e}$ generators in order to find the best $\lambda = n + \tilde{\lambda}$ combinations of generators, where $\tilde{\lambda} = 3$. Finally, an exhaustive search is performed on the remaining λ combinations.

The results of the evaluation are presented in Tab. 2.1. The over-approximation index υ as well as the overall computational time t^{comp} are obtained from 100 randomized zonotopes according to Sec. 2.5.4. The computations were performed on an AMD Athlon64 3700+ processor (single core) in Matlab. Due to the complexity of computing the volume of the original zonotopes which is required in order to obtain υ, the evaluation for dimensions greater than 8 is intractable. It can be seen that Method C allows efficient computation while almost maintaining the performance of the exhaustive search, making this method the choice for the order reduction to zonotope of order greater than one.

Tab. 2.1.: Results for the order reduction to parallelotopes.

Method	mean of	mean	[min,max]	variance
	t^{comp} [s]:	of υ:	of υ:	of υ:
dimension $n = 2$, zonotope order $o = 2$				
A,B,C:	0.0033	1.0492	$[1.0007, 1.1127]$	0.0008
dimension $n = 2$, zonotope order $o = 6$				
A:	0.0278	1.0839	$[1.0421, 1.1154]$	0.0003
B:	0.0190	1.0839	$[1.0421, 1.1154]$	0.0003
C:	0.0036	1.0874	$[1.0427, 1.1670]$	0.0004
dimension $n = 4$, zonotope order $o = 2$				
A,B:	0.0297	1.1610	$[1.0373, 1.2734]$	0.0025
C:	0.0044	1.1610	$[1.0373, 1.2734]$	0.0025
dimension $n = 4$, zonotope order $o = 6$				
A:	5.1344	1.2679	$[1.2089, 1.3227]$	0.0005
B:	0.2046	1.2807	$[1.2137, 1.3447]$	0.0008
C:	0.0111	1.2964	$[1.2147, 1.3735]$	0.0010
dimension $n = 6$, zonotope order $o = 2$				
A,B:	0.3896	1.2631	$[1.1067, 1.3618]$	0.0032
C:	0.0185	1.2660	$[1.1067, 1.4326]$	0.0035
dimension $n = 8$, zonotope order $o = 2$				
A,B:	5.1394	1.3670	$[1.1992, 1.4995]$	0.0036
C:	0.5738	1.3703	$[1.1992, 1.5279]$	0.0040

The numerical results for the order reduction of general zonotopes as proposed in Prop. 2.5 are obtained as follows: The parallelotope Ψ is obtained by method C and the unreduced zonotope $\check{\mathcal{Z}}$ is obtained as described in Heuristic 2.3 by selecting the longest generators. This method is denoted by D_i, where i indicates the number of generators in $\check{\mathcal{Z}}$. The results obtained by the same set-up as for the reduction to parallelotopes are listed in Tab. 2.2. It can be observed that the performance increases only a little with an increasing number of unreduced generators. In addition, the computational times decrease since the number of generators to be reduced decreases, too, which makes the computation of the enclosing parallelotope Ψ more efficient.

Tab. 2.2.: Results for the order reduction to zonotopes of lower order.

Method	mean of t^{comp} [s]:	mean of v:	[min,max] of v:	variance of v:
dimension $n = 2$, zonotope order $o = 6$				
D_0:	0.0040	1.0908	$[1.0424, 1.1504]$	0.0005
D_2:	0.0040	1.0833	$[1.0306, 1.1733]$	0.0006
D_4:	0.0036	1.0628	$[1.0173, 1.1147]$	0.0005
dimension $n = 4$, zonotope order $o = 2$				
D_0:	0.0048	1.1570	$[1.0551, 1.2960]$	0.0025
D_2:	0.0035	1.0981	$[1.0031, 1.2343]$	0.0026
D_4:	0.0012	1.0000	$[1.0000, 1.0000]$	0.0000
dimension $n = 4$, zonotope order $o = 6$				
D_0:	0.0112	1.2930	$[1.2147, 1.3767]$	0.0011
D_2:	0.0110	1.2872	$[1.2068, 1.4200]$	0.0016
D_4:	0.0122	1.2685	$[1.1807, 1.3584]$	0.0014
dimension $n = 6$, zonotope order $o = 2$				
D_0:	0.0182	1.2718	$[1.1322, 1.3969]$	0.0035
D_2:	0.0076	1.2219	$[1.0747, 1.4309]$	0.0044
D_4:	0.0046	1.1163	$[1.0100, 1.2833]$	0.0036
dimension $n = 8$, zonotope order $o = 2$				
D_0:	1.0435	1.3679	$[1.1643, 1.5373]$	0.0045
D_2:	0.0712	1.3220	$[1.1568, 1.4942]$	0.0049
D_4:	0.0148	1.2208	$[1.0750, 1.4446]$	0.0047

Next, it is shown how to shorten the computational time for the conversion from G- to H-representation of zonotopes by reducing the order of the zonotopes with the presented methods.

2.5.6. Speeding up the Halfspace Conversion of Zonotopes

The order reduction techniques of zonotopes presented so far allow the conversion from G- to H-representation to be sped up, since fewer generators have to be considered. A disadvantage of this approach is that the obtained H-representation is an over-approximation, while the exact conversion is possible. However, the computational complexity does not allow an exact conversion to be made in high dimensional spaces. Two different techniques for the computation of an over-approximative H-representation of zonotopes are presented: A direct and an indirect approach.

The direct approach obtains a H-representation as follows:

1. Reduce the order of the zonotope.
2. Apply the conversion from G- to H-representation as presented in Theorem 2.1.

The indirect approach uses a different scheme:

1. Compute the ζ over-approximations of zonotopes by parallelotopes with the lowest

over-approximation index υ.

2. Convert the parallelotopes from G- to H-representation; see Lemma 2.1.
3. Intersect the ζ best parallelotopes in H-representation in order to obtain the over-approximating H-representation.

Both approaches are compared numerically with the same set of randomized zonotopes. The direct method is denoted by HD_i and is computed by the reduction methods D_i, where the indices indicate the number of unreduced generators. For the indirect method, the over-approximating parallelotopes are obtained using Method C and the intersection is performed by a standard software package for polytopes[3]. This method is denoted by HI_i, where the index refers to the number of intersected parallelotopes. The results of both methods are listed in Tab. 2.3 and Tab. 2.4. Note that method HD_2 is faster compared to HD_0 for 8 dimensions, because the computation of Ψ saves more time (due to fewer combinations for Π) than the more complicated halfspace conversion adds.

Tab. 2.3.: Numerical results for the direct conversion to H-representation.

Method	mean of t^{comp} [s]:	mean of υ:	[min,max] of υ:	variance of υ:
dimension $n = 2$, zonotope order $o = 6$				
HD_0:	0.0066	1.0905	$[1.0466, 1.1399]$	0.0005
HD_2:	0.0087	1.0887	$[1.0299, 1.2089]$	0.0010
HD_4:	0.0084	1.0631	$[1.0153, 1.1318]$	0.0006
dimension $n = 4$, zonotope order $o = 2$				
HD_0:	0.0108	1.1541	$[1.0104, 1.2598]$	0.0030
HD_2:	0.0175	1.0987	$[1.0147, 1.2590]$	0.0030
HD_4:	0.0702	1.0000	$[1.0000, 1.0000]$	0.0000
dimension $n = 4$, zonotope order $o = 6$				
HD_0:	0.0158	1.2979	$[1.2021, 1.3970]$	0.0010
HD_2:	0.0261	1.2917	$[1.2282, 1.4269]$	0.0014
HD_4:	0.0781	1.2579	$[1.1863, 1.3643]$	0.0013
dimension $n = 6$, zonotope order $o = 2$				
HD_0:	0.0213	1.2697	$[1.0956, 1.4117]$	0.0035
HD_2:	0.0757	1.2181	$[1.0653, 1.4488]$	0.0047
HD_4:	1.6047	1.1115	$[1.0117, 1.2508]$	0.0035

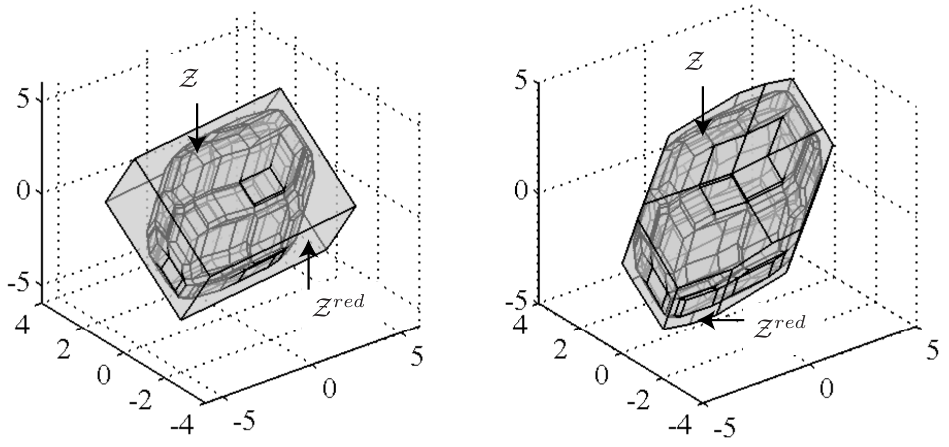
It can be observed that the indirect method clearly outperforms the direct method. The computational times are lower while the accuracy is higher, making the indirect method the preferred one. The difference in accuracy when choosing more or less intersecting parallelotopes is exemplarily illustrated in Fig. 2.5 for a three-dimensional zonotope. The left halfspace representation is obtained using method $\mathrm{HI}_1 (\hat{=}\ \mathrm{C})$ and the right one is obtained using method HI_4.

Finally, a short introduction to interval arithmetics is given.

[3] used tool: MPT-Toolbox [105]

Tab. 2.4.: Numerical results for the indirect conversion to H-representation.

Method	mean of t^{comp} [s]:	mean of v:	[min,max] of v:	variance of v:
dimension $n = 2$, zonotope order $o = 6$				
HI_1:	0.0068	1.0905	$[1.0466, 1.1399]$	0.0005
HI_2:	0.0136	1.0498	$[1.0052, 1.1306]$	0.0007
HI_4:	0.0273	1.0173	$[1.0006, 1.0688]$	0.0002
dimension $n = 4$, zonotope order $o = 2$				
HI_1:	0.0086	1.1541	$[1.0104, 1.2598]$	0.0030
HI_2:	0.0184	1.0814	$[1.0062, 1.1719]$	0.0014
HI_4:	0.0422	1.0300	$[1.0008, 1.0729]$	0.0002
dimension $n = 4$, zonotope order $o = 6$				
HI_1:	0.0158	1.2979	$[1.2021, 1.3970]$	0.0010
HI_2:	0.0252	1.2056	$[1.1306, 1.3010]$	0.0018
HI_4:	0.0479	1.1334	$[1.0842, 1.2587]$	0.0011
dimension $n = 6$, zonotope order $o = 2$				
HI_1:	0.0200	1.2697	$[1.0956, 1.4117]$	0.0035
HI_2:	0.0319	1.1639	$[1.0679, 1.2646]$	0.0022
HI_4:	0.0643	1.0941	$[1.0115, 1.1943]$	0.0011



(a) Method HI_1.

(b) Method HI_4.

Fig. 2.5.: Over-approximative halfspace representations.

2.6. Interval Arithmetics

Multidimensional intervals have already been introduced as a simple representation of sets in Def. 2.4. Besides the simple representation, the advantage of multidimensional intervals is that one can apply interval arithmetics – a technique which can be applied to most standard operations and functions. For this reason, interval arithmetics is often used as a last resort, when more accurate techniques for computing with sets fail or are too time consuming. A drawback of interval arithmetics is that it might be very conservative, resulting in unacceptable over-approximations.

An operation denoted by ∘ on two intervals $\mathcal{a} = [\underline{a}, \overline{a}] \in \mathcal{I}$ and $\mathcal{b} = [\underline{b}, \overline{b}] \in \mathcal{I}$ is generally defined as

$$\mathcal{a} \circ \mathcal{b} = \{a \circ b | a \in \mathcal{a},\, b \in \mathcal{b}\}.$$

In this work, the addition and multiplication of intervals is most frequently applied:

$$\begin{aligned} \mathcal{a} + \mathcal{b} =& [\underline{a} + \underline{b}, \overline{a} + \overline{b}], \\ \mathcal{a} \cdot \mathcal{b} =& [\min(\underline{a}\,\underline{b}, \underline{a}\,\overline{b}, \overline{a}\,\underline{b}, \overline{a}\,\overline{b}), \max(\underline{a}\,\underline{b}, \underline{a}\,\overline{b}, \overline{a}\,\underline{b}, \overline{a}\,\overline{b})]. \end{aligned} \tag{2.3}$$

With the foregoing formulas, one can compute the range of a function such as $\mathcal{c} = \mathcal{a} \cdot \mathcal{b} + \mathcal{a}$ where e.g. $\mathcal{a} = [-2, -1]$ and $\mathcal{b} = [-1, 1]$. Applying interval arithmetics, the computation of $\mathcal{c}$ can be performed in two ways:

$$\mathcal{c} = \mathcal{a} \cdot \mathcal{b} + \mathcal{a} = [-4, 1], \quad \mathcal{c} = \mathcal{a} \cdot (\mathcal{b} + 1) = [-4, 0].$$

Although interval arithmetics can guarantee that the exact solution is always included, only the second computation gives the exact solution. Exact results can be guaranteed if each variable occurs only once in interval computations such as in the second computation of $\mathcal{c}$. This is because for each evaluation of an interval operation, the values of the operands are allowed to take any value within the specified interval regardless of previous occurrences. As a consequence, different values of the same operand contribute to the minimum and maximum values of the corresponding interval operations, although the same operand is not allowed to have different values at the same time. In consistency to [97], expressions with single use of variables are referred to as *single-use expressions* (SUE). The problem of over-approximative results for non-single-use expressions is also referred to as the dependency problem in literature [87].

Interval arithmetics will also be applied to interval matrices $\mathcal{A} = [\underline{A}, \overline{A}] \in \mathcal{I}^{n \times n}$ where $\underline{A} \in \mathbb{R}$ and $\overline{A} \in \mathbb{R}$ are the left and right limit of the interval matrix.

The presented set representations and set operations are applied to reachability analysis in the next chapter.

3. Reachability Analysis

The introduced set representations (polytope, zonotope, multidimensional interval) and operations on them are used for reachability analysis in this chapter. An introduction to reachability analysis and a review of related literature is presented below.

3.1. Introduction and State of the Art

Basically, reachability analysis determines the set of states that a system can reach, starting from a set of initial states under the influence of a set of input trajectories and parameter values. A more exact definition is the following:

Definition 3.1 (Reachable Set at a Point in Time): Given is a dynamical system $\dot{x} = f(x(t), u(t), \rho(t))$, where t is the time, u is the input and ρ is the parameter vector. The set of possible initial states, the input, and the parameters are bounded by sets: $x(0) \in \mathcal{X}^0 \subset \mathbb{R}^n$, $u \in \mathcal{U} \subset \mathbb{R}^m$ and $\rho \in \mathcal{P} \subset \mathbb{R}^p$. The reachable set at a certain point in time r is defined as the union of possible system states at $t = r$:

$$\mathcal{R}(r) = \Big\{ x(r) = \int_0^r f(x(t), u(t), \rho(t)) dt \Big| x(0) \in \mathcal{X}^0,\ u([0,r]) \in \mathcal{U},\ \rho([0,r]) \in \mathcal{P} \Big\}. \qquad \square$$

Note that $u([0,r])$ is a short form of $\bigcup_{t\in[0,r]} u(t)$. The reachable set for a time interval is defined as:

Definition 3.2 (Reachable Set of a Time Interval): The reachable set of a time interval is the union of reachable sets at points in time within the interval $t \in [0, r]$:

$$\mathcal{R}([0,r]) = \bigcup_{t\in[0,r]} \mathcal{R}(t). \qquad \square$$

The extension of the continuous reachable set definition to hybrid systems is given in Sec. 3.5. As already mentioned in the introduction of this thesis, one of the main applications of reachability analysis is to check whether a system can reach a set of unsafe states; see Fig. 1.4. A set of unsafe states might be a set of critical distance between two mobile robots, or a set of dangerous concentration of certain chemicals in a reactor. Besides safety verification, there are other possible applications for reachability analysis:

- Performance assessment of control strategies: It can be checked if the system trajectories stay in a region around a reference trajectory, or reach a goal region around a setpoint.

- Scheduling: Reachability analysis can verify if the optimal schedule of a system (typically a production system) is ensured under all conditions, or if e.g. the supervisory controller might run into a recovery mode.
- Controller synthesis: The safety verification capabilities of reachability analysis can be used to find parameter sets of controllers that satisfy safety constraints.
- Deadlocks: Reachability analysis can determine whether a system might get stuck in a certain region of the continuous state space or an operation mode of a hybrid system.
- Set based observers: Instead of estimating the state of a system with stochastic methods (e.g. Kalman filtering), one can develop set based observers which return the set of possible states. Therefor, the set of successor states has to be computed via reachability analysis.

The exact reachable set of a continuous or hybrid system can only be obtained for certain system classes.

Exact Algorithms

Systems for which the exact reachable set can be computed are listed with the representative dynamics of the continuous state $x \in \mathbb{R}^n$: Timed automata [7] ($\dot{x} = 1$), multirate automata [6] ($\dot{x} = k$, $k \in \mathbb{R}^n$), rectangular automata and linear hybrid automata [8, 81, 143] ($\dot{x} \in \mathcal{P}$, $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P}$ is a polytope), and hybrid automata with linear continuous dynamics with a special system matrix A [107] ($\dot{x} = Ax + Bu$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $u \in \mathbb{R}^m$). Note that linear hybrid automata are differently defined than hybrid automata with linear continuous dynamics.

The above system representations for exact reachable set computations can be used to verify general hybrid systems with linear or nonlinear continuous dynamics. This can be achieved by a conservative abstraction, i.e. all possible behaviors of the complex system are represented by simpler dynamics. In [96] it has been shown how to abstract multi-affine continuous dynamics to a finite state automata. The abstraction of hybrid dynamics to timed automata has been studied in [55, 116, 155, 157] and by rectangular automata in [142]. Abstractions for general hybrid automata to linear hybrid automata have been developed in [64]. The abstraction of general nonlinear and hybrid dynamics to differential inclusions ($\dot{x} = [\underline{k}, \overline{k}]$) without a strict partitioning into linear hybrid automata has been presented in [83, 145, 146]. The partitioning which is not fixed in the state space is also called on-the-fly partitioning below.

Software tools for the verification of timed automata are UPPAAL [19] and Kronos [28]. Exemplary tools for the reachability analysis of linear hybrid automata are HyTech [82] and PHAVer [64], where the latter can also handle general hybrid systems and is more sophisticated.

Over-Approximative Algorithms

Besides the already mentioned system classes, hybrid systems with linear continuous dynamics ($\dot{x} = Ax + Bu$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $u \in \mathbb{R}^m$) have been widely researched. In many works, the linear continuous dynamics is considered without being embedded into a hybrid system, demonstrating the trend that reachability analysis becomes more popular in other communities besides hybrid systems.

As previously stated, the exact reachable set of linear continuous systems can only be computed in special cases [107]. Thus, the reachable set has to be computed in an over-approximative way in order to verify if an unsafe set is possibly reached; see Fig. 1.5. An important question thereby is how to properly represent the over-approximated sets. Note that this problem does not arise for linear hybrid systems since they can be exactly represented by polyhedra[1]. Several geometric representations for linear continuous systems have been investigated: Polytopes [39], griddy polyhedra [10], ellipsoids [102, 103], oriented rectangular hulls [158], zonotopes [69], or support functions [70] which unify the other mentioned representations. For linear systems with uncertain input, zonotopes [69, 72, 100] and support functions [70] have clearly outperformed existing methods, allowing the verification of systems with more than 100 continuous state variables. In [72, 78], a wrapping-free algorithm for linear systems is derived. The wrapping effect terms the problem that over-approximations are propagated through the computations at later time steps, causing an ongoing increase in the over-approximation. This means that the over-approximation in [72, 78] is tight, even when the reachable set is computed for a long time horizon – a property that is nearly as good as the possibility of exactly computing a reachable set.

Analogously to the computation with linear hybrid automata, the algorithms for linear continuous systems can be applied to general nonlinear systems or hybrid systems. This is achieved by conservative linearization, i.e. by considering the linearization error as an additional uncertain input of the linearized system. Abstraction to linear systems and multi-affine systems using a fixed partition has been investigated in [12, 13]. This method is also called *hybridization* since one obtains a hybrid system where each linearization region is subject to a different continuous dynamics. The disadvantage of the hybridization method is the limited scalability. One reason for this is the exponential growth of state space regions with the system dimension n. The other reason is the high computational effort for transitions between state space regions, which can be dropped when using on-the-fly partitioning with overlapping state-space regions. On-the-fly partitioning is applied in this thesis and in [49, 80].

The explicit computation of over-approximated reachable sets has been performed for polynomial nonlinear systems using Bézier control nets in [48] and the Bernstein expansion in [51]. For general nonlinear systems, global optimization techniques [39] and face lifting [50] have been applied.

Software packages that directly compute with general nonlinear dynamics include d/dt [14] and CheckMate [38]. There are more tools, but due to the vast improvements of reachability analysis for linear and nonlinear continuous systems, tools become quickly outdated while the tools for linear hybrid automata are already pretty mature. Assessments of different software tools for the verification of hybrid systems can be found in [114, 115, 152].

The works on reachability analysis of linear and nonlinear continuous systems can be extended to hybrid systems when additionally considering the switching in the continuous

[1] Polyhedra are sets defined by halfspaces. In contrast to polytopes, polyhedra are not necessarily bounded [170].

dynamics. This switching is determined by so-called guard sets, i.e. a transition to a different discrete state is enabled when entering a guard set. The additional consideration of guard sets modeled or over-approximated by polytopes is straightforward for works that represent reachable sets by polytopes; see e.g. [10, 39, 47]. For other representations, such as zonotopes, one deals with the problem that the intersection with a guard set no longer results in the representation of the reachable sets. Over-approximations of intersections with guard sets have been obtained for ellipsoids [27] and zonotopes [71].

In order to accelerate the verification of hybrid systems with reachability computations, general refinement strategies have been developed [40, 156]. An overview of recent progress in reachability analysis for different kinds of dynamic systems is given in [11].

Reachability analysis is also applied in order to obtain the set of successor states of set-based observers. Analogously to the computation of reachable sets, different representations have been used to bound the set of possible states of observers: polytopes [9], ellipsoids [22, 101, 148], multidimensional intervals [144], and zonotopes [5, 43].

Contributions

In this chapter, approaches to computing reachable sets represented by zonotopes are described for several system classes. First, existing methods for the computation of reachable sets for linear systems are presented in Sec. 3.2 – the specialty of this section is that the reachable set can be computed without the wrapping effect according to [72]. Next, this concept is extended to linear systems with uncertain parameters in Sec. 3.3. Note that most previous work focuses on linear systems with known parameters such as in [72]. Two different representations of uncertain parameters in the system matrix of the linear system are considered: interval matrices and matrix zonotopes. In Sec. 3.4, the reachability analysis for linear systems is extended to nonlinear systems. For this, the nonlinear dynamic is abstracted to a linear one using on-the-fly partitioning of the state space. This allows a conservative computation of the set of linearization errors to be obtained, which is added as an additional uncertain input. The last extension in Sec. 3.5 describes how to compute the reachable set when the dynamics can switch as specified by a hybrid automaton. The chapter closes with a short summary.

The part on linear systems with uncertain parameters specified by an interval matrix is based on work published in [180], but revised due to incorrect parts. The work on reachability analysis of nonlinear systems is published in [184] and on hybrid systems in [187, 191].

3.2. Linear Continuous Systems

This section deals with the computation of reachable sets of linear time invariant (LTI) systems with uncertain inputs – a system class which has been widely investigated. The reason for the recapitulation of methods developed for this system class is that these methods are the basis for the newly developed methods. The material of this section is closely related to [47, 69, 72, 78], and differs only in some details and the way the material is presented. In subsequent sections, novel extensions for linear systems with uncertain parameters and nonlinear systems are presented.

The considered linear continuous system can be written in the form

$$\dot{x} = A\,x + u(t), \quad x(0) \in \mathcal{X}^0 \subset \mathbb{R}^n, \quad u(t) \in \mathcal{U} \subset \mathbb{R}^n \tag{3.1}$$

Note that this system class includes linear systems of the form $\dot{x} = A\,x + B\,\mathtt{u}(t)$ with $B \in \mathbb{R}^{n\times m}$ and $\mathtt{u}(t) \in \mathcal{D} \subset \mathbb{R}^m$ when choosing $\mathcal{U} = \{B\,\mathtt{u}|\mathtt{u} \in \mathcal{D}\}$.

One important property of linear systems is that the superposition principle can be applied. This allows the separation of the solution $x(t) = x^h(t) + x^p(t)$ into a homogeneous solution $x^h(t)$ and an inhomogeneous solution $x^p(t)$. The homogeneous solution considers the solution with respect to the initial state when there is no input ($x(0) = x^0, u = 0$) and the inhomogeneous solution accounts for the input into the system when the initial state is the origin ($x(0) = 0, u \neq 0$). The reachable set of the homogeneous and the inhomogeneous solution are denoted by $\mathcal{H}^{\mathcal{R}}(t)$ and $\mathcal{P}^{\mathcal{R}}(t)$, respectively. The reachable set is then obtained as $\mathcal{R}(t) = \{x^h(t) + x^p(t)|x^h(t) \in \mathcal{H}^{\mathcal{R}}(t), x^p(t) \in \mathcal{P}^{\mathcal{R}}(t)\}$ which is the Minkowski addition of the reachable set of the homogeneous and the inhomogeneous solution.

3.2.1. Reachable Sets of Systems without Input

First, the reachable set of the homogeneous solution is presented, obtained by concatenating reachable sets of small time intervals.

Basic Procedure

The basic steps that are undertaken in order to compute the reachable set of the first time interval are in common with other approaches, such as e.g. [39, 69, 158]. The reachable set of a time interval $t \in [0, r]$, ($r \in \mathbb{R}^+$) is obtained by

1. computation of the reachable set $\mathcal{H}^{\mathcal{R}}(t)$ for $t = r$ based on the initial set $\mathcal{X}^0$,
2. generation of the convex hull of $\mathcal{X}^0$ and $\mathcal{H}^{\mathcal{R}}(r)$,
3. enlargement of the convex hull to ensure enclosure of all trajectories of the time interval $t \in [0, r]$.

The enlargement is necessary because of the curvature of trajectories. The mentioned steps are illustrated in Fig. 3.1. The reachable set of the first time interval is later used to compute the sets for further time intervals.

Time Point Solution

The homogeneous solution of a linear time invariant system is well known to be $x^h(r) = e^{Ar}\,x(0)$. After substitution of the initial value by the set of initial values, one obtains

$$\mathcal{H}^{\mathcal{R}}(r) = e^{Ar}\,\mathcal{X}^0.$$

For further time points $t = k \cdot r$, $k \in \mathbb{N}^+$, the reachable set is obtained as $\mathcal{H}^{\mathcal{R}}((k+1)r)$=

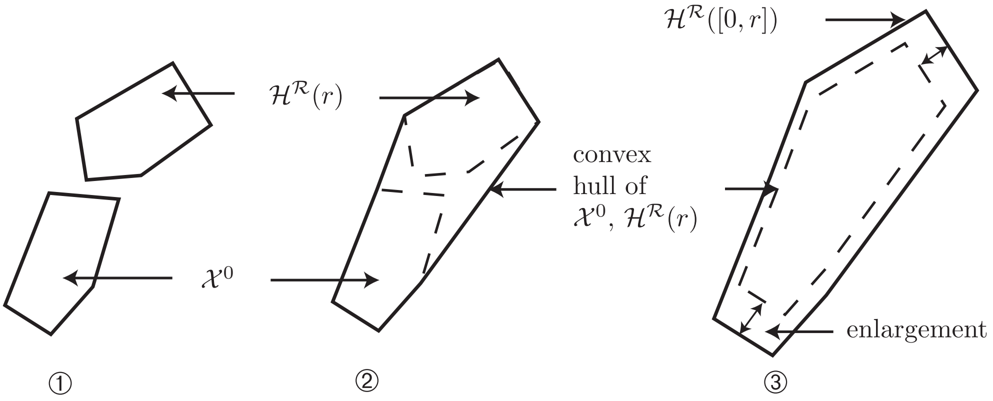

$\mathcal{H}^{\mathcal{R}}([0,r])$
$\mathcal{H}^{\mathcal{R}}(r)$
convex hull of $\mathcal{X}^0$, $\mathcal{H}^{\mathcal{R}}(r)$
$\mathcal{X}^0$
enlargement
①
②
③

Fig. 3.1.: Computation of the reachable set for a time interval.

$e^{Ar}\,\mathcal{H}^{\mathcal{R}}(kr)$, where the linear map is computed as presented in (2.1). For other representations, such as ellipsoids and polytopes, the computation of the linear maps can be found in [102] and [39], respectively.

There exist many different techniques for the computation of the matrix exponential e^{Ar} [122]. One of them is the computation by its Taylor series. Since the Taylor series can only be computed with a finite number of terms, the neglected terms are over-approximated by the remainder $\mathcal{E}(t)$:

$$\begin{aligned} e^{At} &= I + At + \frac{1}{2!}(At)^2 + \frac{1}{3!}(At)^3 + \dots \\ &\subset \sum_{i=0}^{\eta} \frac{1}{i!}(At)^i + \mathcal{E}(t), \end{aligned} \tag{3.2}$$

where

$$\mathcal{E}(t) = [-\mathbf{1}, \mathbf{1}] \frac{(\|A\|_\infty t)^{\eta+1}}{(\eta+1)!} \frac{1}{1-\epsilon}, \quad \epsilon = \frac{\|A\|_\infty t}{\eta+2} \overset{!}{<} 1 \tag{3.3}$$

and $\mathbf{1}$ is a matrix of ones so that $[-\mathbf{1}, \mathbf{1}]$ is an interval matrix whose elements vary between -1 and 1. The computation of the remainder $\mathcal{E}$ is taken from [110]. Note that for the computation of $\mathcal{E}$, it is required to choose η so that $\epsilon < 1$ as $\frac{1}{1-\epsilon}$ is obtained from the geometric series $1 + \epsilon + \epsilon^2 + \dots$ in [110].

Time Interval Solution

Given the solution $x(r) = e^{Ar}x(0)$, the following approximation for the homogeneous solution at intermediate points in time is suggested:

$$x^h(t) \approx x(0) + \frac{t}{r}(e^{Ar}x(0) - x(0)), \quad t \in [0, r].$$

The enclosure of all solutions starting in $\mathcal{X}^0$ is achieved when adding the uncertainty

$\mathcal{F}\,x(0)$, where $\mathcal{F}$ is an interval matrix ($\mathcal{F} \in I^{n\times n}$):

$$x^h(t) \in x(0) + \frac{t}{r}(e^{Ar}x(0) - x(0)) + \mathcal{F}\,x(0), \quad t \in [0, r]. \tag{3.4}$$

The computation of the uncertainty is derived in the following proposition.

Proposition 3.1 (Correction matrix $\mathcal{F}$): The interval matrix $\mathcal{F}$ which determines the enlargement $\mathcal{F}\,x(0)$ in (3.4) can be computed as:

$$\mathcal{F} = \sum_{i=2}^{\eta} [(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}})r^i, 0]\frac{A^i}{i!} + \mathcal{E}(r).$$

$\mathcal{E}(r)$ is the matrix exponential remainder and η the number of Taylor terms according to (3.3). The computations are performed using interval arithmetics. □

Proof: After replacing $x(t)$ by $e^{At}x(0)$ in (3.4), and division by $x(0)$, one obtains

$$e^{At} - I - \frac{t}{r}(e^{Ar} - I) \overset{!}{\in} \mathcal{F}, \quad t \in [0, r].$$

The substitution of e^{At} and e^{Ar} by its finite Taylor series according to (3.2) yields

$$\sum_{i=2}^{\eta}(t^i - t\cdot r^{i-1})\frac{1}{i!}A^i + \mathcal{E}(t) - \frac{t}{r}\mathcal{E}(r) \overset{!}{\in} \mathcal{F}, \quad t \in [0, r].$$

Note that the linear terms cancel out so that the remaining expression contains only terms of quadratic or higher order. The interval of $t^i - t\cdot r^{i-1}$ for $t \in [0, r]$ is obtained exactly by computing the minimum and maximum of $t^i - t\cdot r^{i-1}$ for which only one extreme value exists: $\frac{d}{dt}(t^i - t\cdot r^{i-1}) \overset{!}{=} 0 \to t_{min} = i^{-\frac{1}{i-1}}r$. This means that the maximum values are to be found at the borders of $t \in [0, r]$, which are both 0 for $t = 0$ and $t = r$. From this follows that the exact interval can be computed as

$$[(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}})r^i, 0] = \{t^i - t\cdot r^{i-1} | t \in [0, r]\}.$$

It remains to compute bounds of the interval matrix exponential remainder $\mathcal{E}$ for $t \in [0, r]$. The remainder can be written as $\mathcal{E}(t) = [-\mathbf{1}, \mathbf{1}]\cdot\phi(t)$ after introducing $\phi(t) = \frac{(\|A\|_\infty t)^{\eta+1}}{(\eta+1)!}\frac{1}{1-\epsilon}$; see (3.3). As $\phi(t)$ is strictly increasing, it follows that $\phi(t) \in [0, 1]\cdot\phi(r)$ for $t \in [0, r]$. From this follows that $\phi(t) - \frac{t}{r}\phi(r) \in [0, 1]\phi(r) - [0, 1]\phi(r) \subseteq [-1, 1]\phi(r)$. Thus,

$$\mathcal{E}(t) - \frac{t}{r}\mathcal{E}(r) = [-\mathbf{1}, \mathbf{1}]\left(\phi(t) - \frac{t}{r}\phi(r)\right) \subset [-\mathbf{1}, \mathbf{1}][-1, 1]\phi(r) = \mathcal{E}(r). \qquad \square$$

When the initial state $x(0)$ is substituted by the set of initial states $\mathcal{X}^0$, (3.4) can be generalized to

$$\mathcal{H}^{\mathcal{R}}([0, r]) = \mathtt{CH}(\mathcal{X}^0, e^{Ar}\mathcal{X}^0) + \mathcal{F}\mathcal{X}^0.$$

The use of the convex hull computation follows directly the definition $\mathtt{CH}(\mathcal{X}^0, \mathcal{H}^\mathcal{R}(r)) := \Big\{x(0) + \alpha(x^h(r) - x(0)) | x(0) \in \mathcal{X}^0,\, x^h(r) \in \mathcal{H}^\mathcal{R}(r),\, \alpha \in [0,1]\Big\}$. Because of the representation by zonotopes, the convex hull cannot be exactly obtained so that the over-approximation suggested in (2.2) is applied.

Since the multiplication with e^{Ar} yields the set of successor states after a time increment of r, the next time intervals are obtained by

$$\mathcal{H}^\mathcal{R}([kr, (k+1)r]) = e^{Ar}\mathcal{H}^\mathcal{R}([(k-1)r, kr]).$$

Finally, the algorithm for the computation of reachable sets of linear time invariant systems can be formulated in Alg. 1. Note that for notation reasons, the reachable set at time intervals is indicated by an index only ($\mathcal{H}^\mathcal{R}_k \hat{=} \mathcal{H}^\mathcal{R}([kr, (k+1)r])$).

Algorithm 1 Compute $\mathcal{H}^\mathcal{R}([0, t_f])$

```
Input: Initial set $\mathcal{X}^0$, matrix exponential $e^{Ar}$, correction matrix $\mathcal{F}$, time horizon $t_f$
Output: $\mathcal{H}^\mathcal{R}([0, t_f])$

$\mathcal{H}^\mathcal{R}_0 = \mathtt{CH}(\mathcal{X}^0, e^{Ar}\mathcal{X}^0) + \mathcal{F}\mathcal{X}^0$
for $k = 1 \ldots t_f/r - 1$ do
    $\mathcal{H}^\mathcal{R}_k = e^{Ar}\mathcal{H}^\mathcal{R}_{k-1}$
end for
$\mathcal{H}^\mathcal{R}([0, t_f]) = \bigcup_{k=1}^{t_f/r} \mathcal{H}^\mathcal{R}_{k-1}$
```

Next, uncertain inputs are additionally considered in the reachable set computation.

3.2.2. Reachable Sets of Systems with Input

The inhomogeneous solution $x^p(r)$ of a linear time invariant system with system matrix A and uncertain input $u(t) \in \mathcal{U}$ is well known to be:

$$x^p(r) = e^{Ar} \int_0^r e^{-At} u(t)\, dt = \int_0^r e^{A(r-t)} u(t)\, dt. \tag{3.5}$$

Considering that the possible inputs can be taken from a set $\mathcal{U}$, the inhomogeneous solution is bounded by the following set: $x^p(r) \in \int_0^r e^{A(r-t)} \mathcal{U}\, dt$. For the case that the input is constant within $t \in [0, r]$, the following solution can be obtained:

$$x^p(r) \in \int_0^r e^{A(r-t)} \mathcal{U}\, dt \overset{u=\mathtt{const}}{=} \int_0^r e^{A(r-t)}\, dt\, \mathcal{U} = A^{-1}(e^{Ar} - I)\mathcal{U}. \tag{3.6}$$

However, a constant input within one time step is not the case in general, such that for varying $u(t)$ an over-approximation has to be computed:

Theorem 3.1 (Over-approximation of the Reachable Set due to Inputs): The over-approximated reachable set due to inputs $u(t) \in \mathcal{U}$ can be computed as

$$\mathcal{P}^{\mathcal{R}}(r) = \sum_{i=0}^{\eta} \left(\frac{A^i \, r^{i+1}}{(i+1)!} \mathcal{U} \right) + \mathcal{E}(r) \cdot r \cdot \mathcal{U}, \tag{3.7}$$

where η is the number of Taylor terms used for the over-approximation of $\mathcal{P}^{\mathcal{R}}(r)$ and $\mathcal{E}(r)$ is the remainder of the Taylor expansion; see (3.3). □

The proof can be found in Appendix A.1. The same result has been derived in [78] using support functions. In order to obtain an algorithmic solution for further points in time or time intervals, the following proposition is presented.

Proposition 3.2 (Separation of the Input solution): The reachable set $\mathcal{P}^{\mathcal{R}}(kr+\Delta t)$ due to the input $u(t)$ can be computed as

$$\mathcal{P}^{\mathcal{R}}(kr + \Delta t) = e^{Ar} \mathcal{P}^{\mathcal{R}}((k-1)r + \Delta t) + \mathcal{P}^{\mathcal{R}}(r)$$

which implies

$$\mathcal{P}^{\mathcal{R}}([kr, (k+1)r]) = e^{Ar} \mathcal{P}^{\mathcal{R}}([(k-1)r, kr]) + \mathcal{P}^{\mathcal{R}}(r)$$

when Δt can take values within $[0, r]$. □

Proof: The input solution in (3.5) due to the input $u(t) \in \mathcal{U}$ can be reformulated to

$$\begin{aligned} \mathcal{P}^{\mathcal{R}}(kr + \Delta t) &= \int_0^{kr+\Delta t} e^{A(kr+\Delta t-\tau)} \mathcal{U} \, d\tau \\ &= \int_0^{(k-1)r+\Delta t} e^{A(kr+\Delta t-\tau)} \mathcal{U} \, d\tau + \int_{(k-1)r+\Delta t}^{kr+\Delta t} e^{A(kr+\Delta t-\tau)} \mathcal{U} \, d\tau \\ &= e^{Ar} \int_0^{(k-1)r+\Delta t} e^{A((k-1)r+\Delta t-\tau)} \mathcal{U} \, d\tau + \int_0^{r} e^{A(r-\tau)} \mathcal{U} \, d\tau \\ &= e^{Ar} \mathcal{P}^{\mathcal{R}}((k-1)r + \Delta t) + \mathcal{P}^{\mathcal{R}}(r). \end{aligned}$$

□

Corollary 3.1 (Alternative Separation of the Input solution): The reachable set $\mathcal{P}^{\mathcal{R}}(r + \Delta t)$ can be computed as $\mathcal{P}^{\mathcal{R}}(r + \Delta t) = e^{Ar} \mathcal{P}^{\mathcal{R}}(\Delta t) + \mathcal{P}^{\mathcal{R}}(r)$. □

The proof is omitted since the above corollary is obtained after substituting Δt by $\Delta t - (k-1)r$ in Prop. 3.2.

Based on Prop. 3.2 and the homogeneous solution, the algorithm for the reachable set can be formulated as

$$\begin{aligned} \mathcal{H}^{\mathcal{R}}([kr, (k+1)r]) &= e^{Ar} \mathcal{H}^{\mathcal{R}}([(k-1)r, kr]), \\ \mathcal{P}^{\mathcal{R}}([kr, (k+1)r]) &= e^{Ar} \mathcal{P}^{\mathcal{R}}([(k-1)r, kr]) + \mathcal{P}^{\mathcal{R}}(r), \\ \mathcal{R}([kr, (k+1)r]) &= \mathcal{H}^{\mathcal{R}}([kr, (k+1)r]) + \mathcal{P}^{\mathcal{R}}([kr, (k+1)r]). \end{aligned}$$

which is equivalent to

$$\mathcal{R}([kr, (k+1)r]) = e^{Ar} \mathcal{R}([(k-1)r, kr]) + \mathcal{P}^{\mathcal{R}}(r). \tag{3.8}$$

Since zonotopes are used to represent $\mathcal{H}^{\mathcal{R}}$ and $\mathcal{P}^{\mathcal{R}}$, the addition and multiplication is performed as presented in (2.1). Due to the addition of zonotopes, the order of $\mathcal{R}([kr,(k+1)r])$ is increasing with each time step k. One possibility to cope with this problem is to reduce the order of the zonotope as presented in Sec. 2.5. However, the reduction causes an over-approximation such that this error is propagated through the computations at later time steps, which is also referred to as the *wrapping effect*.

In [72] it has been shown that the reachable set of linear time invariant systems can be computed without the wrapping effect. This is achieved by a clever reordering of the computations in (3.8). Besides the reordering, the `box()`-operator is used in order to prevent the growing order of the zonotopes representing the reachable sets. The wrapping-free algorithm is presented in Alg. 2. Note that indices represent time intervals for better readability, e.g. $\mathcal{R}_k = \mathcal{R}([kr,(k+1)r])$. If $\mathcal{P}^{\mathcal{R}}(r)$ would be exact, one could give the following statement: The over-approximations due to the `box()`-operator are tight in the sense of [102], i.e. the exact reachable set has at least a common point with each of the faces of the over-approximated reachable set. It is further remarked that the wrapping effect cannot be avoided when computing with more complicated system classes, such as linear time varying systems or nonlinear systems.

Algorithm 2 Compute $\mathcal{R}([0,t_f])$

```
Input: Initial set $\mathcal{X}^0$, matrix exponential $e^{Ar}$, input set $\mathcal{U}$, correction matrix $\mathcal{F}$, time horizon $t_f$
Output: $\mathcal{R}([0,t_f])$

  $\mathcal{H}^{\mathcal{R}}_0 = \mathtt{CH}(\mathcal{X}^0, e^{Ar}\mathcal{X}^0) + \mathcal{F}\mathcal{X}^0$
  $V_0 = \sum_{i=0}^{\eta}\left(\frac{A^i r^{i+1}}{(i+1)!}\mathcal{U}\right) + \mathcal{E}(r)\cdot r\cdot\mathcal{U}$
  $\mathcal{P}^{\mathcal{R}}_0 = \mathtt{box}(V_0)$
  $\mathcal{R}_0 = \mathcal{H}^{\mathcal{R}}_0 + \mathcal{P}^{\mathcal{R}}_0$
  for $k = 1\ldots(t_f/r-1)$ do
    $\mathcal{H}^{\mathcal{R}}_k = e^{Ar}\,\mathcal{H}^{\mathcal{R}}_{k-1}$
    $V_k = e^{Ar}\,V_{k-1}$
    $\mathcal{P}^{\mathcal{R}}_k = \mathcal{P}^{\mathcal{R}}_{k-1} + \mathtt{box}(V_k)$
    $\mathcal{R}_k = \mathcal{H}^{\mathcal{R}}_k + \mathcal{P}^{\mathcal{R}}_k$
  end for
  $\mathcal{R}([0,t_f]) = \bigcup_{k=1}^{t_f/r}\mathcal{R}_{k-1}$
```

In Alg. 2 it is assumed that $\mathcal{P}^{\mathcal{R}}([0,r]) = \mathcal{P}^{\mathcal{R}}(r)$ since V_0 is computed according to Theorem 3.1. However, this is only possible when the origin is contained in the set of inputs $\mathcal{U}$ as shown next. The other case when the origin is not contained is presented later.

Origin is Contained in Input Set

The reason for separating the computation in two cases, depending on whether the origin is contained in the input set, is justified by the following proposition.

Proposition 3.3 (Enclosure of the Inhomogeneous Solution): If the origin is contained in the input set ($0 \in \mathcal{U}$), the reachable set of the inhomogeneous solution $\mathcal{P}^{\mathcal{R}}(r)$

encloses the reachable set of a previous point in time, i.e. $\mathcal{P}^{\mathcal{R}}(r) \subseteq \mathcal{P}^{\mathcal{R}}(r+\Delta t)$ and $0 < \Delta t$. □

Proof: If $\mathcal{U}$ contains the origin, the reachable set $\mathcal{P}^{\mathcal{R}}(t)$ and thus $e^{Ar}\mathcal{P}^{\mathcal{R}}(\Delta t)$ contain the origin, too, which follows directly from (3.7). If a general set $B \subset \mathbb{R}^n$ contains the origin ($0 \in B$), it follows that $A \subseteq A + B$, $A \subset \mathbb{R}^n$. Since $e^{Ar}\mathcal{P}^{\mathcal{R}}(\Delta t)$ contains the origin, it follows that $\mathcal{P}^{\mathcal{R}}(r+\Delta t) \overset{\textit{Corollary 3.1}}{=} e^{Ar}\mathcal{P}^{\mathcal{R}}(\Delta t) + \mathcal{P}^{\mathcal{R}}(r) \supseteq \mathcal{P}^{\mathcal{R}}(r)$. □

From this follows that $\mathcal{P}^{\mathcal{R}}([0,r]) = \mathcal{P}^{\mathcal{R}}(r)$. Next, the case is considered when the origin is not contained in the input set.

Origin is not Contained in Input Set

If the origin is not contained in the input set ($0 \notin \mathcal{U}$), the input set is split into a constant vector $\tilde{u}$ and a set $\tilde{\mathcal{U}}$ in which the origin is contained ($0 \in \tilde{\mathcal{U}}$). The reachable set for a time interval due to $\tilde{\mathcal{U}}$ is computed as previously shown and the reachable set of the constant input $\tilde{u}$ is obtained similarly to the time interval solution of the homogeneous solution.

Analogously to the homogeneous solution, the solution due to the constant input $\tilde{u}$ is approximated as

$$\tilde{x}^p(t) \approx \frac{t}{r} A^{-1}(e^{Ar} - I)\,\tilde{u}, \quad t \in [0,r]$$

and enlarged, such that

$$\tilde{x}^p(t) \in \frac{t}{r}\tilde{x}^p(r) + \tilde{\mathcal{F}}\,\tilde{u} = \frac{t}{r} A^{-1}(e^{Ar} - I)\,\tilde{u} + \tilde{\mathcal{F}}\,\tilde{u}, \quad \tilde{\mathcal{F}} = A^{-1}\mathcal{F}, \quad t \in [0,r]. \tag{3.9}$$

The foregoing result $\tilde{\mathcal{F}} = A^{-1}\mathcal{F}$ can be shown analogously to the proof of Prop. 3.1:

$$\tilde{\mathcal{F}} \subseteq A^{-1}(e^{At} - I) - \frac{t}{r}A^{-1}(e^{Ar} - I) = A^{-1}\underbrace{\left[e^{At} - I - \frac{t}{r}(e^{Ar} - I)\right]}_{\in \mathcal{F}}, \quad t \in [0,r].$$

In the case that the inverse of A does not exist, one can multiply A^{-1} with the terms in $\mathcal{F}$ so that A^{-1} cancels out. Because of the similar structure of the homogeneous solution in (3.4) and the inhomogeneous solution caused by the input $\tilde{u}$ in (3.9), both solutions are unified in the following:

$$\begin{array}{llll} x^h(t) + \tilde{x}^p(t) \in & x(0) & +\frac{t}{r}\left[e^{Ar}x(0) - x(0)\right] & +\mathcal{F}\,x(0)+ \\ & 0 & +\frac{t}{r}\left[\tilde{x}^p(r) - 0\right] & +A^{-1}\mathcal{F}\,\tilde{u}, \quad t \in [0,r] \end{array} \tag{3.10}$$

$$\rightarrow \tilde{\mathcal{R}} := \mathcal{H}^{\mathcal{R}}(t) + \tilde{\mathcal{P}}(t) = \mathtt{CH}(\mathcal{X}^0, e^{Ar}\mathcal{X}^0 + \tilde{\mathcal{P}}(r)) + \mathcal{F}\mathcal{X}^0 + A^{-1}\mathcal{F}\,\tilde{u}$$

This result is used to extend Alg. 2 to Alg. 3. In order to keep the previous computation scheme, the inhomogeneous solution $\tilde{\mathcal{P}}$ is subtracted in the computation of $\tilde{\mathcal{R}}_0$ and added to $\mathcal{R}_0$. This ensures that the homogeneous reachable set at points in time kr is $e^{Akr}\mathcal{R}(0)$. In order to increase accuracy, the inhomogeneous solution $\tilde{\mathcal{P}}$ is excluded from the `box()`-operator.

It is also remarked that Alg. 2 is a special case of Alg. 3 for $\tilde{u} = 0$. Since the computations within the loop are the same as for Alg. 2, the more general algorithm does not suffer from the wrapping effect either.

Algorithm 3 Compute $\mathcal{R}([0, t_f])$

```
Input: Initial set $\mathcal{X}^0$, matrix exponential $e^{Ar}$, input set $\mathcal{U}$, correction matrix $\mathcal{F}$, time
    horizon $t_f$
Output: $\mathcal{R}([0, t_f])$

  $\tilde{\mathcal{P}}(r) = A^{-1}(e^{Ar} - I)\tilde{u}$
  $\tilde{\mathcal{R}}_0 = \mathtt{CH}(\mathcal{X}^0, e^{Ar}\mathcal{X}^0 + \tilde{\mathcal{P}}(r)) - \tilde{\mathcal{P}}(r) + \mathcal{F}\mathcal{X}^0 + A^{-1}\,\mathcal{F}\,\tilde{u}$
  $V_0 = \sum_{i=0}^{\eta}\left(\frac{A^i\,r^{i+1}}{(i+1)!}\mathcal{U}\right) + \mathcal{E}(r)\cdot r\cdot\mathcal{U}$
  $\mathcal{P}^{\mathcal{R}}_0 = \tilde{\mathcal{P}}(r) + \mathtt{box}(V_0 - \tilde{\mathcal{P}}(r))$
  $\mathcal{R}_0 = \tilde{\mathcal{R}}_0 + \mathcal{P}^{\mathcal{R}}_0 + \tilde{\mathcal{P}}(r)$
  for $k = 1 \ldots t_f/r - 1$ do
    $\tilde{\mathcal{R}}_k = e^{Ar}\,\tilde{\mathcal{R}}_{k-1}$
    $V_k = e^{Ar}\,V_{k-1}$
    $\mathcal{P}^{\mathcal{R}}_k = \mathcal{P}^{\mathcal{R}}_{k-1} + \mathtt{box}(V_k)$
    $\mathcal{R}_k = \tilde{\mathcal{R}}_k + \mathcal{P}^{\mathcal{R}}_k$
  end for
  $\mathcal{R}([0, t_f]) = \bigcup_{k=1}^{t_f/r}\mathcal{R}_{k-1}$
```

3.2.3. Numerical Examples

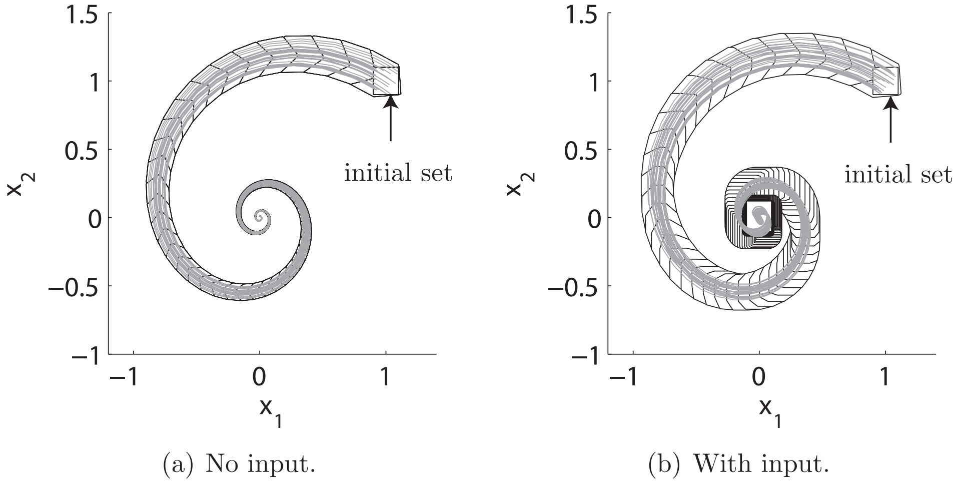
The proposed algorithms 1-3 are applied to numerical examples that have been proposed in [69], but with different input sets $\mathcal{U}$. The first example is given as

$$\dot{x} = \begin{bmatrix} -1 & -4 \\ 4 & -1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t), \quad x(0) \in \begin{bmatrix} [0.9, 1.1] \\ [0.9, 1.1] \end{bmatrix}, \, u(t) \in [-0.1, 0.1].$$

The reachable sets have been computed for a time horizon of $t_f = 5$ and a time step size of $r = 0.04$, which results in 125 iterations. The reachable set of the homogeneous solution is computed according to Alg. 1 and the result is shown in Fig. 3.2(a) together with randomly generated trajectories. The overall reachable set in Fig. 3.2(b) is computed according to Alg. 2, since the origin is enclosed in the input set.

The differential equation of the second numerical example is

$$\dot{x} = \begin{bmatrix} -1 & -4 & 0 & 0 & 0 \\ 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & -3 & 1 & 0 \\ 0 & 0 & -1 & -3 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{bmatrix} x + u(t), \quad x(0) \in \begin{bmatrix} [0.9, 1.1] \\ [0.9, 1.1] \\ [0.9, 1.1] \\ [0.9, 1.1] \\ [0.9, 1.1] \end{bmatrix}, \, u(t) \in \begin{bmatrix} [0.9, 1.1] \\ [-0.25, 0.25] \\ [-0.1, 0.1] \\ [0.25, 0.75] \\ [-0.75, -0.25] \end{bmatrix}. \tag{3.11}$$



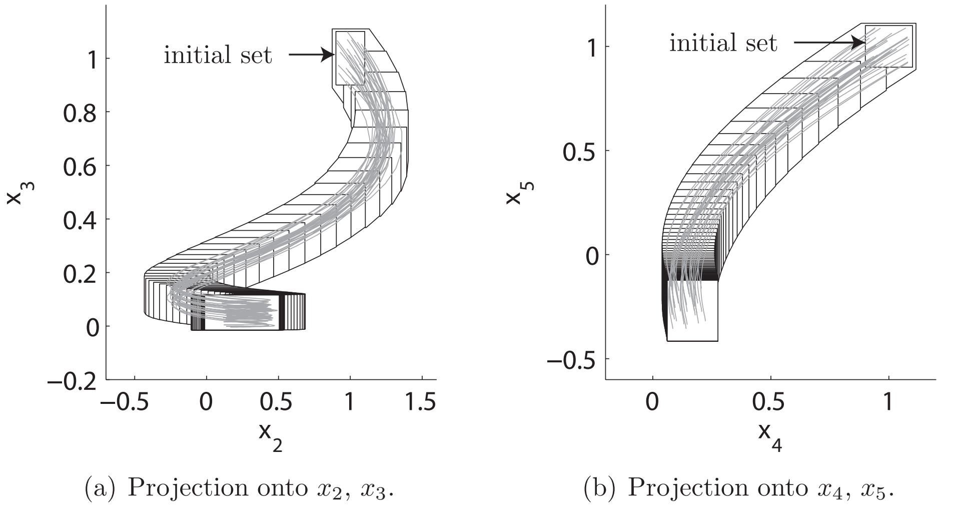
(a) No input.

(b) With input.

Fig. 3.2.: Reachable sets of the two-dimensional example.

For the second example, the time horizon and the time step size have been chosen equally to the previous example to $t_f = 5$ and $r = 0.04$. The reachable set is computed according to Alg. 3 since the origin is not enclosed, and visualized in Fig. 3.3 for selected projections. The two-dimensional projections are obtained by multiplying the computed zonotopes with projection matrices $P \in \mathbb{R}^{2\times n}$.

In order to show the scalability of the algorithm, the computation times for randomly generated examples of higher order are listed in Tab. 3.1. All reachable sets have been computed with 125 iterations. The computations were performed with Matlab on a single core desktop PC with an AMD Athlon64 3700+ processor. Note that the computation times for dimensions 5, 10, and 20 differ only marginally due to the overhead time caused by function calls.



(a) Projection onto x_2, x_3.

(b) Projection onto x_4, x_5.

Fig. 3.3.: Reachable sets of the five-dimensional example.

Tab. 3.1.: Computational times.

Dimension n	5	10	20	50	100
CPU-time [s]	0.08	0.11	0.17	0.41	2.05

3.3. Linear Continuous Systems with Uncertain Parameters

The previously presented approach for linear time invariant (LTI) systems is extended to the case where the system matrix contains uncertain parameters which are constant over time.

There have been early attempts to bound the solution of initial-value problems of linear systems with uncertain parameters in [130–132]. This paper series showed reachability results for a single independent parameter $\theta \in \mathcal{I}$ resulting in a system matrix $A = G_1 + \theta \cdot G_2$, $G_1, G_2 \in \mathbb{R}^{n\times n}$ at certain points in time without input. In this thesis, the more general case in which all elements of A can be uncertain is considered. Additionally, reachability is not limited to the case of discrete time, and time varying inputs are allowed. In [145, 146] uncertain parameters are allowed, too, but the reachable sets are represented more conservatively by multidimensional intervals.

In this thesis, two different forms in which parameter uncertainties may occur in the system matrix are investigated. The first one allows the system matrix to be uncertain within an interval matrix and the second one within a matrix zonotope. Both representations of parameter uncertainties are introduced later in more detail.

In general, the basic procedure for computing the reachable set is not changed compared to the previous one for LTI systems. The considered system class is precisely defined by the differential equation

$$\dot{x} = A(\rho)\, x + u(t), \quad x(0) \in \mathcal{X}^0 \subset \mathbb{R}^n, \quad u(t) \in \mathcal{U} \subset \mathbb{R}^n, \quad \rho \in \mathcal{I}^p, \tag{3.12}$$

where $\rho \in \mathcal{I}^p$ is a constant parameter vector whose elements can take values within intervals, and p is the number of parameters. The elements of the parameter vector are denoted by $\rho^{(i)}$ and may vary pairwise independently within the specified intervals. The set of possible system matrices for a certain set of parameter vectors is defined as $\mathcal{A} = \{A(\rho)|\rho \in \mathcal{I}^p\}$, which allows the homogeneous and inhomogeneous solution to be over-approximated. The homogeneous solution of (3.12) at time $t = r$, starting from the initial state $x(0)$, is well known to be calculated by $x(r) = e^{Ar}x(0)$. Thus, the reachable set of the homogeneous solution can be obtained as

$$\mathcal{H}^{\mathcal{R}}(r) = e^{\mathcal{A}r}\mathcal{X}^0 = \{e^{Ar}x(0)|A \in \mathcal{A}, x(0) \in \mathcal{X}^0\}. \tag{3.13}$$

The problem of computing the exponential for a set of matrices and the multiplication of the resulting set with a zonotope is discussed later. The inhomogeneous solution $x^p(r)$

is computed as previously shown in (3.5) by $x^p(r) = \int_0^r e^{A(r-t)} u(t)\, dt$ so that one can over-approximate the inhomogeneous solution using the set of inputs $\mathcal{U}$:

$$x^p(r) \in \int_0^r e^{\mathcal{A}(r-t)} u(t)\, dt \subseteq \int_0^r e^{\mathcal{A}(r-t)}\, \mathcal{U}\, dt = \int_0^r e^{\mathcal{A}\mu}\, \mathcal{U}\, d\mu = \mathcal{P}^{\mathcal{R}}(r), \qquad \text{with } \mu = r - t. \tag{3.14}$$

The computation of the homogeneous and inhomogeneous reachable sets according to (3.13) and (3.14) are presented for the case that $\mathcal{A}$ is bounded by interval matrices first, and for matrix zonotopes afterwards.

3.3.1. System Matrix Bounded by Interval Matrices

The first considered class of uncertain system matrices $\mathcal{A}$ are *interval matrices*. This class is denoted by $\mathscr{A}$ and is a specialized form of general uncertain matrices $\mathcal{A}$. The left and right limit of an interval matrix $\mathscr{A}$ is denoted by $\underline{A}$ and $\overline{A}$ such that $\mathscr{A} = [\underline{A}, \overline{A}]$. Each element of $\mathscr{A}$ can be changed independently within the specified intervals. A possibility to specify an interval matrix by the parameter vector $\rho \in [\underline{\rho}, \overline{\rho}]$ is to associate a parameter $\rho^{(i)}$ to each element of the system matrix A_{lm}, where $i = 1, \ldots, n^2$. The index of the parameters is in parentheses in order to avoid confusion with the power of a variable.

Reachable Sets of Systems without Input

For the reachable set of the homogeneous solution, (3.13) has to be evaluated. Two problems arise. First, the matrix exponential of an interval matrix $e^{\mathscr{A}r} = \{e^{Ar} | A \in \mathscr{A}, r \in \mathbb{R}^+\}$ has to be computed. Second, the result has to be multiplied with the zonotope $\mathcal{X}^0$.

Interval Matrix Exponential The first problem of computing an interval matrix exponential is tackled by its Taylor series; see (3.2):

$$e^{\mathscr{A}t} \subset \sum_{i=0}^{\eta} \frac{1}{i!} (\mathscr{A}t)^i + \mathscr{E}(t), \tag{3.15}$$

In order to obtain the exponential matrix according to the Taylor series, one has to compute the power of interval matrices and the addition of interval matrices. The interval matrix addition and the power of interval matrices can be computed straightforwardly by interval arithmetics using the addition and multiplication rule in (2.3). However, the important question is if the computation of the interval matrix exponential can be formulated as a single use expression as discussed in Sec. 2.6.

Unfortunately this is not possible as the power of an interval matrix already cannot be formulated as a single use expression in general. This has been recently investigated in [97], where also the multiplication of different interval matrices has been considered. The simplest case of computing $\mathscr{C} = \mathscr{M} \cdot \mathscr{N}$, where $\mathscr{M} \in \mathcal{I}^{l \times m}$ and $\mathscr{N} \in \mathcal{I}^{m \times l}$, is done elementwise so that $\mathscr{C}_{ij} = \sum_{k=1}^{m} \mathscr{M}_{ik} \mathscr{N}_{kj}$, which results in single use expressions (SUEs). Thus, the result is exact when applying interval arithmetics. The square of a matrix $\mathscr{C} = \mathscr{M}^2$ can be formulated as a single use expression, too:

$$\begin{aligned} \forall i \neq j : \mathcal{C}_{ij} &= \mathcal{M}_{ij} \left(\mathcal{M}_{ii} + \mathcal{M}_{jj}\right) + \sum_{k:k\neq i, k\neq j} \mathcal{M}_{ik}\, \mathcal{M}_{kj} \\ \forall i : \mathcal{C}_{ii} &= \mathcal{M}_{ii}^2 + \sum_{k:k\neq i} \mathcal{M}_{ik} \mathcal{M}_{ki} \end{aligned} \tag{3.16}$$

In contrast to these two examples, the multiplication of three interval matrices ($\mathcal{M} \cdot \mathcal{N} \cdot \mathcal{Q}$) and the cube of a matrix ($\mathcal{N}^3$) cannot be written as a single use expression. It is noteworthy that the exact computation of these two problems is NP-hard; see [97]. It is also remarked that matrix multiplication is not associative when using interval arithmetics: $(\mathcal{M} \cdot \mathcal{M})\mathcal{M} \neq \mathcal{M}(\mathcal{M} \cdot \mathcal{M})$.

From this follows that the exact computation of the exponential of an interval matrix (up to the remainder $\mathcal{E}(t)$) is also NP-hard. In order to obtain a feasible but still tight over-approximation of the interval matrix exponential, the idea of computing the square of a matrix exactly is further developed.

Lemma 3.1 (Exact Computation of $\mathcal{A}t + \frac{1}{2}\mathcal{A}^2 t^2$): The expression $\mathcal{W}(t) = \mathcal{A}t + \frac{1}{2}\mathcal{A}^2 t^2$ can be exactly computed by the following procedure using interval arithmetics:

$$\begin{aligned} \forall i \neq j : \mathcal{W}_{ij} =& \mathcal{A}_{ij}(t + \frac{1}{2}(\mathcal{A}_{ii} + \mathcal{A}_{jj})t^2) + \frac{1}{2} \sum_{k:k\neq i, k\neq j} \mathcal{A}_{ik} \mathcal{A}_{kj} t^2 \\ \forall i : \mathcal{W}_{ii} =& \Big[\kappa(\mathcal{A}_{ii}, t), \max\big(\{\underline{A}_{ii} t + \frac{1}{2}\underline{A}_{ii}^2 t^2, \overline{A}_{ii} t + \frac{1}{2}\overline{A}_{ii}^2 t^2\}\big)\Big] + \frac{1}{2} \sum_{k:k\neq i} \mathcal{A}_{ik} \mathcal{A}_{ki} t^2 \\ \kappa(\mathcal{A}_{ii}, t) =& \begin{cases} \min\big(\{\underline{A}_{ii} t + \frac{1}{2}\underline{A}_{ii}^2 t^2, \overline{A}_{ii} t + \frac{1}{2}\overline{A}_{ii}^2 t^2\}\big), & -\frac{1}{t} \notin \mathcal{A}_{ii} \\ -\frac{1}{2}, & -\frac{1}{t} \in \mathcal{A}_{ii} \end{cases} \end{aligned} \tag{3.17}$$

□

Proof: As one can reformulate the computation of the non-diagonal elements $\mathcal{W}_{ij}$ to a single-use expression (SUE) as presented in (3.17), the computation of $\mathcal{W}_{ij}$ with interval arithmetics is exact. The computation of the diagonal elements $\mathcal{W}_{ii}$ cannot be reformulated to a SUE. However, one can split $\mathcal{W}_{ii}$ into a part with and without a single variable occurrence:

$$\mathcal{W}_{ii} = \underbrace{\mathcal{A}_{ii} t + \frac{1}{2}\mathcal{A}_{ii}^2 t^2}_{non-SUE} + \underbrace{\frac{1}{2} \sum_{k:k\neq i} \mathcal{A}_{ik} \mathcal{A}_{ki} t^2}_{SUE} .$$

It remains to obtain the exact interval of $\gamma(\mathcal{A}_{ii}) := \mathcal{A}_{ii} t + \frac{1}{2}\mathcal{A}_{ii}^2 t^2$ by computing the minimum and maximum. The function $\gamma(\mathcal{A}_{ii})$ has only one minimum at $\mathcal{A}_{ii} = -1/t$ and is monotone elsewhere, so that the maximum has to be at the borders: $\gamma_{max} = \max(\{\underline{A}_{ii} t + \frac{1}{2}\underline{A}_{ii}^2 t^2, \overline{A}_{ii} t + \frac{1}{2}\overline{A}_{ii}^2 t^2\})$. Where the global minimum ($a_{min} = -1/t$) is an element of $\mathcal{A}_{ii}$, one obtains: $\gamma_{min} = -1/2$. In the other case, the minimum is computed as $\gamma_{min} = \min(\{\underline{A}_{ii} t + \frac{1}{2}\underline{A}_{ii}^2 t^2, \overline{A}_{ii} t + \frac{1}{2}\overline{A}_{ii}^2 t^2\})$. □

The exact computation of the second order Taylor expansion of the interval matrix exponential $e^{\mathcal{A}t}$ allows the following over-approximation of $e^{\mathcal{A}t}$ to be formulated.

Theorem 3.2 (Over-approximation of $e^{\mathcal{A}t}$): The over-approximation of the interval matrix exponential with a η-th order Taylor expansion, denoted by $e^{\mathcal{A}t}_{[\eta]}$, where $\mathcal{A} \in \mathcal{I}^{n\times n}$ and $t \in \mathbb{R}^+$ is obtained by

$$e^{\mathcal{A}t}_{[\eta]} = I + \mathcal{W}(t) + \sum_{i=3}^{\eta} \frac{1}{i!}(\mathcal{A}t)^i + \mathcal{E}(t),$$

$$\mathcal{E}(t) = [-\mathbf{1}, \mathbf{1}] \frac{(\|\mathcal{A}\|_\infty t)^{\eta+1}}{(\eta+1)!} \frac{1}{1-\epsilon}, \quad \epsilon = \frac{\|\mathcal{A}\|_\infty t}{\eta+2} \overset{!}{<} 1$$

where $\|\mathcal{A}\|_\infty := \max(\|A\|_\infty | A \in \mathcal{A}) = \|A^*\|_\infty$ and $A^*_{ml} = \max(|\underline{A}_{ml}|, |\overline{A}_{ml}|)$. □

Proof: The interval matrix exponential is computed by a finite Taylor expansion; see (3.15). The computation up to second order ($I + \mathcal{W}$) is exact as I is not an interval matrix and the exactness of $\mathcal{W} = \mathcal{A}t + \frac{1}{2}\mathcal{A}^2t^2$ has been shown in Lemma 3.1. The remaining sums of the Taylor expansion up to order η are computed by interval arithmetics, which results in over-approximations. As the Taylor expansion is finite, it is necessary to give an overapproximation for the remainder $\mathcal{E}$, which is computed as shown in (3.3); see also [110]. It is left to show that $\|\mathcal{A}\|_\infty = \|A^*\|_\infty$, which becomes obvious by looking at the definition of the matrix infinity norm: $\|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |A_{ij}|$. □

The quality of the over-approximation is illustrated by a simple example.

Example 3.1 (Over-Approximation of the Interval Matrix Exponential): Given is the following interval matrix $\mathcal{A}$:

$$\mathcal{A} = [\underline{A}, \overline{A}] = \begin{bmatrix} [-1.1, -0.9] & [-4.1, -3.9] \\ [3.9, 4.1] & [-1.1, -0.9] \end{bmatrix}, \quad t = 0.04$$

The result of $e^{\mathcal{A}t}_{[4]}$ ($\eta = 4$) can be compared to the exemplary and exact results of $e^{\underline{A}t}$ and $e^{\overline{A}t}$:

$$e^{\mathcal{A}t}_{[4]} = \begin{bmatrix} [0.94396, 0.95309] & [-0.15765, -0.14852] \\ [0.14852, 0.15765] & [0.94396, 0.95309] \end{bmatrix},$$

$$e^{\underline{A}t} = \begin{bmatrix} 0.94474 & -0.15627 \\ 0.14865 & 0.94474 \end{bmatrix}, \quad e^{\overline{A}t} = \begin{bmatrix} 0.95233 & -0.14984 \\ 0.15753 & 0.95233 \end{bmatrix}$$

□

Interval Matrix Multiplication with Zonotopes The previously computed over-approximation of the interval matrix exponential has to be multiplied by the initial set to compute the reachable set of the homogeneous solution: $\mathcal{H}^{\mathcal{R}}(r) = e^{\mathcal{A}r}\mathcal{X}^0$.

The multiplication of a matrix with a zonotope has already been presented in (2.1). In order to formulate an extension to interval matrices, the following two lemmas are introduced first.

Lemma 3.2 (Symmetric Interval Matrix Multiplication with Matrices): The interval matrix obtained after the multiplication of a regular matrix M with a symmetric interval matrix $\mathcal{S} = [-\overline{S}, \overline{S}]$ (both with proper dimension) can be exactly computed by

$$M\mathcal{S} = [-|M|\overline{S}, |M|\overline{S}], \quad \mathcal{S}M = [-\overline{S}|M|, \overline{S}|M|]. \tag{3.18}$$

The absolute value is applied elementwise. □

Proof: Without loss of generality, the multiplication $M\mathcal{S}$ is proven for the multiplication of $m \in \mathbb{R}^{1\times n}$ and $\mathcal{s} = [-\overline{s}, \overline{s}] \in \mathcal{I}^{n\times 1}$. The maximum value of $m\mathcal{s}$ is obtained by selecting $\mathcal{s}_i = \texttt{sign}(m_i)\overline{s}_i, \forall i = 1..n$ so that $\max(m\mathcal{s}) = \sum_{i=1}^{n} |m_i|\overline{s}_i$. Analogously, the minimum is obtained by choosing $\mathcal{s}_i = -\texttt{sign}(m_i)\overline{s}_i$ so that $\min(m\mathcal{s}) = -\sum_{i=1}^{n} |m_i|\overline{s}_i$. □

Remark 3.1 (Tight Interval Enclosure): It is noted that the resulting interval matrix of Lemma 3.2 is exact; however, the resulting set is not necessarily an interval matrix. E.g. when $\mathcal{S} = [-1, 1]$ and $M = g \in \mathbb{R}^n$, the exact result of $\hat{l} = [-1, 1]g$ is a line from $-g$ to g, while the result of Lemma 3.2 is a multidimensional interval which exactly encloses the line $\hat{l}$. The reason for the exact enclosure is that each element of $\hat{l}$ is computed by a SUE: $\hat{l}_i = [-1, 1]g_i$. Thereby, the interval $[-1, 1]$ is assumed to be independent for each component of $\hat{l}_i$, although it is not. □

The foregoing lemma is extended to the case when a zonotope is multiplied by an interval matrix.

Lemma 3.3 (Symmetric Interval Matrix Multiplication with Zonotopes): The linear map of a zonotope $\mathcal{Z}$ specified by a symmetric interval matrix $\mathcal{S} = [-\overline{S}, \overline{S}]$ is over-approximated by a hyperrectangle with center 0:

$$\begin{aligned}
\mathcal{S}\mathcal{Z} &= \Big\{x \in \mathbb{R}^n \Big| x = [-\overline{S}, \overline{S}]c + \sum_{i=1}^{e} \beta^{(i)}[-\overline{S}, \overline{S}]g^{(i)}, \quad -1 \leq \beta^{(i)} \leq 1\Big\} \\
&= (0, f^{(1)}, \dots, f^{(n)}) \\
f_j^{(i)} &= \begin{cases} \overline{S}_j(|c| + \sum_{k=1}^{e} |g|^{(k)}), \text{ for } i = j \\ 0, \text{ for } i \neq j \end{cases}
\end{aligned}$$

and the subscript j of $f_j^{(i)}$ denotes the j-th element of $f^{(i)}$ and $\overline{S}_j$ denotes the j-th row of $\overline{S}$. □

Proof: The multiplication is performed analogous to the regular matrix multiplication in (2.1) with the difference that the enclosing multidimensional intervals of $[-\overline{S}, \overline{S}]c$ and $[-\overline{S}, \overline{S}]g^{(i)}$ are computed according to Lemma 3.2:

$$[-\overline{S}, \overline{S}]c + \sum_{i=1}^{e} \underbrace{[-1, 1][-\overline{S}, \overline{S}]}_{=[-\overline{S}, \overline{S}]} g^{(i)} = [-\overline{S}(|c| + \sum_{i=1}^{e} |g^{(i)}|), \overline{S}(|c| + \sum_{i=1}^{e} |g^{(i)}|)].$$

The obtained multidimensional intervals are transformed to zonotope notation according to Prop. 2.1 resulting in the generators $f^{(i)}$. □

Remark 3.2 (Over-approximation of Lemma 3.3): The computations $[-\overline{S}, \overline{S}]c$ and $[-1, 1][-\overline{S}, \overline{S}]g^{(i)}$ of the foregoing proof result in a tight enclosure by multidimensional intervals as they are computed by SUEs; see Remark 3.1. The over-approximation is demonstrated by the example

$$\mathcal{S} = \mathtt{diag}([-1,1],[-1,1]) \text{ and } \mathcal{Z} = \begin{bmatrix}1\\1\end{bmatrix} + [-1,1]\begin{bmatrix}1\\1\end{bmatrix} + [-1,1]\begin{bmatrix}1\\-1\end{bmatrix}$$

which is visualized in Fig. 3.4. □

After comparing Prop. 2.2 with Lemma 3.3, one can see that the same over-approximation is obtained by $\mathcal{S}\,\mathtt{box}(\mathcal{Z})$. The multiplication of a general interval matrix with a zonotope is performed by separating the interval matrix into a regular matrix and a symmetric interval matrix:

Theorem 3.3 (Interval Matrix Multiplication with Zonotopes): The multiplication of an interval matrix $\mathcal{L} = L + \mathcal{S}$ where $\mathcal{S} = [-\overline{S}, \overline{S}]$ with a zonotope $\mathcal{Z} = (c, g^{(1)}, \dots, g^{(e)})$ is over-approximated by:

$$\mathcal{L}\mathcal{Z} = (Lc, Lg^{(1)}, \dots, Lg^{(e)}, f^{(1)}, \dots, f^{(n)})$$
$$f_j^{(i)} = \begin{cases} \overline{S}_j(|c| + \sum_{k=1}^{e} |g|^{(k)}), & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases} \qquad \square$$

Proof: The interval matrix multiplication with $\mathcal{L}$ is split up into a matrix multiplication with L and a symmetric interval matrix multiplication with $[-\overline{S}, \overline{S}]$: $\mathcal{L}\mathcal{Z} \subseteq L\mathcal{Z} + [-\overline{S}, \overline{S}]\mathcal{Z}$. Applying Lemma 3.3 and the Minkowski addition rule in (2.1) yields the final result. □

Again, it is remarked that the result of Theorem 3.3 is an over-approximation. One source of the over-approximation is Lemma 3.3 when computing $[-\overline{S}, \overline{S}]\mathcal{Z}$ and the second source is the split of $\mathcal{L}$ since already the split of a regular matrix $(C + D)\mathcal{Z} \subseteq C\,\mathcal{Z} + D\,\mathcal{Z}$ with $C, D \in \mathbb{R}^{n \times n}$ causes an over-approximation.

The over-approximation is demonstrated for a zonotope with a single generator g:

$$\mathcal{L} = \mathtt{diag}([0.5,1],[0.5,1]) \text{ and } \mathcal{Z} = \begin{bmatrix}0\\0\end{bmatrix} + [-1,1]\begin{bmatrix}1\\0.5\end{bmatrix}.$$

The exact set and the over-approximated zonotope are visualized in Fig. 3.5.

With the result of Theorem 3.2 and Theorem 3.3, the homogeneous solution for a time point r can be computed: $\mathcal{H}^{\mathcal{R}}(r) = e^{\mathcal{A}r}\mathcal{X}^0$. The reachable set for the time interval $[0, r]$ is computed analogously to LTI systems (see Sec. 3.2.1): $\mathcal{H}^{\mathcal{R}}([0, r]) = \mathtt{CH}(\mathcal{X}^0, e^{\mathcal{A}r}\mathcal{X}^0) + \mathcal{F}\mathcal{X}^0$. It remains to consider the reachable set due to the uncertain input, which is done next.

Reachable Sets of Systems with Input

The reachable set originating from the set of inputs is computed analogously to Theorem 3.1, except that the system matrix is now an interval matrix $\mathcal{A}$:

$$\mathcal{P}^{\mathcal{R}}(r) = \sum_{i=0}^{\eta} \left(\frac{\mathcal{A}^i\, r^{i+1}}{(i+1)!} \mathcal{U} \right) + \mathcal{E}(r) \cdot r \cdot \mathcal{U}.$$

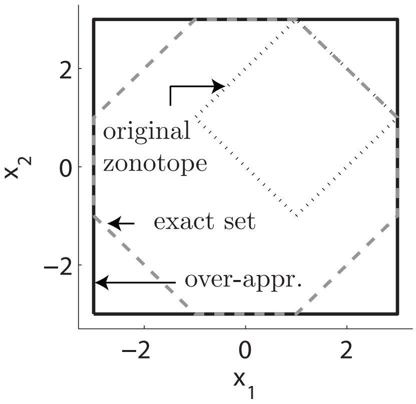


Fig. 3.4.: Over-approximation caused by Lemma 3.3.

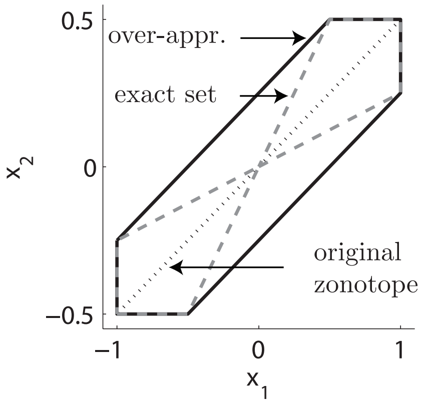


Fig. 3.5.: Over-approximation caused by Theorem 3.3.

It is remarked that $\mathcal{A}^2$ can be computed exactly, while the higher powers suffer from the dependency effect of interval arithmetics. The computation of the input solution differs from the one presented in [180], which is only valid if the input is constant within the time intervals $[kr, (k+1)r]$.

In analogy to linear systems without uncertain parameters, the case when the input set $\mathcal{U}$ does not contain the origin has to be considered by the additional correction term $\mathcal{A}^{-1}\,\mathcal{F}\,\tilde{u} = \tilde{\mathcal{F}}\,\tilde{u}$. However, this scheme poses the problem that the computation of the inverse of the system matrix by interval arithmetics introduces additional uncertainties while the exact computation of the inverse interval matrix is NP-hard [44]. This problem can be circumvented by computing $\tilde{\mathcal{F}}$ from its Taylor expansion. Another advantage of the Taylor expansion is that it can also be applied to regular system matrices whose inverse does not exist.

Proposition 3.4 (Input Correction Matrix $\tilde{\mathcal{F}}$): The interval matrix $\tilde{\mathcal{F}}$ which determines the enlargement $\tilde{\mathcal{F}}\,\tilde{u}$ in (3.9) can be computed as:

$$\tilde{\mathcal{F}} \subseteq \left[\sum_{i=2}^{\eta}[(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}})r^i, 0]\frac{\mathcal{A}^{i-1}}{i!}\right] + \frac{\mathcal{E}(r)}{\|\mathcal{A}\|_\infty}, \quad t \in [0, r].$$

where $\mathcal{E}(r)$ is the matrix exponential remainder and η the number of Taylor terms according to (3.3). The computations are performed using interval arithmetics. □

The proof is omitted since Prop. 3.4 follows directly when inserting the computation of $\mathcal{F}$ from Prop. 3.1 into $\tilde{\mathcal{F}} = \mathcal{A}^{-1}\mathcal{F}$ and substituting A by $\mathcal{A}$ and considering that $\mathcal{A}^{-1}\mathcal{E} = \mathcal{E}/\|\mathcal{A}\|_\infty$ since $\mathcal{E}$ is computed using the infinity norm. Next, the algorithmic realization of the presented extension is discussed.

Algorithmic Realization

In principle, the reachable sets of linear systems with or without uncertain parameters are computed the same way. Besides the basic computation routine in (3.8), called Method B below, the alternative Method A is also thinkable:

$$
\begin{array}{r|ll}
\text{Meth. } A\text{:} & e^{\mathcal{A}\,kr} & = e^{\mathcal{A}\,r}\, e^{\mathcal{A}\,(k-1)r} \\
& \mathcal{H}^{\mathcal{R}}([kr,(k+1)r]) & = e^{\mathcal{A}\,kr}\mathcal{H}^{\mathcal{R}}([0,r]) \\
& \mathcal{P}^{\mathcal{R}}([kr,(k+1)r]) & = e^{\mathcal{A}\,kr}\mathcal{P}^{\mathcal{R}}(r) + \ldots + e^{\mathcal{A}\,r}\mathcal{P}^{\mathcal{R}}(r) + \mathcal{P}^{\mathcal{R}}(r) \\
& & \\
\text{Meth. } B\text{:} & \mathcal{H}^{\mathcal{R}}([kr,(k+1)r]) & = e^{\mathcal{A}r}\mathcal{H}^{\mathcal{R}}([(k-1)r,kr]) \\
& \mathcal{P}^{\mathcal{R}}([kr,(k+1)r]) & = e^{\mathcal{A}r}\mathcal{P}^{\mathcal{R}}([(k-1)r,kr]) + \mathcal{P}^{\mathcal{R}}(r).
\end{array}
\tag{3.19}
$$

Method A computes the interval matrices $e^{\mathcal{A}\,kr}$ first and multiplies them with the sets $\mathcal{H}^{\mathcal{R}}([0,r])$ and $\mathcal{P}^{\mathcal{R}}(r)$ in the end. In contrast to this technique, Method B computes intermediate reachable sets $\mathcal{H}^{\mathcal{R}}([(k-1)r,kr])$ and $\mathcal{P}^{\mathcal{R}}([(k-1)r,kr])$.

For linear systems without uncertain parameters, both techniques compute the same results. The situation is completely different for linear systems with uncertain parameters, where both methods suffer under the wrapping effect. In this case, Method B produces significantly better results, as shown below.

The computation of the homogeneous reachable set $\mathcal{H}^{\mathcal{R}}([(k-1)r,kr])$ is examined first. Applying Method A, the homogeneous reachable set of the second time interval is computed as $\mathcal{H}^{\mathcal{R}}([r,2r]) = e^{\mathcal{A}r}\mathcal{H}^{\mathcal{R}}([0,r]) = M\,\mathcal{H}^{\mathcal{R}}([0,r]) + \mathcal{D}\,\mathtt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$ according to Theorem 3.3, where $e^{\mathcal{A}r}$ is split into the regular matrix M and the symmetric interval matrix $\mathcal{D}$. Further, the previously mentioned equality of $\mathcal{D}\,\mathcal{H}^{\mathcal{R}}([0,r])$ and $\mathcal{D}\,\mathtt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$ when applying Lemma 3.3 is used. This scheme yields for the third time interval

$$
\begin{aligned}
\mathcal{H}^{\mathcal{R}}([2r,3r]) &= e^{\mathcal{A}r^2}\,\mathcal{H}^{\mathcal{R}}([0,r]) = (M+\mathcal{C})^2\,\mathcal{H}^{\mathcal{R}}([0,r]) \\
&= M^2\,\mathcal{H}^{\mathcal{R}}([0,r]) + (M\mathcal{C} + \mathcal{C}M + \mathcal{C}^2)\mathtt{box}(\mathcal{H}^{\mathcal{R}}([0,r])).
\end{aligned}
$$

Note that $M\mathcal{C}$, $\mathcal{C}M$ and $\mathcal{C}^2$ are symmetric interval matrices which follows directly from Lemma 3.2. The result of this procedure for different time steps is summarized as follows:

Method A.	$\mathcal{H}^{\mathcal{R}} = \mathcal{Z} + \mathcal{I}$	
Time step k	Zonotope $\mathcal{Z}$	Multidimensional Interval $\mathcal{I}$
1	$M\,\mathcal{H}^{\mathcal{R}}([0,r])$	$\mathcal{D}\,\mathtt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$
2	$M^2\,\mathcal{H}^{\mathcal{R}}([0,r])$	$(M\mathcal{C} + \mathcal{C}M + \mathcal{C}^2)\mathtt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$
$\vdots$	$\vdots$	$\vdots$
k	$M^k\,\mathcal{H}^{\mathcal{R}}([0,r])$	$((M+\mathcal{D})^k - M^k)\mathtt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$

It can be observed that for a stable exponential matrix M ($|\hat{\lambda}_i(M)| < 1, \hat{\lambda}_i$: eigenvalues) the zonotopial part $\mathcal{Z}$ of $\mathcal{H}^{\mathcal{R}}$ converges to the origin, such that the reachable set is over-approximated by a multidimensional interval in the limit, which causes a large over-approximation.

The alternative Method B does not require the reachable set to be over-approximated as much by multidimensional intervals:

Method B.	$\mathcal{H}^{\mathcal{R}} = \mathcal{Z} + \mathcal{I}$	
Time step k	Zonotope $\mathcal{Z}$	Multidimensional Interval $\mathcal{I}$
1	$\mathcal{Z}(1) = M\,\mathcal{H}^{\mathcal{R}}([0,r])$	$\mathcal{I}(1) = \mathcal{D}\,\texttt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$
2	$\mathcal{Z}(2) = M^2\,\mathcal{H}^{\mathcal{R}}([0,r])$ $+M\,\mathcal{D}\,\texttt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$	$\mathcal{I}(2) = \mathcal{C}\,\texttt{box}(M\,\mathcal{H}^{\mathcal{R}}([0,r]))$ $+\mathcal{C}^2\,\texttt{box}(\mathcal{H}^{\mathcal{R}}([0,r]))$
$\vdots$	$\vdots$	$\vdots$
k	$\sum_{i=1}^{k} M^i\,\mathcal{I}(k-i)$	$\sum_{i=1}^{k} \mathcal{C}^i\,\texttt{box}(\mathcal{Z}(k-i))$

The same conclusion can be drawn for the reachable set $\mathcal{P}^{\mathcal{R}}$ of the input solution so that Method B is applied in the following computations. Combining $\mathcal{H}^{\mathcal{R}}$ and $\mathcal{P}^{\mathcal{R}}$ of Method B to $\mathcal{R} = \mathcal{H}^{\mathcal{R}} + \mathcal{P}^{\mathcal{R}}$ yields

$$\mathcal{R}([kr,(k+1)r]) = e^{\mathcal{A}r}\mathcal{R}([(k-1)r,kr]) + \mathcal{P}^{\mathcal{R}}(r).$$

This result and the remaining steps to compute the reachable set are formulated in Alg. 4. The reduction operation `reduce()` in Alg. 4 is applied since the zonotope order is constantly increasing due to the multiplication with $= e^{\mathcal{A}r}$ and the addition of $\mathcal{P}_0^{\mathcal{R}}$. The reduction is performed according to Heuristic 2.4. The interval matrix $\mathcal{C}(r) := \int_0^r e^{\mathcal{A}\,t}$ is computed as $\mathcal{C}(r) = \sum_{i=0}^{\eta} \frac{1}{(i+1)!}\mathcal{A}^i r^{i+1} + \mathcal{E}(r)r$ $(\hat{=}\mathcal{A}^{-1}(e^{\mathcal{A}r} - I))$ for which the terms up to second order can be exactly computed in analogy to Lemma 3.1.

Algorithm 4 Compute $\mathcal{R}([0,t_f])$

Input: Initial set $\mathcal{X}^0$, interval matrix exponential $e^{\mathcal{A}r}$, $\mathcal{C}(r) := \int_0^r e^{\mathcal{A}\,t}\,dt$, input set $\mathcal{U}$, correction matrices $\mathcal{F}$, $\tilde{\mathcal{F}}$, time horizon t_f

Output: $\mathcal{R}([0,t_f])$

```
$\tilde{\mathcal{P}}(r) = \mathcal{C}(r)\tilde{u}$
$\tilde{\mathcal{R}}_0 = \mathtt{CH}(\mathcal{X}^0, e^{\mathcal{A}r}\mathcal{X}^0 + \tilde{\mathcal{P}}(r)) - \tilde{\mathcal{P}}(r) + \mathcal{F}\mathcal{X}^0 + \tilde{\mathcal{F}}\,\tilde{u}$
$\mathcal{P}_0^{\mathcal{R}} = \sum_{i=0}^{\eta}\left(\frac{\mathcal{A}^i\,r^{i+1}}{(i+1)!}\mathcal{U}\right) + \mathcal{E}(r)\cdot r\cdot\mathcal{U}$
$\mathcal{R}_0 = \tilde{\mathcal{R}}_0 + \mathcal{P}_0^{\mathcal{R}}$
for $k = 1 \dots t_f/r - 1$ do
  $\mathcal{R}_k = e^{\mathcal{A}r}\mathcal{R}_{k-1} + \mathcal{P}_0^{\mathcal{R}}$
  $\mathcal{R}_k = \mathtt{reduce}(\mathcal{R}_k)$
end for
$\mathcal{R}([0,t_f]) = \bigcup_{k=1}^{t_f/r}\mathcal{R}_{k-1}$
```

In contrast to the algorithms for linear systems without parameter uncertainties, this algorithm is not free of the wrapping effect. The over-approximations of each time step occur due to

- the over-approximative computation of $e^{\mathcal{A}r}$;
- the over-approximative multiplication of interval matrices with zonotopes ($e^{\mathcal{A}r}\,\mathcal{R}([(k-1)r,kr])$);

- the order reduction of zonotopes by `reduce`().

Finally, some numerical results for linear systems specified by interval matrices are presented.

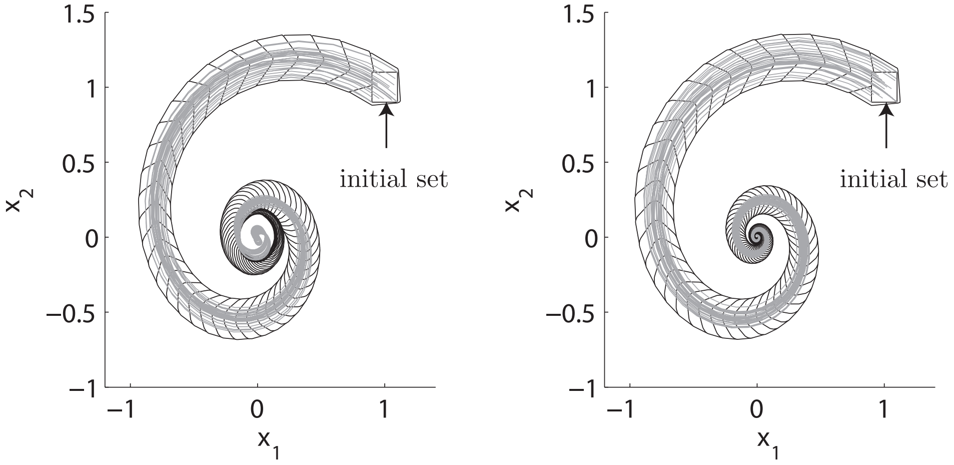
Numerical Examples

The proposed Alg. 4 is tested with the same numerical examples as for linear systems in Sec. 3.2.3, except that uncertainties are added to the system matrix and the input uncertainties are changed. The two-dimensional example including uncertain values of the system matrix is given as

$$\begin{aligned}&\dot{x} = A\,x + \begin{bmatrix}1\\1\end{bmatrix} u(t),\\ &A \in \begin{bmatrix}[-1.05,-0.95] & [-4.05,-3.95]\\ [3.95,4.05] & [-1.05,-0.95]\end{bmatrix},\ x(0)\in\begin{bmatrix}[0.9,1.1]\\ [0.9,1.1]\end{bmatrix},\ u(t)\in[-0.1,0.1].\end{aligned} \tag{3.20}$$

The reachable set is computed according to Alg. 4. In analogy to the previous examples, a time horizon of $t_f = 5$ and a time step size of $r = 0.04$ has been used, which results in 125 iterations. The order of the zonotopes has been restricted to $\hat{\varrho} = 10$.

Because Alg. 4 is not free of the wrapping effect, it is demonstrated in Fig. 3.6(a) how much the reachable set over-approximates when there are no parameter uncertainties[2]. The situation where there are uncertain parameters, but no uncertain inputs, is shown in Fig. 3.6(b). The reachable set for both, uncertain parameters and inputs can be seen in Fig. 3.7(a). To demonstrate how dominant the effect on the maximum order of the zonotopes is, the reachable set was computed with zonotopes of order $\hat{\varrho} = 20$ in Fig. 3.7(b). Beyond this order, the result did not improve considerably for this example.

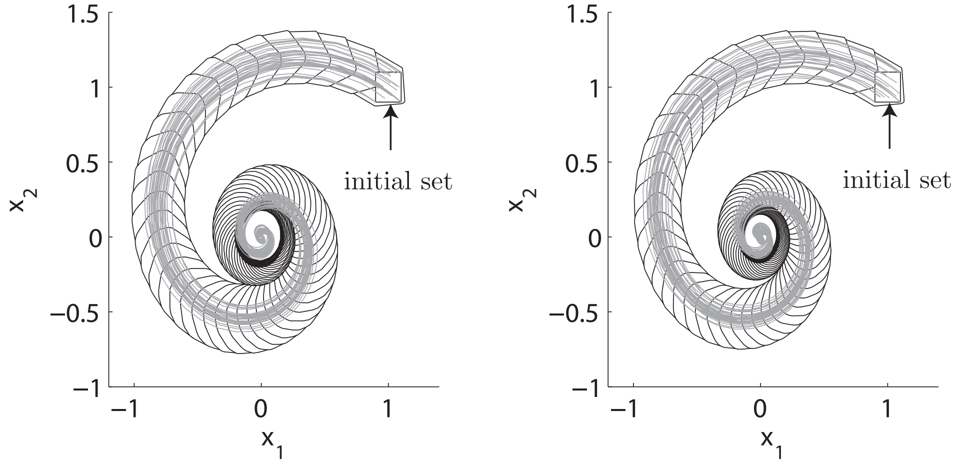


(a) No uncertain parameters.

(b) No input.

Fig. 3.6.: Reachable sets of the two-dimensional example.

[2] The values of the system matrix are chosen as the centers of the intervals



(a) Zonotope order: 10.

(b) Zonotope order: 20.

Fig. 3.7.: Reachable sets of the two-dimensional example – different zonotope order.

The previously used five-dimensional example with changed input uncertainty and uncertainties in the system matrix is $\dot{x} = A\,x + u(t)$, where

$$A \in \begin{bmatrix} [-1.05,-0.95] & [-4.05,-3.95] & 0 & 0 & 0 \\ [3.95,4.05] & [-1.05,-0.95] & 0 & 0 & 0 \\ 0 & 0 & [-3.2,-2.8] & [0.8,1.2] & 0 \\ 0 & 0 & [-1.2,-0.8] & [-3.2,-2.8] & 0 \\ 0 & 0 & 0 & 0 & [-2.2,-1.8] \end{bmatrix},$$

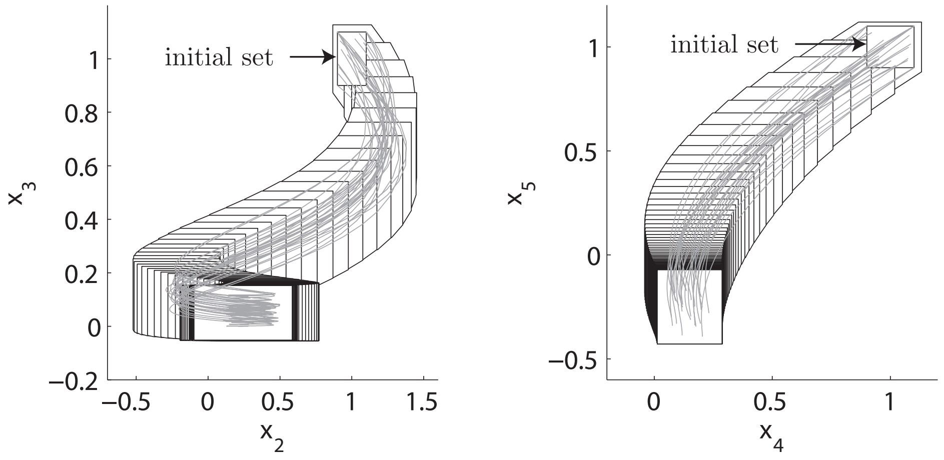
$$x(0) \in \begin{bmatrix} [0.9,1.1] \\ [0.9,1.1] \\ [0.9,1.1] \\ [0.9,1.1] \\ [0.9,1.1] \end{bmatrix}, \; u(t) \in \begin{bmatrix} [0.9,1.1] \\ [-0.25,0.25] \\ [-0.1,0.1] \\ [0.25,0.75] \\ [-0.75,-0.25] \end{bmatrix}.$$

The five-dimensional example is also computed with a time horizon of $t_f = 5$, a time step size of $r = 0.04$ and a zonotope order of $\hat{\varrho} = 10$. The result is displayed in Fig. 3.8 for two different projections on two dimensions.

The scalability of Alg. 4 is demonstrated by listing the computation times for randomly generated examples of higher dimension in Tab. 3.2. All reachable sets were computed with 125 iterations. The computations were performed with Matlab on a single core desktop PC with an AMD Athlon64 3700+ processor. Note that the computation times for dimensions 5, 10, and 20 differ only marginally due to the overhead time caused by function calls.

Tab. 3.2.: Computational times.

Dimension n	5	10	20	50	100
CPU-time [s]	0.14	0.20	0.35	1.72	7.96



(a) Projection onto x_2, x_3.

(b) Projection onto x_4, x_5.

Fig. 3.8.: Reachable sets of the five-dimensional example.

3.3.2. System Matrix Bounded by Matrix Zonotopes

The second considered class of uncertain system matrices $\mathcal{A}$ are *matrix zonotopes*. This class is denoted by $\mathcal{A}^{\mathcal{Z}}$ and is a specialized form of general uncertain matrices $\mathcal{A}$. Unlike the previously presented interval matrices, which are best specified by their left and right limit, matrix zonotopes are composed of real-valued matrices $\hat{A}^{(i)}$. Together with the value of the parameter vector $\hat{\rho}$, the matrices $\hat{A}^{(i)}$ determine the value of the system matrix $A(\hat{\rho})$:

$$A(\hat{\rho}) = \sum_{i=1}^{\kappa} \hat{\rho}_i \, \hat{A}^{(i)}, \quad \hat{\rho}_i \in [\underline{\hat{\rho}}_i, \overline{\hat{\rho}}_i], \, \hat{A}^{(i)} \in \mathbb{R}^{n \times n}. \tag{3.21}$$

Here, the normalized version of (3.21) is used, which is

$$A(\rho) = \hat{A}^{(0)} + \sum_{i=1}^{\kappa} \rho_i \hat{A}^{(i)}, \quad \rho_i \in [-1, 1]. \tag{3.22}$$

The formulas to obtain the normalized matrices $\hat{A}^{(i)}$ from the unnormalized ones are

$$\hat{A}^{(0)} = 0.5 \sum_{i=1}^{\kappa} (\overline{\hat{\rho}}_i + \underline{\hat{\rho}}_i) \hat{A}^{(i)}, \quad \hat{A}^{(i)} = 0.5(\overline{\hat{p}}^{(i)} - \underline{\hat{p}}^{(i)}) \hat{A}^{(i)}.$$

The structure for the uncertain matrix $A(\rho)$ in (3.22) has also been used in a work on stability of uncertain linear systems [37]. There, the possible set of matrices $\mathcal{A}$ is called a *matrix hypercube*, whereas in this work the term *matrix zonotope* is preferred. The reason for the use of the latter notation is the similarity to the specification of zonotopes in Def. 2.3, which is also reflected by a superscripted $\mathcal{Z}$ in the notation of a matrix zonotope $\mathcal{A}^{\mathcal{Z}}$.

Due to the analogy to zonotopes, the matrix $\hat{A}^{(0)}$ is called the *center matrix* and the matrices $\hat{A}^{(i)}$, $i = 1 \dots \kappa$ are called *generator matrices*. The extension of the proposed method to the case when the parameters are additionally restricted to $\sum \rho_i = 1$, yielding a *matrix polytope*, is presented later in Sec. 3.3.3.

The proposed structure of the uncertainty in (3.22) is a generalization of the previously introduced interval matrices, which can be brought into the form for matrix zonotopes by e.g. introducing n^2 ($n \hat{=}$dimension) matrices $\hat{A}^{(i)}$ containing only a single non-zero entry:

Example 3.2 (Convert an Interval Matrix to a Matrix Zonotope):

$$\mathcal{A} = \hat{A}^{(0)} + \sum_{i=1}^{\kappa} \rho_i \hat{A}^{(i)} = \begin{bmatrix} [-1.1, -0.9] & [-4.1, -3.9] \\ [3.9, 4.1] & [-1.1, -0.9] \end{bmatrix}$$

$$\hat{A}^{(0)} = \begin{bmatrix} -1 & -4 \\ 4 & -1 \end{bmatrix}, \hat{A}^{(1)} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0 \end{bmatrix}, \hat{A}^{(2)} = \begin{bmatrix} 0 & 0.1 \\ 0 & 0 \end{bmatrix}, \hat{A}^{(3)} = \begin{bmatrix} 0 & 0 \\ 0.1 & 0 \end{bmatrix}, \hat{A}^{(4)} = \begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

□

The opposite problem of finding a tight enclosure of a matrix zonotope by an interval matrix is addressed in the next example.

Example 3.3 (Over-approximate a Matrix Zonotope by an Interval Matrix):

$$\mathcal{A}^{\mathcal{Z}} = \hat{A}^{(0)} + \rho_1 \hat{A}^{(1)} \subset \begin{bmatrix} [-1.1, -0.9] & [-4.1, -3.9] \\ [3.9, 4.1] & [-1.1, -0.9] \end{bmatrix}$$

$$\hat{A}^{(0)} = \begin{bmatrix} -1 & -4 \\ 4 & -1 \end{bmatrix}, \hat{A}^{(1)} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}.$$

□

The computation of reachable sets when the system matrix is uncertain within a matrix zonotope is presented next.

Reachable Sets of Systems without Input

Analogously to the problem of computing the reachable set of the homogeneous solution $\mathcal{H}^{\mathcal{R}}(r) = e^{\mathcal{A}r}\mathcal{X}^0$ when $\mathcal{A}$ is an interval matrix, a solution has to be found when $\mathcal{A}$ is a matrix zonotope. For interval matrices, this task has been split into two subproblems: First, the matrix exponential of the interval matrix is computed, and second, the result is multiplied with a zonotope. In contrast to this, these two steps are interleaved in the approach for matrix zonotopes.

As for the interval matrix approach, the multiplication of the matrix exponential $e^{\mathcal{A}^{\mathcal{Z}}}$ with a zonotope $\mathcal{Z}$ is based on the Taylor series of the matrix exponential. For this, the Taylor expansion is split into a part $\hat{\mathcal{L}}^{\mathcal{Z}}$ which is linear in the parameter vector ρ and a part $\mathcal{N}^{\mathcal{Z}}$ which is nonlinear in the parameter vector:

$$\begin{aligned} e^{\mathcal{A}^{\mathcal{Z}}t}\mathcal{Z} &= \Big(I + \mathcal{A}^{\mathcal{Z}}t + \frac{1}{2!}(\mathcal{A}^{\mathcal{Z}}t)^2 + \frac{1}{3!}(\mathcal{A}^{\mathcal{Z}}t)^3 + \dots + \mathcal{E}(t)\Big)\mathcal{Z} \\ &\subseteq \underbrace{\Big(I + \mathcal{A}^{\mathcal{Z}}t\Big)}_{\hat{\mathcal{L}}^{\mathcal{Z}}}\mathcal{Z} + \underbrace{\Big(\frac{1}{2!}(\mathcal{A}^{\mathcal{Z}}t)^2 + \frac{1}{3!}(\mathcal{A}^{\mathcal{Z}}t)^3 + \dots + \mathcal{E}(t)\Big)}_{\mathcal{N}^{\mathcal{Z}}}\mathcal{Z}. \end{aligned} \tag{3.23}$$

The first part $\hat{\mathcal{L}}^{\mathcal{Z}}\,\mathcal{Z}$ is computed such that it produces the tightest convex hull of the exact result. The second part $\mathcal{N}^{\mathcal{Z}}\,\mathcal{Z}$ is computed by over-approximating $\mathcal{A}^{\mathcal{Z}}$ with an interval matrix $\mathcal{A} \supseteq \mathcal{A}^{\mathcal{Z}}$ so that interval arithmetics can be applied. The interval matrix $\mathcal{A}$ is simply obtained by applying interval arithmetics to (3.22): $\mathcal{A} = \hat{A}^{(0)} + \sum_{i=1}^{\kappa}[-1,1] \cdot \hat{A}^{(i)}$. Although this procedure causes an over-approximation, the lost accuracy is acceptable as the intervals of the higher order terms are small, compared to the uncertain values in the linear part $\hat{\mathcal{L}}^{\mathcal{Z}}$ when using typical values for the time increment r. Next, the computation of $\mathcal{N}^{\mathcal{Z}}$ is rewritten to

$$\mathcal{N}^{\mathcal{Z}}(t) = (\mathcal{A}t)^2 \sum_{i=2}^{\eta} \frac{1}{i!}(\mathcal{A}t)^{i-2} + \mathcal{E}(t),$$

where $\eta \geq 2$ is the number of considered Taylor terms. The reason for factoring out $(\mathcal{A}t)^2$ is that the square of an interval matrix can be computed exactly as mentioned in Sec. 3.3.1 or [97], resulting in a tighter over-approximation. The remainder $\mathcal{E}(t)$ is computed as shown in Theorem 3.2.

In a further step, the interval matrix $\mathcal{N}^{\mathcal{Z}}$ is split into a real valued matrix $N \in \mathbb{R}^{n\times n}$ and a symmetric interval matrix $\mathcal{S}$ ($\mathcal{S} = [-\overline{S}, \overline{S}]$) such that $\mathcal{N}^{\mathcal{Z}} = N + \mathcal{S}$. The real valued part N is added to $\hat{\mathcal{L}}^{\mathcal{Z}}$, such that (3.23) changes to

$$e^{\mathcal{A}^{\mathcal{Z}}t}\mathcal{Z} = \underbrace{\Big(I + N(t) + \mathcal{A}^{\mathcal{Z}}t\Big)}_{\mathcal{L}^{\mathcal{Z}}(t)}\,\mathcal{Z} + \mathcal{S}(t)\,\mathcal{Z}, \tag{3.24}$$

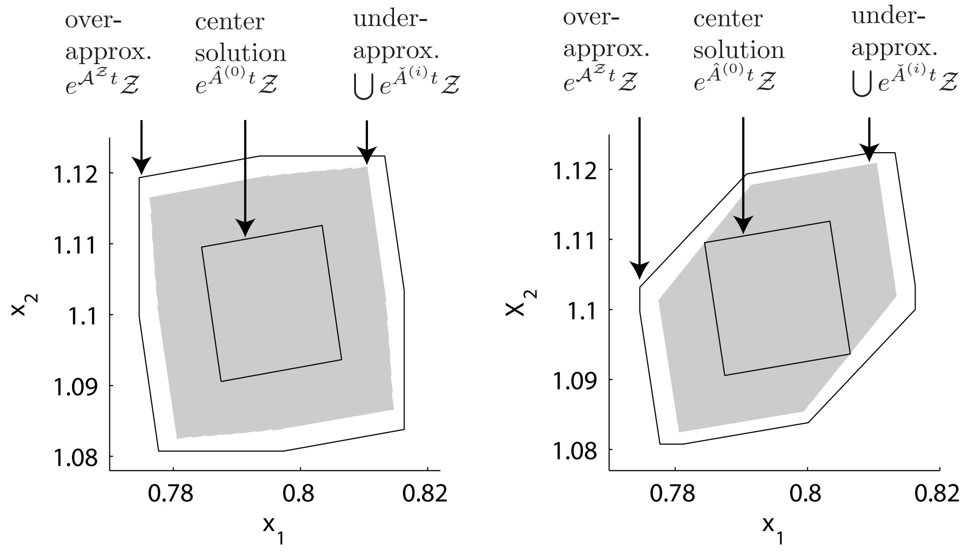
where $\mathcal{L}^{\mathcal{Z}}(t)$ is a matrix zonotope with center matrix $L^{(0)}(t) = I + N(t) + \hat{A}^{(0)}t$ and generator matrices $L^{(i)}(t) = \hat{A}^{(i)}t$. The multiplication of the symmetric interval matrix $\mathcal{S}$ with the zonotope $\mathcal{Z}$ is computed as proposed in Lemma 3.3, the multiplication $\mathcal{L}^{\mathcal{Z}}\,\mathcal{Z}$ is presented next.

Proposition 3.5 (Matrix Zonotope Map): The set resulting from the product of a matrix zonotope $\mathcal{L}^{\mathcal{Z}} = \{L^{(0)} + \sum_{i=1}^{\kappa}\rho_i\,L^{(i)}|\rho_i \in [-1,1]\}$ and a zonotope $\mathcal{Z} = (c, g^{(1)}, \ldots, g^{(e)})$ is over-approximated by a zonotope:

$$\begin{aligned}\mathcal{L}^{\mathcal{Z}}\,\mathcal{Z} &= \Big\{L^{(0)}\,\mathcal{Z} + \sum_{i=1}^{\kappa}\rho_i\,L^{(i)}\,\mathcal{Z}\big|\rho_i \in [-1,1]\Big\} \\ &= (L^{(0)}c, L^{(0)}g^{(1)}, \ldots, L^{(0)}g^{(e)}, \\ &\quad\; L^{(1)}c, L^{(1)}g^{(1)}, \ldots, L^{(1)}g^{(e)}, \ldots, \\ &\quad\; L^{(\kappa)}c, L^{(\kappa)}g^{(1)}, \ldots, L^{(\kappa)}g^{(e)}).\end{aligned} \qquad \square$$

Proof: The result follows directly from the addition and multiplication rule of zonotopes; see (2.1). The result is not exact because of the separate multiplication of the zonotope $\mathcal{Z}$ with the generator matrices $L^{(i)}$. This causes an over-approximation since distributivity does not hold for the following expression: $(C + D)\mathcal{Z} \subseteq C\mathcal{Z} + D\mathcal{Z}$, where $C, D \in \mathbb{R}^{n\times n}$. This is easily understood since the result of $C\mathcal{Z} + D\mathcal{Z}$ has twice as many generators as the result of $(C + D)\mathcal{Z}$. Only in special cases, e.g. when C, D are positive scalars, the expressions $(C + D)\mathcal{Z}$ and $C\mathcal{Z} + D\mathcal{Z}$ yield equal zonotopes since the resulting generators of $C\mathcal{Z}$ and $D\mathcal{Z}$ are aligned. □

In order to get an idea of the conservativeness of the computation $\mathcal{H}^{\mathcal{R}}(r) = e^{\mathcal{A}^{\mathcal{Z}} t}\,\mathcal{Z}$ in (3.24), an under-approximation of $\mathcal{H}^{\mathcal{R}}(r)$ is computed and compared to the over-approximation. Given a set of sampled matrices $\check{A}^{(i)} \in \mathcal{A}^{\mathcal{Z}}$, the under-approximation is obtained from $\bigcup_i e^{\check{A}^{(i)} t}\,\mathcal{Z}$. The under- and over-approximations are computed for the introductory examples (3.2) and (3.3) with a time step of $r = 0.04$ and are shown in Fig. 3.9(a) and 3.9(b). It can be seen that the over-approximation encloses the under-approximation quite tightly.



(a) Independent parameters from Example 3.2.

(b) Dependent parameters from Example 3.3.

Fig. 3.9.: Over- and under-approximated mapping of a zonotope by an uncertain matrix exponential.

The reachable set of the homogeneous solution for a time interval $[0, r]$ is computed analogously to the linear system with interval matrices as $\mathcal{H}^{\mathcal{R}}([0,r]) = \mathtt{CH}(\mathcal{X}^0, e^{\mathcal{A}^{\mathcal{Z}} r}\mathcal{X}^0) + \mathscr{F}\mathcal{X}^0$. It remains to consider the reachable set due to the uncertain input, which is done next.

Reachable Sets of Systems with Input

The reachable set originating from the set of inputs is computed similar to Theorem 3.1. The following proposition describes the necessary adaptations for linear systems with matrix zonotopes:

Proposition 3.6 (Reachable Set of the Input Solution): The reachable set due to the uncertain input is computed as

$$\mathcal{P}^{\mathcal{R}}(r) = r\,\mathcal{U} + \mathcal{A}^{\mathcal{Z}}\,\frac{r^2}{2!}\mathcal{U} + \sum_{i=2}^{\eta}\left(\frac{\mathscr{A}^i r^{i+1}}{(i+1)!}\mathcal{U}\right) + \mathscr{E}(r)\cdot r\cdot\mathcal{U}. \qquad \square$$

The proof is omitted since the proposition follows directly from Theorem 3.1. The minor modification is that the matrix zonotope $\mathcal{A}^{\mathcal{Z}}$ is over-approximated by its enclosing interval matrix $\mathscr{A}$ for powers greater than two. It is remarked that according to Theorem 3.1, it is not allowed to factorize the computation, e.g. to compute with $(r+\mathcal{A}^{\mathcal{Z}}\,\frac{r^2}{2!})\mathcal{U}$ instead of $r\,\mathcal{U}+\mathcal{A}^{\mathcal{Z}}\,\frac{r^2}{2!}\,\mathcal{U}$.

For input sets $\mathcal{U}$ where the origin is not contained, the input correction matrix $\tilde{\mathcal{F}}$ has to be computed as proposed in Prop. 3.4. In order to apply this proposition, the matrix zonotope $\mathcal{A}^{\mathcal{Z}}$ is over-approximated by its enclosing interval matrix.

Algorithmic Realization

The algorithmic realization of computing reachable sets of linear systems specified by matrix zonotopes is identical to Alg. 4. The only differences are the modified computations of $\mathcal{R}([kr,(k+1)r]) = e^{\mathcal{A}^{\mathcal{Z}}r}\mathcal{R}([(k-1)r,kr])$ and of the inhomogeneous solution sets $\tilde{\mathcal{P}}(r), \mathcal{P}^{\mathcal{R}}(r)$.

Clearly, the resulting reachable sets are more accurate when the system matrix is specified by a few parameters. However, the computational costs are higher, which is mainly caused by the matrix zonotope multiplication of zonotopes; see Prop. 3.5. Not only does the computation of this formula itself cause higher computational costs, but the resulting zonotopes also have a greater order than from an interval matrix multiplication; see Theorem 3.3. From this follows that the reduction operations in `reduce`() are more demanding. The tradeoff between accuracy and computational costs is also discussed for the following numerical examples.

Numerical Examples

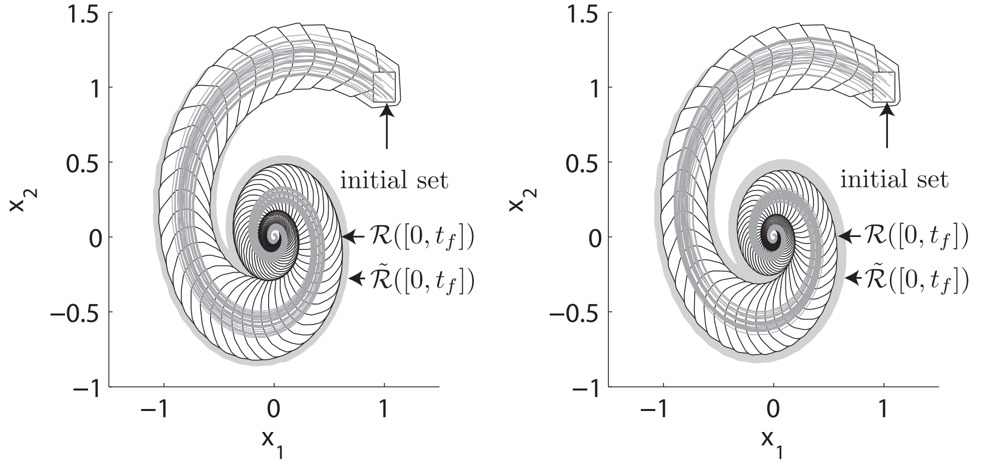
In analogy to the previous sections, the two- and five-dimensional examples are used to demonstrate the proposed computational techniques. In contrast to the previous numerical examples, not only are the results demonstrated, but also compared to the results obtained from the interval matrix approach in Sec. 3.3.1.

The two-dimensional example is specified by a matrix zonotope and is given as

$$\dot{x} = A\,x, \quad A \in \begin{bmatrix}-1 & -4\\ 4 & -1\end{bmatrix} + [-1,1]\begin{bmatrix}0.15 & 0.15\\ 0.15 & 0.15\end{bmatrix},\; x(0) \in \begin{bmatrix}[0.9,1.1]\\ [0.9,1.1]\end{bmatrix}. \tag{3.25}$$

In contrast to the numerical example of linear systems specified by interval matrices in (3.20), the enclosing interval matrix of this example has larger uncertainties while the input uncertainties are not considered in this example. This allows the different results of both algorithms to be better emphasized. The reachable set of this example is computed according to Alg. 4 using the new methods for the computation with matrix zonotopes. As in previous examples, a time horizon of $t_f = 5$ and a time step size of $r = 0.04$ has been used, resulting in 125 iterations.

The reachable set $\mathcal{R}([0,t_f])$ for a zonotope order of $\hat{\varrho} = 10$ and $\hat{\varrho} = 20$ is shown in Fig. 3.10(a) and Fig. 3.10(b). Beyond the order of $\hat{\varrho} = 20$, the result did not improve considerably. The reachable set $\tilde{\mathcal{R}}([0,t_f])$ of the corresponding interval matrices (using the methods of Sec. 3.3.1) is shown as a gray set in the background.



(a) Zonotope order: 10.

(b) Zonotope order: 20.

Fig. 3.10.: Reachable sets of the two-dimensional example – different zonotope order.

The five-dimensional example is given as $\dot{x} = A\,x + u(t)$, where

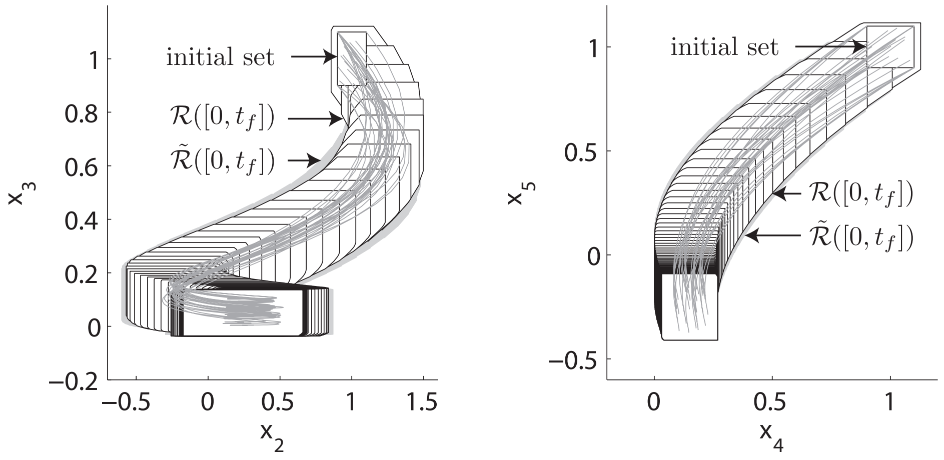
$$A \in \begin{bmatrix} -1 & -4 & 0 & 0 & 0 \\ 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & -3 & 1 & 0 \\ 0 & 0 & -1 & -3 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{bmatrix} + [-1,1] \begin{bmatrix} 0.1 & 0.1 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.1 & 0 \\ 0 & 0 & 0.1 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix},$$

$$x(0) \in \begin{bmatrix} [0.9,1.1] \\ [0.9,1.1] \\ [0.9,1.1] \\ [0.9,1.1] \\ [0.9,1.1] \end{bmatrix}, \; u(t) \in \begin{bmatrix} [0.9,1.1] \\ [-0.25,0.25] \\ [-0.1,0.1] \\ [0.25,0.75] \\ [-0.75,-0.25] \end{bmatrix}.$$

For this example, the same time horizon $t_f = 5$, time step size $r = 0.04$, and zonotope order $\hat{\varrho} = 10$ than for previous five-dimensional examples is used. The resulting reachable set is displayed in Fig. 3.11 for two different projections on two dimensions. Again, the reachable set $\tilde{\mathcal{R}}([0,t_f])$, obtained from the methods used for interval matrices, is shown in the background for comparison.

The scalability of the computations is shown by listing the computation times for randomly generated examples of higher dimension in Tab. 3.3. Besides the dimension n, the number of generator matrices κ of the matrix zonotope $\mathcal{A}^{\mathcal{Z}}$ has been varied. All reachable sets are computed with 125 iterations. The computations were performed with Matlab on a single core desktop PC with an AMD Athlon64 3700+ processor. Note that the computation times for dimensions 5, 10, and 20 differ only marginally due to the overhead time caused by function calls.

Remark 3.3 (Interval Matrix vs. Matrix Zonotope): It remains to discuss when it is advantageous to use the computational methods for interval matrices in Sec. 3.3.1 and when it is better to use the methods developed for zonotope matrices in Sec. 3.3.2. In the event that all parameters of an uncertain matrix $\mathcal{A}$ vary independently, the uncertain



(a) Projection onto x_2, x_3.

(b) Projection onto x_4, x_5.

Fig. 3.11.: Reachable sets of the five-dimensional example.

Tab. 3.3.: Computational times.

Dimension n	5	10	20	50	100
Number of generator matrices $\kappa = 1$					
CPU-time [s]	0.09	0.15	0.36	2.58	12.74
Number of generator matrices $\kappa = 2$					
CPU-time [s]	0.09	0.18	0.44	3.66	16.97
Number of generator matrices $\kappa = 4$					
CPU-time [s]	0.11	0.26	0.73	5.99	25.95

model should definitely be modeled as an interval matrix and the methods from Sec. 3.3.1 should be applied. Firstly, the computation is more efficient, as can be observed from the comparison of computation times in Tab. 3.2 and Tab. 3.3. Secondly, this computational technique is more accurate when all elements of $\mathcal{A}$ vary independently. One reason for the better accuracy is that the first two Taylor terms of $e^{\mathcal{A}t}$ are exact according to Lemma 3.1, which is not the case when applying (3.24). Another reason is that the multiplication of an interval matrix with a zonotope results in a zonotope of lower order compared to a multiplication with a matrix zonotope. Consequently, a smaller over-approximation is created when order reduction is applied.

If the uncertainties of the uncertain system matrix $\mathcal{A}$ depend on a few parameters ρ_i, the problem should be formulated by matrix zonotopes and the methods for matrix zonotopes in Sec. 3.3.2 should be applied. This is because the results are more accurate, as can be observed by the numerical examples, while the computational times are not significantly higher for a small number of generator matrices κ (compare Tab. 3.2 and Tab. 3.3). □

3.3.3. System Matrix Bounded by Matrix Polytopes

The final class of uncertain system matrices to be presented are matrix polytopes. This class of uncertain matrices is more general than the representation of sets of matrices by matrix zonotopes or interval matrices, as shown later. In analogy to the V-representation of polytopes (see Def. 2.2), a matrix polytope is defined as

$$\mathcal{A} = \Big\{ \sum_{i=1}^{\kappa} \rho_i \tilde{A}^{(i)} \Big| \tilde{A}^{(i)} \in \mathbb{R}^{n\times n}, \rho_i \in \mathbb{R}, \rho_i \geq 0, \sum_{i=1}^{\kappa} \rho_i = 1 \Big\}. \tag{3.26}$$

This definition is identical to the one of matrix zonotopes in (3.21), except that the sum of parameters $\sum_{i=1}^{\kappa} \rho_i$ has to be one. In analogy to the V-representation of polytopes in Def. 2.2, the matrices $\tilde{A}^{(i)}$ are referred to as *matrix vertices*. For this class of uncertain system matrices, no new computational methods are presented. In order to solve this problem class, the matrix polytopes are over-approximated by interval matrices or matrix zonotopes such that the reachability problem can be solved with the methods introduced previously.

In order to use methods for the over-approximation of polytopes in the Euclidean space, the matrix polytopes are mapped to polytopes in the Euclidean space. There, the polytopes are over-approximated by zonotopes or multidimensional intervals. The enclosing zonotopes or multidimensional intervals are then transformed into the space of the matrices so that one receives over-approximating matrix zonotopes or interval matrices. Thus, it is required that the mapping $f(X) = Y$ from the matrix space X to the Euclidean space Y is bijective, i.e. for every $y \in Y$, there is exactly one $x \in X$ such that $f(x) = y$.

This can be achieved by storing the elements of the matrix vertices $\tilde{A}^{(i)}$ in a vector $\hat{a}^{(i)} \in \mathbb{R}^{n^2}$ of dimension n^2 which can be interpreted as a vertex in the n^2-dimensional Euclidian space. This can be done by e.g. concatenating the column vectors of $\tilde{A}^{(i)} = [a_1^{(i)}, \dots, a_n^{(i)}]$, resulting in the column vector $\hat{a}^{(i)} = [a_1^{(i)T}, \dots, a_n^{(i)T}]^T$. The same idea has been used to generate random matrices in [160]. By doing so, one can write

$$f(\mathcal{A}) = \mathcal{P} = \Big\{ \sum_{i=1}^{\kappa} \rho_i \hat{a}^{(i)} \Big| \hat{a}^{(i)} \in \mathbb{R}^{n^2}, \rho_i \in \mathbb{R}, \rho_i \geq 0, \sum_{i=1}^{\kappa} \rho_i = 1 \Big\},$$

which is a polytope. Next, the polytope $\mathcal{P}$ is over-approximated by a multidimensional interval $\mathcal{I}$ in order to obtain an interval matrix $\mathscr{A}$ or by a zonotope $\mathscr{Z}$ in order to obtain a matrix zonotope $\mathcal{A}^{\mathscr{Z}}$. The over-approximation by a multidimensional interval $\mathcal{I}$ is performed according to Prop. 2.3 and the enclosure by a zonotope $\mathscr{Z}$ of order one (parallelotope) is performed according to Prop. 2.4. The enclosure by zonotopes of higher order is the subject of future work. In order to apply the parallelotope enclosure suggested in Prop. 2.4, the linear map by Λ has to be determined. This is done by principal component analysis (PCA) as presented in [158].

The over-approximating interval matrix $\mathscr{A}$ and the over-approximating matrix zonotope $\mathcal{A}^{\mathscr{Z}}$ are then obtained by reversing the rewriting from matrices to vectors, such that $\mathscr{A}$ and $\mathcal{A}^{\mathscr{Z}}$ contain the values from $\mathcal{I}$ and $\mathscr{Z}$, respectively.

Numerical Examples

In order to demonstrate the proposed technique, the following matrix polytope

$$\mathcal{A} = \Big\{ \sum_{i=1}^{3} \rho_i \hat{A}^{(i)} \Big| \tilde{A}^{(i)} \in \mathbb{R}^{2\times 2}, \rho_i \in \mathbb{R}, \rho_i \geq 0, \sum_{i=1}^{3} \rho_i = 1 \Big\},$$
$$\hat{A}^{(1)} = \begin{bmatrix} -1.1 & -4.1 \\ 4 & -1 \end{bmatrix}, \hat{A}^{(2)} = \begin{bmatrix} -1 & -4 \\ 4.1 & -1.1 \end{bmatrix}, \hat{A}^{(3)} = \begin{bmatrix} -1 & -3.9 \\ 3.9 & -1.1 \end{bmatrix}.$$

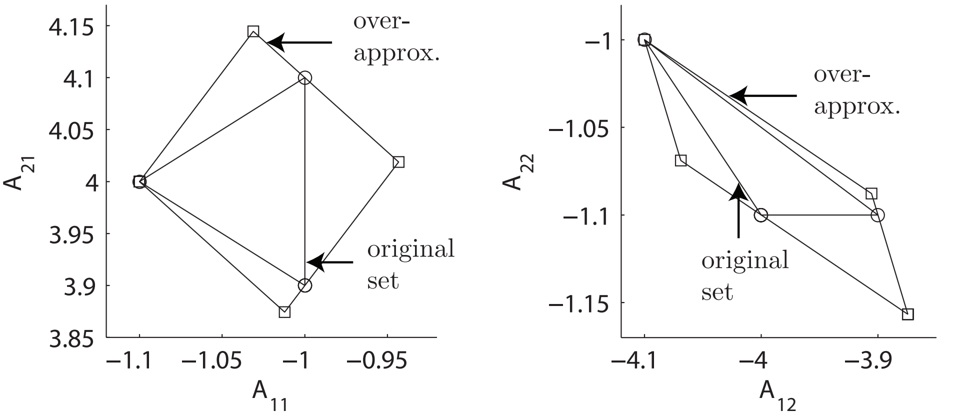
is over-approximated by an interval matrix and a matrix zonotope. The interval matrix is

$$\mathscr{A} = \begin{bmatrix} [-1.1, -1] & [-4.1, -3.9] \\ [3.9, 4.1] & [-1.1, -1] \end{bmatrix}$$

which can be easily checked by searching the minimum and maximum values of all matrix vertices. The computation of the over-approximating matrix zonotope is more challenging. After applying the principal component analysis as proposed in [158], the resulting enclosing matrix zonotope is

$$\mathcal{A}^{\mathcal{Z}} = \Big\{ \begin{bmatrix} -1.02 & 4.01 \\ -3.99 & -1.08 \end{bmatrix} + \sum_{i=1}^{2} \rho_i \hat{A}^{(i)} \Big| \rho_i \in [-1, 1] \Big\},$$
$$\hat{A}^{(1)} = 0.01 \begin{bmatrix} -4.4 & 6.3 \\ -9.7 & 4.4 \end{bmatrix}, \hat{A}^{(2)} = 0.01 \begin{bmatrix} -3.4 & -7.2 \\ -1.6 & 3.4 \end{bmatrix}.$$

Projections of possible values of the elements of $\mathcal{A}^{\mathcal{Z}}$ for pairs of matrix elements are illustrated in Fig. 3.12.



(a) Projection on elements A_{11}, A_{21}. (b) Projection on elements A_{12}, A_{22}.

Fig. 3.12.: Over-approximation of a matrix polytope by a matrix zonotope.

3.4. Nonlinear Continuous Systems with Uncertain Parameters

So far, reachable sets of linear continuous systems have been investigated. Although a fairly large group of dynamic systems can be described by linear continuous systems, the extension to nonlinear continuous systems is an important step for the analysis of more complex systems. The analysis of nonlinear systems is much more complicated since many valuable properties are no longer valid. One of them is the superposition principle, which allows the homogeneous and the inhomogeneous solution to be obtained separately. Another one is that reachable sets of LTI systems can be computed by a linear map. This allows the property to be employed that geometric representations such as ellipsoids, zonotopes, and polytopes are invariant under linear transformations, i.e. they are again mapped to ellipsoids, zonotopes and polytopes, respectively. Because of these reasons, the reachability analysis of nonlinear systems is based on linearization in this thesis. Since the linearization causes additional errors, the linearization errors are determined in an over-approximative way and added as an additional uncertain input so that an over-approximative computation is ensured.

In this section, general nonlinear continuous systems with uncertain parameters and Lipschitz continuity are considered. In analogy to the previous linear systems, the initial state $x(0)$ can take values from a set $\mathcal{X}^0 \subset \mathbb{R}^n$. The dynamics depends on a set of uncertain, but constant model parameters ρ_i. The parameters are bounded by intervals ($\rho_i \in \mathcal{I}$), such that the parameter vector ρ stays within a multidimensional interval $\mathcal{P} \in \mathcal{I}^p$, where p is the number of parameters. The input u takes values from a set $\mathcal{U} \subset \mathbb{R}^m$. The evolution of the state x is defined by the following differential equation:

$$\begin{gathered}\dot{x} = f(x(t), u(t), \rho), \qquad (3.27)\\ x(0) \in \mathcal{X}^0 \subset \mathbb{R}^n, \quad \rho \in \mathcal{P} \subset \mathcal{I}^p, \quad u(t) \in \mathcal{U} \subset \mathbb{R}^m,\end{gathered}$$

where $u(t)$ and $f()$ are assumed to be globally Lipschitz continuous so that the Taylor expansion for the state and the input can always be computed, which is required for the linearization. Thus, the system has no finite escape time. It is further remarked that the system can be instable as long as it is globally Lipschitz.

In linear systems, inputs are additive while the parameters are associated with the system matrix. For nonlinear systems, this distinction is not so clear. In this thesis, the classification is made after the linearization is performed: Parameters are the variables associated with the obtained system matrix and inputs are additive. However, it is also possible to interpret a parameter as a non-additive input. An overview of the conservative linearization procedure is presented below.

3.4.1. Overview of Reachable Set Computations

A brief visualization of the overall concept for computing the reachable set $\mathcal{R}([0, t_f])$ is shown in Fig. 3.13. As in the previous approaches, the reachable set is iteratively computed

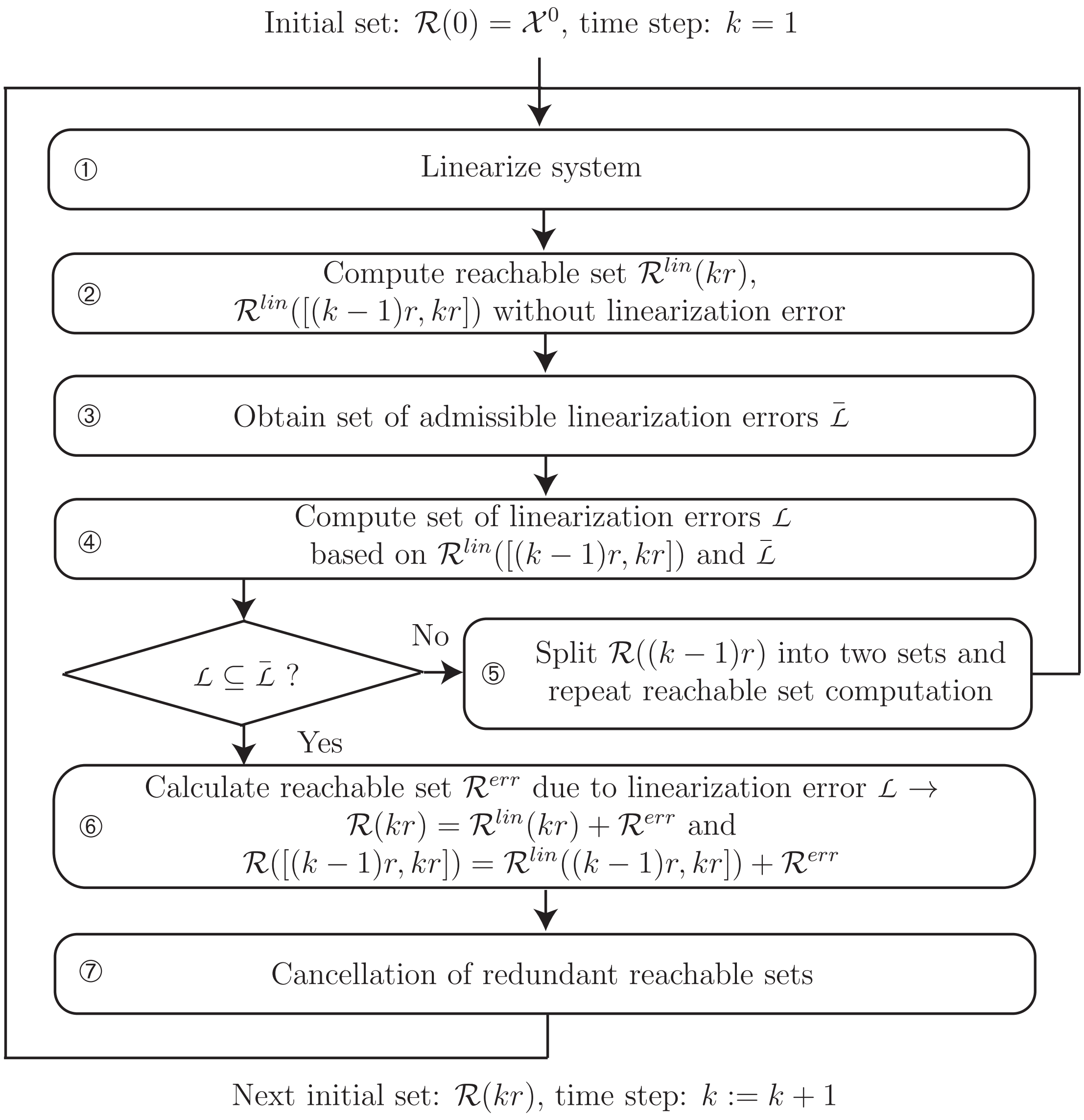

Initial set: $\mathcal{R}(0) = \mathcal{X}^0$, time step: $k = 1$
① Linearize system
② Compute reachable set $\mathcal{R}^{lin}(kr)$, $\mathcal{R}^{lin}([(k-1)r, kr])$ without linearization error
③ Obtain set of admissible linearization errors $\bar{\mathcal{L}}$
④ Compute set of linearization errors $\mathcal{L}$ based on $\mathcal{R}^{lin}([(k-1)r, kr])$ and $\bar{\mathcal{L}}$
$\mathcal{L} \subseteq \bar{\mathcal{L}}$?
No
⑤ Split $\mathcal{R}((k-1)r)$ into two sets and repeat reachable set computation
Yes
⑥ Calculate reachable set $\mathcal{R}^{err}$ due to linearization error $\mathcal{L} \rightarrow$ $\mathcal{R}(kr) = \mathcal{R}^{lin}(kr) + \mathcal{R}^{err}$ and $\mathcal{R}([(k-1)r, kr]) = \mathcal{R}^{lin}((k-1)r, kr]) + \mathcal{R}^{err}$
⑦ Cancellation of redundant reachable sets
Next initial set: $\mathcal{R}(kr)$, time step: $k := k + 1$

Fig. 3.13.: Computation of reachable sets – overview.

for smaller time intervals $t \in [(k-1)r, kr]$ where $k \in \mathbb{N}^+$, such that $\mathcal{R}([0, t_f])$ is obtained by their union: $\mathcal{R}([0, t_f]) = \bigcup_{k=1 \ldots t_f/r} \mathcal{R}([(k-1)r, kr])$. The procedure for computing the reachable sets of the consecutive time intervals is as follows:

① The nonlinear system $\dot{x} = f(x, u, \rho)$ is linearized to a system of the form $\dot{x} = f^{lin}(x, u, \rho) = A(\rho)\Delta x + B(\rho)\Delta u + d^{\mathrm{lin}}(\rho)$. The system and input matrix $A(\rho)$ and $B(\rho)$, as well as the vector $d^{\mathrm{lin}}(\rho)$, are of proper dimension, all depending on the parameter vector ρ. The set of linearization errors $\mathcal{L}$ ensures that $f(x, u, \rho) \in f^{lin}(x, u, \rho) + \mathcal{L}$, which allows the reachable set to be computed in an over-approximative way.

② Due to the superposition principle of linear systems, the reachable set $\mathcal{R}^{lin}(kr)$ and $\mathcal{R}^{lin}([(k-1)r, kr])$ of $f^{lin}(x, u, \rho)$ are computed without consideration of the linearization errors ($\mathcal{L} = 0$). Later, the reachable set $\mathcal{R}^{err}$ caused by the set of linearization errors $\mathcal{L}$ is added to $\mathcal{R}^{lin}$.

③ The expansion of the reachable set due to the linearization error is restricted by a set $\bar{\mathcal{R}}^{err}$ in which $\mathcal{R}^{err}$ has to be enclosed. This set allows the set $\bar{\mathcal{L}}$ of admissible linearization errors obtained from the linearized system dynamics $f^{lin}(x, u, \rho)$ to be computed.

④ The admissible reachable set $\bar{\mathcal{R}}([(k-1)r, kr]) := \mathcal{R}^{lin}([(k-1)r, kr]) + \bar{\mathcal{R}}^{err}$ defines the set for which the over-approximative linearization error has to be found. The computation of the linearization error is based on interval arithmetics so that it is bounded by a multidimensional interval $\mathcal{L} \in I^n$.

⑤ Where $\mathcal{L} \nsubseteq \bar{\mathcal{L}}$, the linearization error is not admissible, requiring the initial reachable set $\mathcal{R}((k-1)r)$ to be split into two reachable sets. This implies performing the reachable set computations for both of the newly obtained sets once more. Hence, the number of reachable set representations for this time interval has increased by one.

⑥ If $\mathcal{L} \subseteq \bar{\mathcal{L}}$, the linearization error is accepted and the reachable set is obtained by superposition of the reachable set without linearization error and the one due to the linearization error: $\mathcal{R}(kr) = \mathcal{R}^{lin}(kr) + \mathcal{R}^{err}$ and $\mathcal{R}([(k-1)r, kr]) = \mathcal{R}^{lin}([(k-1)r, kr]) + \mathcal{R}^{err}$.

⑦ It remains to increase the time step ($k := k+1$) and cancel redundant reachable sets that are already covered by previously computed reachable sets. This decreases the number of reachable sets that have to be considered in the next time interval. The initial set for the next time step is $\mathcal{R}(kr)$.

Besides the splitting of the reachable set in the state space, it is also possible to split the input and parameter set in an analogous way. However, splitting of the reachable set is more effective since the split of the input or parameter set results in largely overlapping reachable sets.

3.4.2. Linearization

The local linearization of the nonlinear system (3.27) is performed by a Taylor series. In order to introduce a concise notation, the state and input vector are combined to a new vector

$$z = \begin{bmatrix} x \\ u \end{bmatrix}.$$

This allows a Taylor series of the nonlinear system dynamics (3.27) with a parameter vector $\rho \in \mathcal{P}$ to be formulated:

$$\begin{aligned} \dot{x}_i = f_i(z, \rho) = f_i(z^*, \rho) &+ \frac{\partial f_i(z, \rho)}{\partial z}\Big|_{z=z^*}(z - z^*) \\ &+ \frac{1}{2}(z - z^*)^T \frac{\partial^2 f_i(z, \rho)}{\partial z^2})\Big|_{z=z^*}(z - z^*) + \dots \end{aligned}$$

The infinite Taylor series can be over-approximated by a first order Taylor series and its Lagrange remainder:

$$\dot{x}_i \in \underbrace{f_i(z^*,\rho) + \frac{\partial f_i(z,\rho)}{\partial z}\Big|_{z=z^*}(z-z^*)}_{1^{st}\text{ order Taylor series}} + \underbrace{\frac{1}{2}(z-z^*)^T \frac{\partial^2 f_i(\xi,\rho)}{\partial z^2}(z-z^*)}_{\text{Lagrange remainder}\mathcal{L}_i}\,. \tag{3.28}$$

Let z, z^* be fixed, then the Lagrange remainder $\mathcal{L}$ can take any value that results from $\xi \in \{z^* + \alpha(z - z^*)|\alpha \in [0,1]\}$; see [23]. The computation of the set $\mathcal{L}$ resulting from the set of possible values of z is presented in the following subsection. In order to obtain the standard notation of the linearized system, the z vector is separated into the state vector x and the input vector u.

$$\begin{aligned}\dot{x} &\in f(z^*,\rho) + \frac{\partial f(z,\rho)}{\partial z}\Big|_{z=z^*}(z-z^*) + \mathcal{L}\\ &= f(x^*,u^*,\rho) + \underbrace{\frac{\partial f(x,u,\rho)}{\partial x}\Big|_{x=x^*,u=u^*}(x-x^*)}_{A\Delta x} + \underbrace{\frac{\partial f(x,u,\rho)}{\partial u}\Big|_{x=x^*,u=u^*}(u-u^*)}_{B\Delta u} + \mathcal{L}\end{aligned} \tag{3.29}$$

with $\Delta x = x - x^*$, $\Delta u = u - u^*$. Where there are no uncertain parameters ρ, one obtains matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, whose elements are real numbers. If the system contains uncertain parameters, one has the choice to over-approximate the uncertain system and input matrix by interval matrices or matrix zonotopes. Once the uncertain input matrix $\mathcal{B} = \{B(\rho)|\rho \in \mathcal{I}^p\}$ has been found, it is multiplied by the set of inputs to obtain the overall set of possible inputs. Where $\mathcal{B}$ is represented by an interval matrix, the multiplication is performed according to Theorem 3.3 and according to Prop. 3.5 if $\mathcal{B}$ is modeled as a zonotope matrix. Examples of both representation forms are given below.

Example 3.4 (Linearization of a Nonlinear Dynamic System):

$$\dot{x}_1 = \rho_1 x_1 + \rho_2(x_1+1)^3 x_2 + \rho_1\cos(x_1)u, \qquad \dot{x}_2 = \rho_1 x_2^2 u,$$
$$\rho_1 \in [0,1],\ \rho_2 \in [-1,2],\ u \in \mathcal{U} = [1,2] = (1.5, 0.5), \quad x_1^* = 0,\ x_2^* = 1,\ u^* = 2.$$

Note that $(1.5, 0.5)$ is the zonotope notation of the interval $[1,2]$. The system matrix and input matrix are after linearization around x^*, u^*:

$$A = \begin{bmatrix}\rho_1 + 3\rho_2 & \rho_2\\ 0 & 4\rho_1\end{bmatrix}, \quad B = \begin{bmatrix}\rho_1\\ \rho_1\end{bmatrix}.$$

The over-approximation by interval matrices or matrix zonotopes yields:

Interval matrix:	Matrix zonotope ($\rho_1^*, \rho_2^* \in [-1,1]$):
$\mathcal{A} = \begin{bmatrix}[-3,7] & [-1,2]\\ 0 & [0,4]\end{bmatrix}$	$\mathcal{A}^{\mathcal{Z}} = \begin{bmatrix}2 & 0.5\\ 0 & 2\end{bmatrix} + \rho_1^*\begin{bmatrix}0.5 & 0\\ 0 & 2\end{bmatrix} + \rho_2^*\begin{bmatrix}4.5 & 1.5\\ 0 & 0\end{bmatrix}$
$\mathcal{U}^* = \mathcal{B}\mathcal{U} = \begin{bmatrix}[0,1]\\ [0,1]\end{bmatrix}(1.5, 0.5)$	$\mathcal{U}^* = \mathcal{B}^{\mathcal{Z}}\mathcal{U} = \left(\begin{bmatrix}0.5\\ 0.5\end{bmatrix} + \rho_1^*\begin{bmatrix}0.5\\ 0.5\end{bmatrix}\right)(1.5, 0.5)$
$\overset{\text{Theorem 3.3}}{=} \left(\begin{bmatrix}0.75\\ 0.75\end{bmatrix}, \begin{bmatrix}0.25\\ 0.25\end{bmatrix}, \begin{bmatrix}1\\ 0\end{bmatrix}, \begin{bmatrix}0\\ 1\end{bmatrix}\right)$	$\overset{\text{Prop. 3.5}}{=} \left(\begin{bmatrix}0.75\\ 0.75\end{bmatrix}, \begin{bmatrix}0.25\\ 0.25\end{bmatrix}, \begin{bmatrix}0.75\\ 0.75\end{bmatrix}, \begin{bmatrix}0.25\\ 0.25\end{bmatrix}\right)$

□

The reachable set $\mathcal{R}^{lin}$ of the linearized system is obtained as described in Sec. 3.2 or Sec. 3.3 depending on the existence of uncertain parameters. Next, the computation of the set of linearization errors is presented.

3.4.3. Computation of the Set of Linearization Errors

As described in the overview of the linearization approach (see Fig. 3.13), the linearization error $\mathcal{L}$ is obtained by an evaluation of the Lagrange remainder. After defining

$$J_i(\xi, \rho) := \frac{\partial^2 f_i(\xi, \rho)}{\partial z^2},$$

where i refers to the i-th coordinate of f, one can write the Lagrange remainder in (3.28) as

$$\begin{aligned} \mathcal{L}_i &= \frac{1}{2}(z - z^*)^T J_i(\xi, \rho)(z - z^*), \\ \xi(z, z^*) &\in \{z^* + \alpha(z - z^*) | \alpha \in [0, 1]\}. \end{aligned} \tag{3.30}$$

In order to determine the set $\mathcal{L}_i$ for the time interval $t \in [0, r]$, one has to consider the possible values of z within this time interval. The values of z are within $\hat{\mathcal{Z}}([0, r])$, which is the Cartesian product $\hat{\mathcal{Z}}([0, r]) := \mathcal{R}([0, r]) \times \mathcal{U}$ of possible states restricted to $x([0, r]) \in \mathcal{R}([0, r])$ and inputs restricted to $u \in \mathcal{U}$. As $\mathcal{R}([0, r])$ is not known when the linearization error is computed, it is over-approximated by $\mathcal{R}^{lin}([0, r]) + \bar{\mathcal{R}}^{err}$, where $\bar{\mathcal{R}}^{err}$ is the largest allowed set caused by the set of linearization errors $\mathcal{L}$. The specification of $\bar{\mathcal{R}}^{err}$ is discussed in the next subsection. In order to determine the linearization error set $\mathcal{L}_i$ for $z \in \hat{\mathcal{Z}}([0, r])$ in an efficient way, the following over-approximation is computed:

Proposition 3.7 (Over-approximation of the Set of Linearization Errors): The absolute values of the Lagrange remainder can be over-approximated for $z \in \hat{\mathcal{Z}}$ where $\hat{\mathcal{Z}}$ is a zonotope $\hat{\mathcal{Z}} = (c, g^{(1)}, \dots, g^{(e)})$. The over-approximation is obtained by the following computations:

$$\begin{gathered} |\mathcal{L}_i| \subseteq [0, l_i] \\ \text{with } l_i = \frac{1}{2}\gamma^T \mathtt{max}(|J_i(\xi(z, z^*), \rho)|)\gamma, \; z \in \hat{\mathcal{Z}}, \, \rho \in \mathcal{P} \\ \text{and } \gamma = |c - z^*| + \sum_{i=1}^{e} |g^{(i)}|, \end{gathered}$$

where c is the center and $g^{(i)}$ are the generators of the zonotope $\mathcal{Z}$. The `max`()-operator and the absolute values are applied elementwise. □

Proof: The following over-approximations apply for the absolute value of $\mathcal{L}_i$.

$$\begin{aligned}
|\mathcal{L}_i| &= \Big\{\frac{1}{2}|(z-z^*)^T\, J_i(\xi(z,z^*),\rho)\,(z-z^*)|\,\Big|z\in\hat{\mathcal{Z}},\rho\in\mathcal{P}\Big\} \\
&\subseteq \frac{1}{2}\Big[0,\mathtt{max}_{z\in\hat{\mathcal{Z}},\rho\in\mathcal{P}}\big(|(z-z^*)^T\, J_i(\xi(z,z^*),\rho)\,(z-z^*)|\big)\Big] \\
&\subseteq \frac{1}{2}\Big[0,\mathtt{max}_{z\in\hat{\mathcal{Z}},\rho\in\mathcal{P}}\big(|z-z^*|^T\,|J_i(\xi(z,z^*),\rho)|\,|z-z^*|\big)\Big] \\
&\subseteq \frac{1}{2}\Big[0,\mathtt{max}_{z\in\hat{\mathcal{Z}}}(|z-z^*|)^T\,\mathtt{max}_{z\in\hat{\mathcal{Z}},\rho\in\mathcal{P}}(|J_i(\xi(z,z^*),\rho)|)\,\mathtt{max}_{z\in\hat{\mathcal{Z}}}(|z-z^*|)\Big]
\end{aligned}$$

The expression $\mathtt{max}_{z\in\hat{\mathcal{Z}}}(|z-z^*|)$ can be further rewritten since $z\in\hat{\mathcal{Z}}$ is within a zonotope with center c and generators $g^{(i)}$:

$$\mathtt{max}_{z\in\hat{\mathcal{Z}}}(|z-z^*|) = \mathtt{max}_{\beta_i\in[-1,1]}(|c-z^*+\sum_{i=1}^{p}\beta_i g^{(i)}|) \leq |c-z^*|+\sum_{i=1}^{e}|g^{(i)}| = \gamma$$

such that the expression of Prop. 3.7 is obtained. □

The expression $\mathtt{max}_{z\in\hat{\mathcal{Z}},\rho\in\mathcal{P}}(|J_i(\xi(z,z^*),\rho|)$ in Prop. 3.7 is computed via interval arithmetics [87]. To do so, the values of z have to be over-approximated by an interval vector as shown in Prop. 2.2: $z\in\mathtt{box}(\hat{\mathcal{Z}})$. From this follows that $\xi(z,z^*)\in\{z^*+\alpha(z-z^*)|\alpha\in[0,1]\}$ also becomes an interval vector and the parameter values ρ are already specified as intervals. Note that it is also possible to directly apply interval arithmetics to (3.30) without the intermediate computations in Prop. 3.7. So far, it has not been investigated in which cases the direct application of interval arithmetics is beneficial.

As a byproduct of Prop. 3.7, the linearization point z^* that minimizes the set of Lagrange remainders is easily found.

Corollary 3.2 (Optimal Linearization Point): The bounding vector ℓ of the absolute value of the Lagrange remainder is minimized by choosing $z^*=c$ as the linearization point, where c is the center of the reachable set $\hat{\mathcal{Z}}$. □

Proof: The value of γ is minimized by $z^*=c$, which can be directly checked from its computation in Prop. 3.7. By choosing $z^*=c$, it follows that the set $\{\xi(z,z^*=c)|z\in\hat{\mathcal{Z}}\}=\hat{\mathcal{Z}}$ is independent of z^*, such that $\mathtt{max}(|J_i(\xi(z,z^*),\rho)|)$ is not affected by the linearization point z^*. From this follows that $z^*=c$ minimizes ℓ. □

After choosing $z^*=c$, it remains to solve the problem that the center c of $\hat{\mathcal{Z}}([0,r])$ is not known, since $\hat{\mathcal{Z}}([0,r])$ is computed after the linearization. As a solution, c is approximated based on the center $\hat{c}$ of $\hat{\mathcal{Z}}(0)$:

$$z^*=\hat{c}+\frac{r}{2}f(\hat{c},\mathtt{center}(\rho))\approx c$$

The operator $\mathtt{center}()$ returns the center of an interval vector. Next, the restriction of the linearization error is addressed.

3.4.4. Restriction of the Linearization Error

It is obvious that the Lagrange remainder strongly depends on the size of the reachable set. The larger the reachable set becomes, the more the set of linearization errors is expanding, which again enlarges the reachable set. This self-energizing process is limited in this thesis by restricting the linearization error $\mathcal{R}^{err}$ to a multidimensional interval $\bar{\mathcal{R}}^{err}$. The rate of growth of the admissible expansion of $\bar{\mathcal{R}}^{err}$ is set by the expansion vector $\theta \in \mathbb{R}^n$, which has to be specified by the user.

$$\bar{\mathcal{R}}^{err}(r) := [-\theta \cdot r, \theta \cdot r]. \tag{3.31}$$

Assuming that the linearization error is constant for a time interval $[kr, (k+1)r]$ and that the system matrix has no uncertainties, the following relation between the admissible reachable set and the admissible linearization error $\bar{\mathcal{L}} = [-\bar{\ell}, \bar{\ell}]$ exists according to (3.6):

$$\bar{\mathcal{R}}^{err}(r) = [-\theta \cdot r, \theta \cdot r] = A^{-1}(e^{Ar} - I)[-\bar{\ell}, \bar{\ell}]$$

After left multiplication with $(e^{Ar} - I)^{-1}A$, the vector bounding the admissible errors $\bar{\ell}$ is obtained as

$$\bar{\ell} = \big|(e^{Ar} - I)^{-1}A\big|\theta \cdot r.$$

Where the system matrix is an uncertain matrix, a representative of the uncertain matrix is chosen, e.g. the center matrix. Thus, the computation with uncertain matrices does not allow an exact computation of $\bar{\ell}$; however, the choice of the expansion vector θ by the user is already more or less arbitrary. When the constraint $\mathcal{L} \subseteq \bar{\mathcal{L}}$ is not fulfilled, which is equivalent to $\ell \leq \bar{\ell}$, the reachable set is split as explained next.

Splitting of Reachable Sets

In contrast to polytopes, which can be split into two polytopes by a separating hyperplane, zonotopes can only be split this way in special cases. However, a possibility to split a zonotope by splitting the j-th generator can be presented:

Proposition 3.8 (Splitting of Zonotopes): A zonotope $\mathcal{Z} = (c, g^{(1)}, \ldots, g^{(e)})$ is split into two zonotopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$ such that $\mathcal{Z}_1 \cup \mathcal{Z}_2 = \mathcal{Z}$ and $\mathcal{Z}_1 \cap \mathcal{Z}_2 = \mathcal{Z}^*$, where

$$\begin{array}{llllll} \mathcal{Z}_1 & = (c - \frac{1}{2}g^{(j)}, & g^{(1)}, \ldots, g^{(j-1)}, & \frac{1}{2}g^{(j)}, & g^{(j+1)}, \ldots, g^{(e)}) \\ \mathcal{Z}_2 & = (c + \frac{1}{2}g^{(j)}, & g^{(1)}, \ldots, g^{(j-1)}, & \frac{1}{2}g^{(j)}, & g^{(j+1)}, \ldots, g^{(e)}) \\ \mathcal{Z}^* & = (c, & g^{(1)}, \ldots, g^{(j-1)}, & & g^{(j+1)}, \ldots, g^{(e)}) \end{array} \qquad \square$$

Proof: First, a zonotope $(0, g^{(j)})$ that only consists of the j-th generator is considered. This generator can be split into two generators:

$$(0, g^{(j)}) = \left(-\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}\right) \cup \left(\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}\right)$$

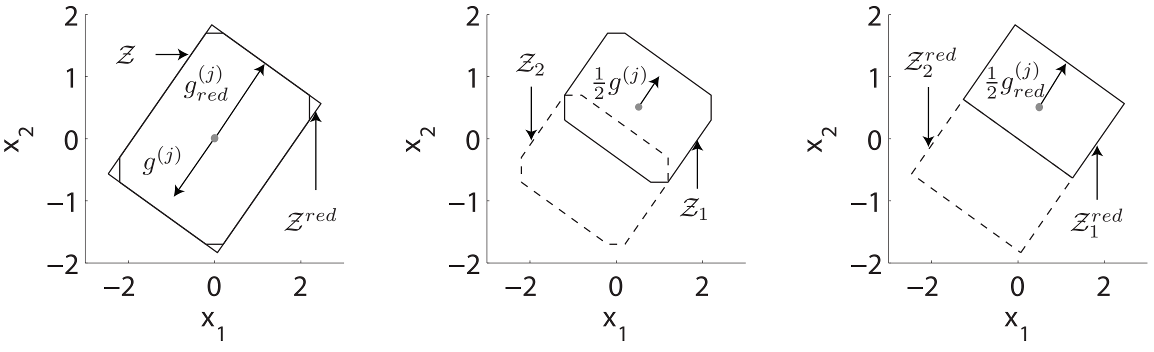
Adding $\mathcal{Z}^*$ to both sides of the above statement and using $A+(B\cup C)=(A+B)\cup(A+C)$, where A, B, C are sets in the Euclidean space, yields

$$\mathcal{Z}^* + (0, g^{(j)}) = \mathcal{Z}^* + \Big((-\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) \cup (\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) \Big)$$
$$= \Big(\mathcal{Z}^* + (-\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) \Big) \cup \Big(\mathcal{Z}^* + (\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) \Big).$$

The addition of zonotopes is performed by adding the centers and concatenating the generators as shown in (2.1), such that $\mathcal{Z} = \mathcal{Z}^* + (0, g^{(j)})$, $\mathcal{Z}_1 = \mathcal{Z}^* + (-\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)})$ and $\mathcal{Z}_2 = \mathcal{Z}^* + (\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)})$, resulting in $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$.
Furthermore, it can be derived from $(-\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) \cap (\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) = \{0\}$ that

$$\mathcal{Z}_1 \cap \mathcal{Z}_2 = \Big(\mathcal{Z}^* + (-\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) \Big) \cap \Big(\mathcal{Z}^* + (\frac{1}{2}g^{(j)}, \frac{1}{2}g^{(j)}) \Big) = \mathcal{Z}^*. \qquad \square$$

The greater the order of a zonotope $\mathcal{Z}$ is, the larger the overlapping zonotope $\mathcal{Z}^*$ is and, consequently, the less effective a split is. For this reason, zonotopes should be reduced to a certain order with, e.g., the methods presented in Sec. 2.5. The order reduction of zonotopes is performed in an over-approximated way such that for the reduced zonotope $\mathcal{Z}^{red}$ holds: $\mathcal{Z}^{red} \supseteq \mathcal{Z}$. The advantages and disadvantages of the order reduction are illustrated for a zonotope $\mathcal{Z}$ in a two-dimensional example with 4 generators according to Fig. 3.14(a). The split zonotopes of the original zonotope $\mathcal{Z}$ are denoted by $\mathcal{Z}_1$, $\mathcal{Z}_2$ and the ones of the reduced zonotope $\mathcal{Z}^{red}$ are denoted by $\mathcal{Z}_1^{red}$, $\mathcal{Z}_2^{red}$. The sets of the unreduced zonotope can be found in Fig. 3.14(b) and in Fig. 3.14(c) for the reduced zonotope. The advantage of the split of the unreduced zonotope is that the split zonotopes cover a smaller region: $\mathcal{Z}_1 \cup \mathcal{Z}_2 = \mathcal{Z} \subseteq \mathcal{Z}^{red} = \mathcal{Z}_1^{red} \cup \mathcal{Z}_2^{red}$. However, $\mathcal{Z}_1$ and $\mathcal{Z}_2$ overlap more than $\mathcal{Z}_1^{red}$ and $\mathcal{Z}_2^{red}$: $\mathcal{Z}_1 \cap \mathcal{Z}_2 = \mathcal{Z}^* \supseteq \mathcal{Z}^{red*} = \mathcal{Z}_1^{red} \cap \mathcal{Z}_2^{red}$. In order to obtain an optimal result, one has to find a compromise between the overlapping and the over-approximation of reachable sets.



(a) $\mathcal{Z}$, $\mathcal{Z}^{red}$

(b) $\mathcal{Z}_1$, $\mathcal{Z}_2$

(c) $\mathcal{Z}_1^{red}$, $\mathcal{Z}_2^{red}$

Fig. 3.14.: Split of a zonotope and the corresponding reduced zonotope.

It remains to find the j-th generator that splits the reachable set $\mathcal{R}(kr)$ into $\mathcal{R}_{1,j}(kr)$ and $\mathcal{R}_{2,j}(kr)$ in an optimal way. For this purpose, a performance measure for a split is defined based on the Lagrange remainder computed from $\mathcal{R}_{1,j}([kr,(k+1)r])$ and $\mathcal{R}_{2,j}([kr,(k+1)r])$.

Since the choice of the generator has a big influence on further computations, each generator is checked.

Heuristic 3.1 (Generator Selection for Splitting of Reachable Sets): The splitting of the zonotopial reachable set $\mathcal{R}(kr)$ is decided by a performance measure ϱ_j:

$$\varrho_j = \mathtt{max}(\ell_{1,j}/\bar{\ell}) \cdot \mathtt{max}(\ell_{2,j}/\bar{\ell}).$$

The divisions $\ell_{1,j}/\bar{\ell}$ and $\ell_{2,j}/\bar{\ell}$ are applied elementwise and `max()` returns the maximum value of the resulting vectors. The component j with the lowest value in the performance vector ϱ corresponds to the generator that has to be split. □

The linearization error limit is fulfilled if the lowest performance measure is less than 1 since this implies that $\ell_{1,j} \leq \bar{\ell}$ and $\ell_{2,j} \leq \bar{\ell}$, which in turn implies that $\mathcal{L}_{1,j} \subseteq \bar{\mathcal{L}}$ and $\mathcal{L}_{2,j} \subseteq \bar{\mathcal{L}}$. Where the lowest performance value is greater than 1, the obtained split sets have to be split again recursively.

Cancellation of Redundant Reachable Sets

The problem of splitting reachable sets is that the computational effort grows linearly with the number of splits. This effect can be reduced by canceling reachable sets that have already been reached.

In order to identify those sets, the set difference operation is used. As the set difference of two zonotopes is not a zonotope anymore, the zonotopes are converted to H-polytopes in an over-approximated way as presented in Sec. 2.5.6. The over-approximation is performed for the reachable sets of the past ζ time steps, where ζ can be freely chosen. If a polytope of the current time interval is empty after the set difference with the polytopes of the past ζ time steps, this reachable segment is canceled.

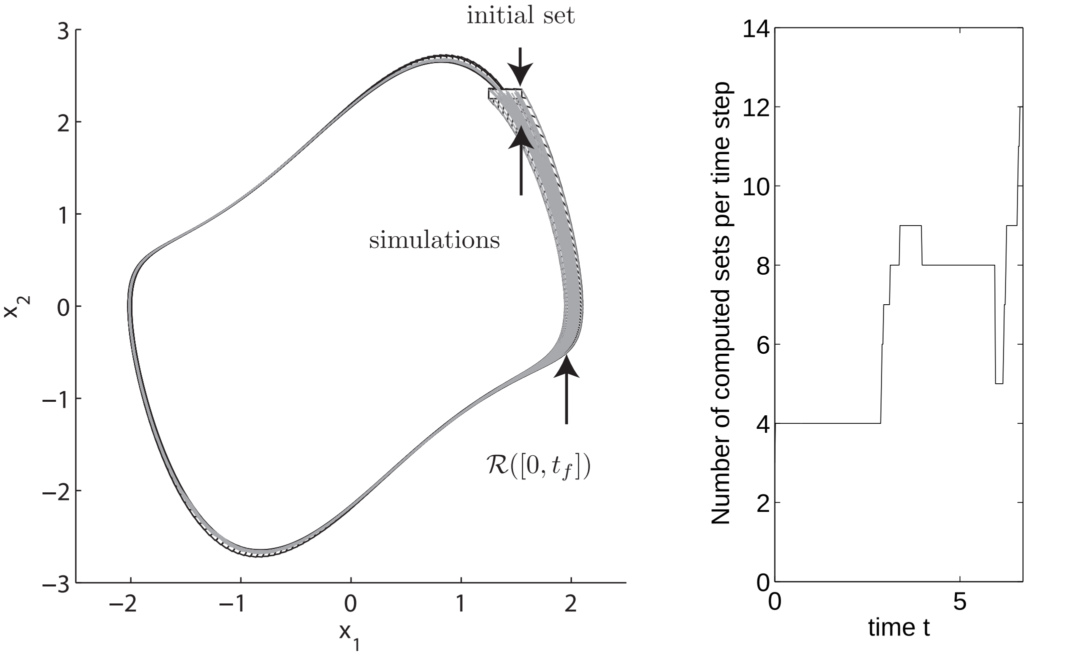
After the cancellation, the remaining polytopes are transformed back to zonotopes as presented in Prop. 2.4. Since the cancellation of reachable sets leads to an over-approximation, and in addition is computationally expensive, the described procedure is only applied every $\delta \in \mathbb{N}^+$ time steps, where δ is set by the user.

3.4.5. Numerical Examples

The approach for the computation of reachable sets of nonlinear systems is demonstrated by two examples. The first example is a Van-der-Pol oscillator, which is a standard example of nonlinear systems that have a limit cycle. The differential equations together with two different initial sets are

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= (1-x_1^2)x_2 - x_1.\end{aligned} \qquad x(0)^{small} \in \begin{bmatrix}[1.25, 1.55]\\ [2.25, 2.35]\end{bmatrix}, \quad x(0)^{large} \in \begin{bmatrix}[0.8, 2]\\ [2.25, 2.35]\end{bmatrix}.$$

The reachable set for this example is computed with a time increment of $r = 0.02$. The expansion vector is set to $\theta = [0.05, 0.05]^T$ and the cancellation of reachable sets is performed every $\delta = 100$ time steps. The reachable set for the small initial set is visualized in Fig. 3.15(a) and the one for the large initial set is shown in Fig. 3.16(a). The number of reachable sets that have to be computed in parallel for each time step is plotted in Fig. 3.15(b) and Fig. 3.16(b) for the small and large initial set, respectively. In both plots, it can be observed that redundant reachable sets are canceled after 4 and 6 time units. It can be further observed that for the large initial set, which is 4 times larger in the x_1-direction, also about 4 times as many sets have to be computed, resulting in computational times which differ by the same factor. The computational time for the small initial set is 18.91 s and the one for the larger initial set is 97.19 s. The computations were performed on a single core desktop PC with an AMD Athlon64 3700+ processor in Matlab. A byproduct of this example is that it can be shown that the Van-der-Pol oscillator has a stable limit cycle, since the reachable set enters the initial set after one cycle.



(a) Reachable set.

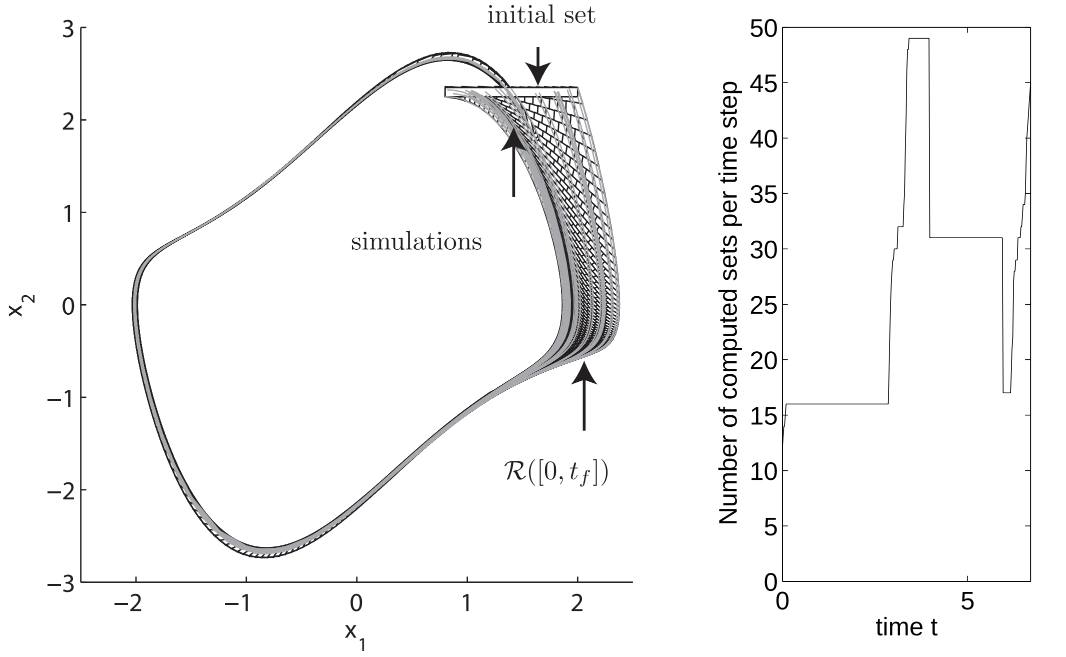
(b) Number of computed sets per time step.

Fig. 3.15.: Van-der-Pol oscillator: Small initial set

As a second example, a water tank system with uncertain inputs and parameters as illustrated in Fig. 3.17 is considered. The states x_i are the water levels of each tank and u is the water flow into the system. The inflow is controlled by measuring the water level of the last tank. This example is chosen as it can be easily formulated for different numbers of states by adding additional water tanks. The differential equation for the water level of the first tank is given by Torricelli's law:

$$\dot{x}_1 = \frac{1}{a_1}(u + v - k_1\sqrt{2gx_1}),$$

where a_1 and k_1 are tank specific parameters, g is the gravity constant, u is the inflow and v is a disturbance. The inflow u is chosen as $u = 0.1 + \kappa(4 - x_n)$ and x_n is the water level



(a) Reachable set.

(b) Number of computed sets per time step.

Fig. 3.16.: Van-der-Pol oscillator: Large initial set

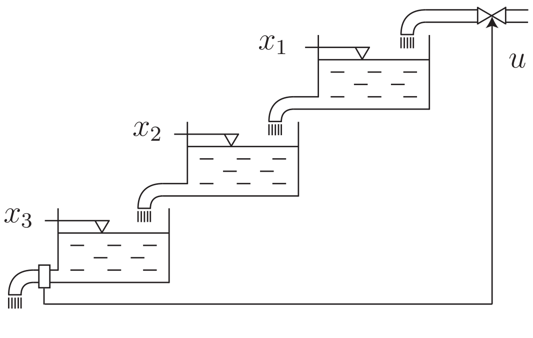
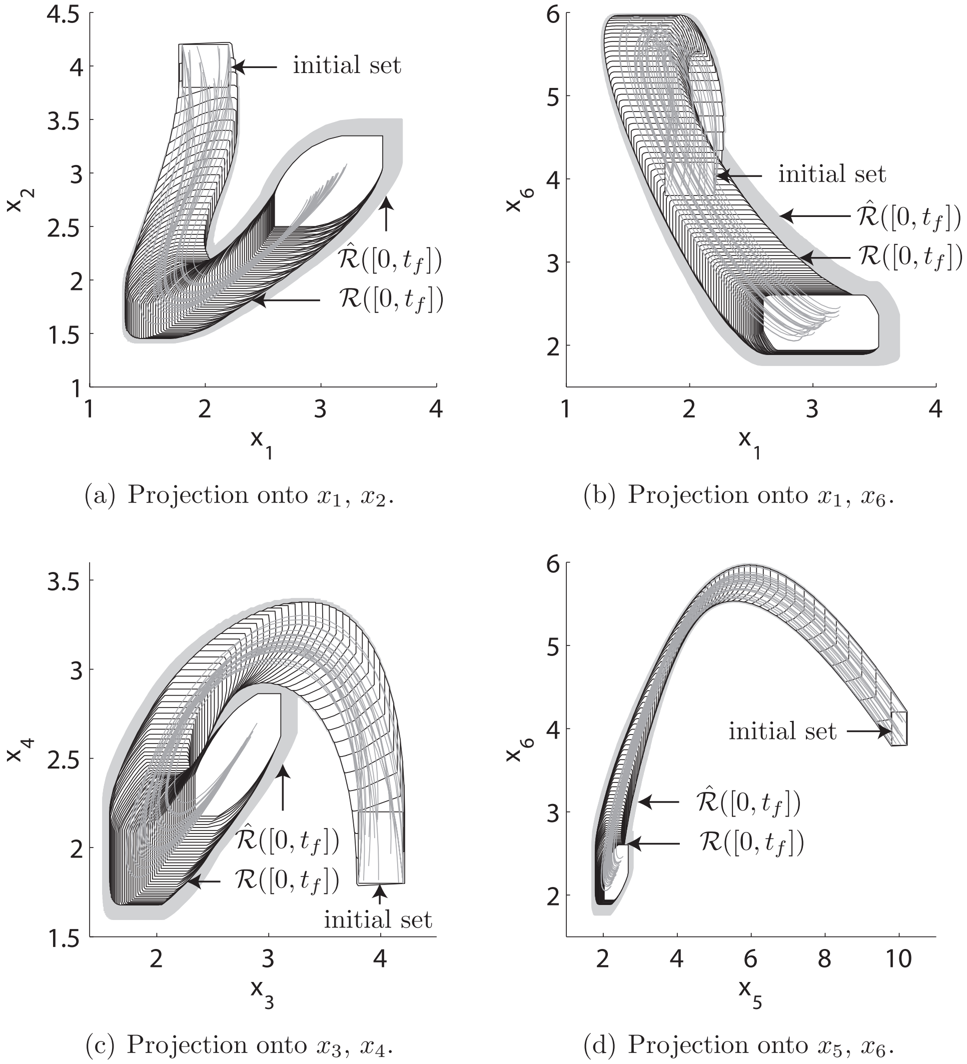


Fig. 3.17.: Water tank system.

of the last tank. The differential equation for the i-th tank is

$$\dot{x}_i = \frac{1}{a_i}(k_{i-1}\sqrt{2gx_{i-1}} - k_i\sqrt{2gx_i}).$$

For simplicity, all a_i are set to $a_i = 1$. The reachable set for $t \in [0, 400]$ with $v \in [-0.005, 0.005]$ is computed for a time step of $r = 4$ and an expansion vector of $\theta_i = 0.001$ for all coordinates i. The reachable set was computed for cases when the parameters are certain ($k_i = 0.015$) and for cases when they are uncertain ($k_i \in [0.0148, 0.015]$). Where the parameters are certain, the reachable set is denoted by $\mathcal{R}([0, t_f])$, and $\hat{\mathcal{R}}([0, t_f])$ when the parameters are uncertain. The reachable set for 6 tanks is shown in Fig. 3.18 for different two-dimensional projections together with exemplary trajectories starting from the initial set[3]. Computational times for different system dimensions using the same parameters and settings than for the 6-tank system are presented in Tab. 3.4. It is noted that no splits had to be performed for any water tank example. All computations were performed using Matlab on a desktop computer with an AMD Athlon64 3700+ processor (single core). It can be observed from Tab. 3.4 that the computation time moderately increases with the system dimension due to the efficient computation of reachable sets using zonotopes.



(a) Projection onto x_1, x_2.

(b) Projection onto x_1, x_6.

(c) Projection onto x_3, x_4.

(d) Projection onto x_5, x_6.

Fig. 3.18.: Reachable sets of the tank system.

3.5. Hybrid Systems

In many complex control systems, underlying processes are often controlled by continuous controllers while discrete or also called supervisory controllers are used for high level

[3] The exemplary trajectories are only computed for constant v values, although the values may be time varying.

Tab. 3.4.: Computational times.

Dimension n	5	10	20	50	100
No uncertain parameters					
CPU-time [s]	1.19	1.73	3.11	11.59	35.78
With uncertain parameters					
CPU-time [s]	6.83	12.92	28.94	119.58	523.56

control. The combination of the discrete control with the continuous dynamics of the underlying process forms a hybrid system. Typical examples of hybrid systems include chemical plants and driving assistance systems. For instance, the temperature of a liquid in a chemical plant is controlled by a continuous controller if the temperature is below a certain threshold. If the threshold of the exothermic reaction is exceeded, an emergency program is started, adding another liquid which slows down the exothermic reaction. Another example is a hybrid controller for adaptive cruise control (ACC) in passenger cars. In order to automatically keep a safe distance and speed, the controller has the modes *vehicle following* and *speed control*. The mode *vehicle following* keeps a safe distance to a vehicle driving ahead and is active when the distance to the leading vehicle is below a certain threshold. Otherwise, the mode *speed control* is active and keeps the desired speed.

In between mode switching, e.g. from *vehicle following* to *speed control*, the hybrid system behaves like a continuous system. For this reason, the approaches developed in the previous sections for continuous systems are also applied to hybrid systems. The necessary extensions for hybrid systems are concerned with the switching of the continuous dynamics. This includes the problem of determining which part of the continuous reachable set enters regions that enable the switching of the continuous dynamics.

However, before extending the algorithms for reachable set computations, the formal definition of hybrid dynamics by hybrid automata is addressed.

3.5.1. Hybrid Automaton

In this thesis, hybrid systems are modeled by hybrid automata. Clearly, besides a continuous state x, there also exists a discrete state y for hybrid systems. The continuous initial state may take values within continuous sets while only a single initial discrete state is assumed without loss of generality[4]. The switching of the continuous dynamics is triggered by so-called guard sets. Jumps in the continuous state are considered after the discrete state has changed. One of the most intuitive examples where jumps in the continuous state can occur is the bouncing ball example, where the velocity of the ball is instantaneously changed when hitting the ground.

The formal definition of the hybrid automaton is similarly defined as in [158]. The main difference is the consideration of uncertain parameters and the restrictions on jumps and

[4] In the case of several initial discrete states, the reachability analysis can be performed for each discrete state separately

guard sets:

Definition 3.3 (Hybrid Automaton): A hybrid automaton $HA = (Y, y^0, \mathcal{X}, \mathcal{X}^0, \mathcal{U}, \mathcal{P}, \texttt{inv}, \texttt{T}, \texttt{g}, \texttt{h}, \texttt{f})$, as it is considered in this thesis, consists of:

- the finite set of locations $Y = \{y_1, \dots, y_\xi\}$ with an initial location $y^0 \in Y$.
- the continuous state space $\mathcal{X} \subseteq \mathbb{R}^n$ and the set of initial continuous states $\mathcal{X}^0$ such that $\mathcal{X}^0 \subseteq \texttt{inv}(y^0)$.
- the continuous input space $\mathcal{U} \subseteq \mathbb{R}^m$.
- the parameter space $\mathcal{P} \subseteq \mathcal{I}^p$.
- the mapping[5] $\texttt{inv}$: $Y \to 2^{\mathcal{X}}$, which assigns an invariant $\texttt{inv}(y) \subseteq \mathcal{X}$ to each location y.
- the set of discrete transitions $\texttt{T} \subseteq Y \times Y$. A transition from $y_i \in Y$ to $y_j \in Y$ is denoted by (y_i, y_j).
- the guard function $\texttt{g} : \texttt{T} \to 2^{\mathcal{X}}$, which associates a guard set $\texttt{g}((y_i, y_j))$ for each transition from y_i to y_j, where $\texttt{g}((y_i, y_j)) \cap \texttt{inv}(y_i) \neq \emptyset$.
- the jump function $\texttt{h} : \texttt{T} \times \mathcal{X} \to \mathcal{X}$, which returns the next continuous state when a transition is taken.
- the flow function $\texttt{f} : Y \times \mathcal{X} \times \mathcal{U} \times \mathcal{P} \to \mathbb{R}^{(n)}$, which defines a continuous vector field for the time derivative of x: $\dot{x} = \texttt{f}(y, x, u, \rho)$.

The invariants $\texttt{inv}(y)$ and the guard sets $\texttt{g}((y_i, y_j))$ are modeled by polytopes. The jump function is restricted to a linear map

$$x' = K_{(y_i,y_j)}\, x + k_{(y_i,y_j)}, \tag{3.32}$$

where x' denotes the state after the transition is taken and $K_{(y_i,y_j)} \in \mathbb{R}^{n\times n}$, $k_{(y_i,y_j)} \in \mathbb{R}^n$ are specific for a transition (y_i, y_j). The input sets $\mathcal{U}_y$ are modeled by zonotopes and are also dependent on the location y. Note that in order to use the results from reachability analysis of nonlinear systems, the input $u(t)$ is assumed to be locally Lipschitz continuous. The set of parameters $\mathcal{P}_y$ can also be chosen differently for each location y. □

The evolution of the hybrid automaton is described informally as follows. Starting from an initial location $y(0) = y^0$ and an initial state $x(0) \in \mathcal{X}^0$, the continuous state evolves according to the flow function that is assigned to each location y. If the continuous state is within a guard set, the corresponding transition can be taken and has to be taken if the state would leave the invariant $\texttt{inv}(y)$. When the transition from the previous location y_i to the next location y_j is taken, the system state is updated according to the jump function and the continuous evolution within the next invariant.

The definition for the continuous reachable set in Def. 3.1 has to be enhanced for hybrid automata:

Definition 3.4 (Exact Continuous and Discrete Reachable Set): The continuous reachable set $\mathcal{R}^e$ and the discrete reachable set $\mathcal{R}^z$ of a HA for a given initial location

[5] $2^{\mathcal{X}}$ is the powerset of $\mathcal{X}$.

y^0 and a set of initial states $\mathcal{X}^0$ at time $t = r$ are:

$$\mathcal{R}^e(r) = \Big\{x(r)\big|\,(y(t),x(t)) \text{ is a solution of HA}\,\forall t\in[0,r], y(0)=y^0, x(0)\in\mathcal{X}^0\Big\},$$
$$\mathcal{R}^z(r) = \Big\{y(r)\big|\,(y(t),x(t)) \text{ is a solution of HA}\,\forall t\in[0,r], y(0)=y^0, x(0)\in\mathcal{X}^0\Big\},$$

where $y(t)$ and $x(t)$ denotes the discrete and continuous state at time t. □

Because the reachability of discrete states is simply a question of determining if the continuous reachable set hits certain guard sets, the focus below is on the continuous reachable sets. Clearly, as for the continuous systems, the reachable set $\mathcal{R}^e$ of the hybrid system has to be over-approximated by $\mathcal{R}(r) \supseteq \mathcal{R}^e(r)$ in order to verify the safety of the system. The over-approximated reachable set for a time interval is defined as in Def. 3.2 as $\mathcal{R}([0,r]) = \bigcup_{t\in[0,r]}\mathcal{R}(t)$. An illustration of a reachable set of a hybrid automaton is given in Fig. 3.19. Next, an overview of the procedure for computing reachable sets of hybrid systems is presented.

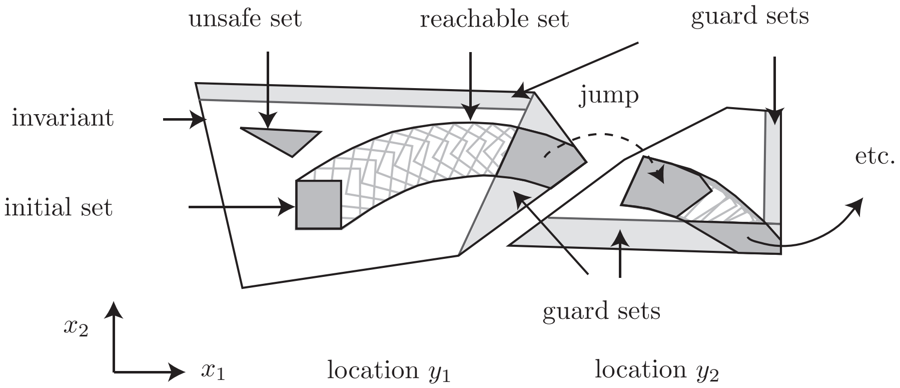


Fig. 3.19.: Illustration of the reachable set of a hybrid automaton.

3.5.2. Overview of Reachable Set Computations

An overview of how the reachability analysis of continuous systems can be extended to hybrid systems is presented in Fig. 3.20. It can be seen that the computation of the continuous reachable set within one location is completely encapsulated in the approach, such that one can refer to its computation in Sec. 3.2 – 3.4. One of the main extensions for hybrid systems is the consideration of guard sets and invariants. The extended approach is explained in more detail below:

① As a first step, the continuous reachable set is computed based on the set of continuous initial states $\mathcal{X}^0$ within the initial location y^0 as described previously in Sec. 3.2 - 3.4.
In addition to the computation of the reachable sets $\mathcal{R}([(k-1)r, kr])$ of time subintervals, it has to be checked that the reachable set has not left the invariant, i.e. if $\mathcal{R}([(k-1)r, kr]) \cap \texttt{inv}(y_i) \neq \emptyset$. The time when the invariant is left is denoted by $t^{end}_{\texttt{inv}}$.

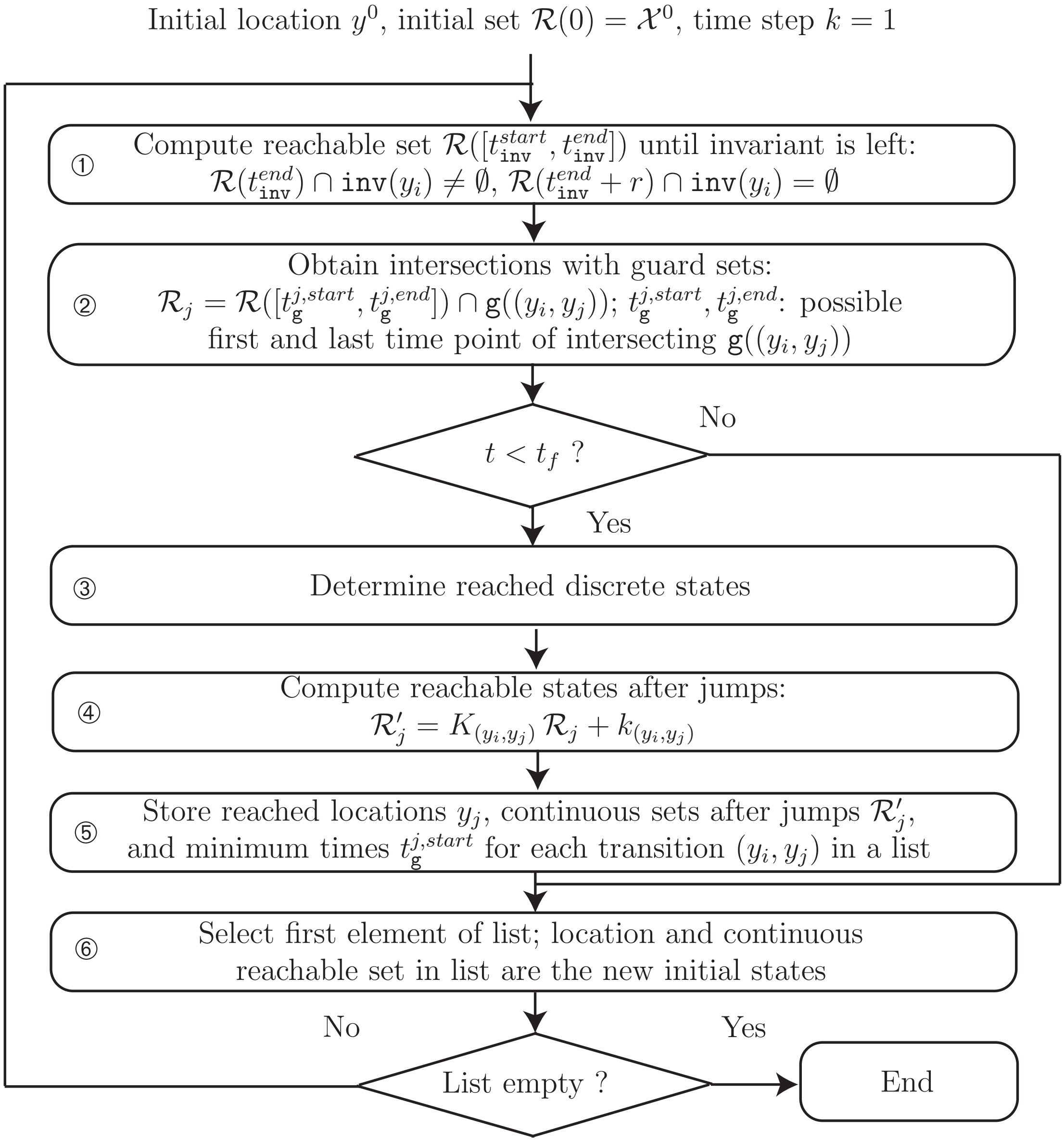

Initial location y^0, initial set $\mathcal{R}(0) = \mathcal{X}^0$, time step $k = 1$
①
Compute reachable set $\mathcal{R}([t^{start}_{\texttt{inv}}, t^{end}_{\texttt{inv}}])$ until invariant is left:
$\mathcal{R}(t^{end}_{\texttt{inv}}) \cap \texttt{inv}(y_i) \neq \emptyset$, $\mathcal{R}(t^{end}_{\texttt{inv}} + r) \cap \texttt{inv}(y_i) = \emptyset$
②
Obtain intersections with guard sets:
$\mathcal{R}_j = \mathcal{R}([t^{j,start}_{\texttt{g}}, t^{j,end}_{\texttt{g}}]) \cap \texttt{g}((y_i, y_j))$; $t^{j,start}_{\texttt{g}}, t^{j,end}_{\texttt{g}}$: possible
first and last time point of intersecting $\texttt{g}((y_i, y_j))$
$t < t_f$?
No
Yes
③
Determine reached discrete states
④
Compute reachable states after jumps:
$\mathcal{R}'_j = K_{(y_i,y_j)}\, \mathcal{R}_j + k_{(y_i,y_j)}$
⑤
Store reached locations y_j, continuous sets after jumps $\mathcal{R}'_j$,
and minimum times $t^{j,start}_{\texttt{g}}$ for each transition (y_i, y_j) in a list
⑥
Select first element of list; location and continuous
reachable set in list are the new initial states
No
Yes
List empty ?
End

Fig. 3.20.: Computation of reachable sets - overview.

② Once the reachable set has left the invariant, one has to check which guard sets have been hit and intersect the reachable set with them. As the reachable set is represented by zonotopes and the guard sets by polytopes, it is necessary to transform both types of sets into a halfspace representation in order to perform the intersection with standard software packages[6]. After the intersection with the guard set, a zonotope has to be found which encloses the intersection such that one can continue the computation of reachable sets with zonotopes.

③ The next location after the hit of a guard set is determined by the transition (y_i, y_j) associated with the guard set.

④ Besides the new location, the initial continuous reachable set of the new location has to be computed by the jump function: $\mathcal{R}'_j = K_{(y_i,y_j)}\, \mathcal{R}_j + k_{(y_i,y_j)}$ and $\mathcal{R}_j$ is the

[6] Used Matlab tool: MPT-Toolbox [105]

intersection of the reachable set with the guard set. The resulting set is again a zonotope as the mapping is restricted to linear functions (see (2.1)).

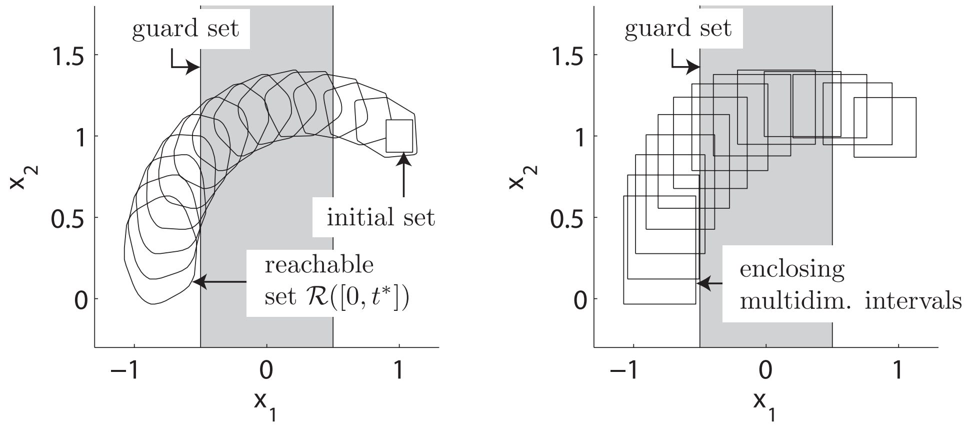
⑤ The reachable set within one location can hit several guard sets. However, one can only continue the computation for one location, such that the possible next locations y_j, the reachable sets after jumps $\mathcal{R}_j$, and the minimum time $t_{\mathrm{g}}^{j,start}$ for enabling transitions, have to be stored in a list.

⑥ The computation of reachable sets is continued with the first data structure of the list, containing the new initial location $y^0 = y_j$, the new set of initial states $\mathcal{X}^0 = \mathcal{R}_j$ and the initial time $t^0 = t_{\mathrm{g}}^{j,start}$. If the list is empty, the computations are terminated.

The most challenging part of the presented procedure is the intersection of the reachable set with guard sets and the enclosure of this intersection by a zonotope. This aspect is detailed in the following subsection.

3.5.3. Intersections of Zonotopes with Polytopes

Before reachable sets are intersected with guard sets, the subintervals $t \in [(k-1)r, kr]$ in which an intersection might take place have to be efficiently estimated. This is done by over-approximating the guard sets and the reachable sets by multidimensional intervals; see Fig. 3.21, using Prop. 2.3 for guard sets and Prop. 2.2 for reachable sets. The intersection check of multidimensional intervals is computationally inexpensive as it is simply checked whether the intervals for each dimension intersect. Only if there is an intersection for each dimension, the multidimensional intervals intersect. The resulting over-approximative time span of intersection is denoted by $[\tilde{t}_{\mathrm{g}}^{j,start}, \tilde{t}_{\mathrm{g}}^{j,end}]$, where the index j refers to the new location y_j after the transition of the guard set is taken.



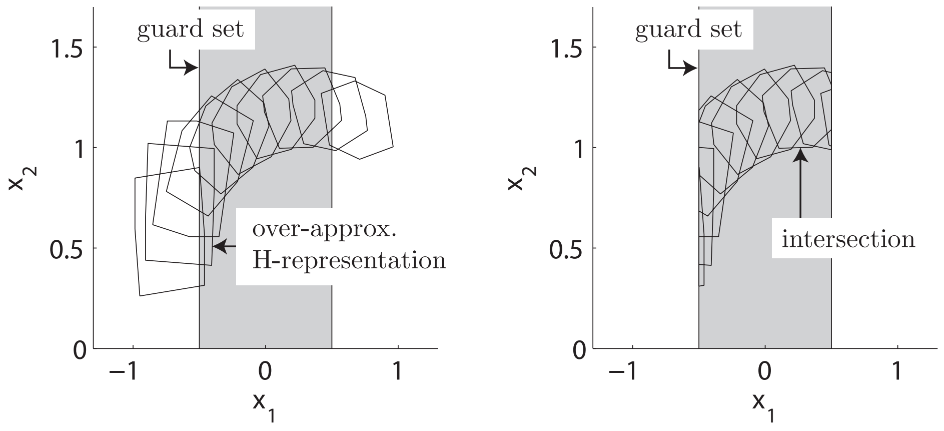
(a) Reachable sets.

(b) Enclosure by multidimensional intervals.

Fig. 3.21.: Over-approximation of the reachable set by multidimensional intervals.

For the obtained over-approximative time span $[\tilde{t}_{\mathrm{g}}^{j,start}, \tilde{t}_{\mathrm{g}}^{j,end}]$, the representation of reachable sets is changed from G- to H-representation. This allows the intersection with the guard set[7] to be computed by standard software packages[8], which is illustrated in Fig. 3.22. As the direct conversion of reachable sets from G- to H-representation is computationally demanding (see Theorem 2.1), over-approximative methods presented in Sec. 2.5.6 are applied.

The intersection of the H-representations allows the time interval of intersection $[t_{\mathtt{g}}^{j,start}, t_{\mathtt{g}}^{j,end}]$ to be narrowed, as some reachable sets only intersected when over-approximated by multidimensional intervals. The same principle is applied to determining the time $t_{\mathtt{inv}}^{end}$ when the reachable set leaves the invariant. First, multidimensional intervals are used to over-approximate the reachable sets and the invariant. This allows the conservative estimation $\tilde{t}_{\mathtt{inv}}^{end}$ to be obtained, which is refined to $t_{\mathtt{inv}}^{end} \leq \tilde{t}_{\mathtt{inv}}^{end}$ by checking intersections after conversion to H-representation backwards in time.



(a) H-representation.

(b) Intersection.

Fig. 3.22.: Over-approximative H-representation of the reachable set and its intersection with the guard set.

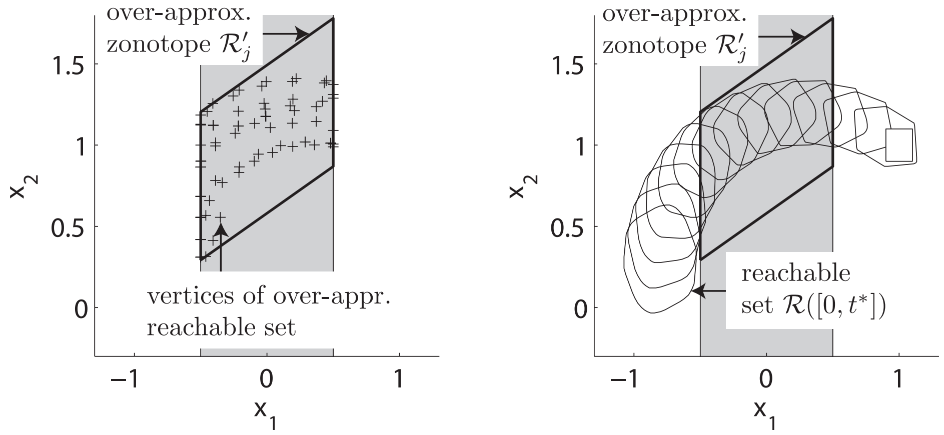
After the intersection of the reachable set with the guard set, the reachable set is represented by polytopes. In order to continue the computation of reachable sets with zonotopes, the obtained polytopes are enclosed by a zonotope as shown in Fig. 3.23, which is discussed later in detail. In Fig. 3.23, the enclosing zonotope is also plotted together with the reachable set, which allows the exactness of the enclosure to be assessed.

In order to formulate the overall procedure for the computation of reachable sets within one location, the following operators are introduced: The operator `reach`() returns continuous reachable sets as presented in Sec. 3.2 – 3.4. Further, the operator `intersectionTimes`() over-approximates the time interval of guard intersection, and the operators for the conversion to H- and G-representation are denoted by `halfspaceConv`() and `generatorConv`(), respectively. The superscript H of the reachable sets $\mathcal{R}^H$ indicates that the reachable set has a halfspace representation. The algorithm for computing reachable sets within one location is formulated in Alg. 5.

Note that there are two options for computing $\mathcal{R}([kr,(k+1)r]) = \mathtt{reach}(\ldots)$: one is to

[7] The guard set may be represented by a V- or a H-polytope. Where the guard set is given by a V-polytope, there exist standard algorithms for changing the representation to H-polytopes.

[8] Used Matlab tool: MPT-Toolbox [105]



(a) V-representation.

(b) Over-approximating zonotope.

Fig. 3.23.: Over-approximative V-representation of the reachable set and its enclosure by a zonotope.

compute the reachable set of the next time interval based on $\mathcal{R}([(k-1)r, kr])$. This has the disadvantage that the obtained reachable set may contain trajectories that leave and re-enter the invariant without changing the continuous dynamics, which causes an over-approximation. The other option is to compute the reachable set of the next time interval based on $\mathcal{R}^H([(k-1)r, kr])$. In this case, no trajectories are contained that may re-enter the invariant, however there is another over-approximation due to the halfspace conversion ($\mathcal{R}^H([(k-1)r, kr]) \supseteq \mathcal{R}([(k-1)r, kr])$).

Algorithm 5 Compute $\mathcal{R}'_j$ within a location y_i

```
Input: Invariant inv, guard set $\mathtt{g}((y_i, y_j))$, arguments for continuous reachable sets (see
  Sec. 3.2 - 3.4)
Output: $\mathcal{R}^H[0, t^{end}_{\mathtt{inv}}]$, $\mathcal{R}_j$
  $\mathcal{R}([0, r]) = \mathtt{reach}$(arguments for continuous reachable sets)
  $\mathcal{R}^H([0, r]) = \mathtt{halfspaceConv}\big(\mathcal{R}([0, r])\big) \cap \mathtt{inv}$
  $k := 1$
  while $\mathcal{R}^H([(k-1)r, kr]) \neq \emptyset$ or $t < t_f$ do
    $\mathcal{R}([kr, (k+1)r]) = \mathtt{reach}$(arguments for continuous reachable sets)
    $\mathcal{R}^H([kr, (k+1)r]) = \mathtt{halfspaceConv}\big(\mathcal{R}([kr, (k+1)r])\big) \cap \mathtt{inv}$
    $k := k + 1$
  end while
  $t^{end}_{\mathtt{inv}} = (k-1)r$
  $\mathcal{R}^H([0, t^{end}_{\mathtt{inv}}]) = \bigcup_{k=1}^{t^{end}_{\mathtt{inv}}/r} \mathcal{R}^H([(k-1)r, kr])$
  $[t^{j,start}_{\mathtt{g}}, t^{j,end}_{\mathtt{g}}] = \mathtt{intersectionTimes}\big(\mathcal{R}^H([0, t^{end}_{\mathtt{inv}}]), \mathtt{g}((y_i, y_j))\big)$
  $\mathcal{R}_j = \mathtt{generatorConv}\big(\mathcal{R}^H([t^{j,start}_{\mathtt{g}}, t^{j,end}_{\mathtt{g}}]) \cap \mathtt{g}((y_i, y_j))\big)$
```

3.5.4. Enclosure of Polytopes by a Zonotope

This subsection deals with the challenge of finding a suitable over-approximating zonotope for several polytopes $\mathcal{P}_1, \ldots, \mathcal{P}_\mu$. This is done according to Prop. 2.4, where a polytope is enclosed by a parallelotope: $\mathcal{Z}^{encl} = \Lambda\, \mathtt{box}(\Lambda^{-1}\, \mathcal{P})$. The enclosure is restricted to parallelotopes ($\hat{=}$ zonotope of order 1) since the enclosure by a general zonotope is too complicated. In Prop. 2.4 only a single polytope $\mathcal{P}$ is enclosed. However, since the computation is based on the vertices of the polytope, one can extend the approach to the enclosure of many polytopes by computing with the union of vertices of all polytopes.

It remains to find a good solution for the linear transformation by Λ^{-1}. The column vectors of Λ determine the direction of the generators of the enclosing parallelotope. The problem can be reformulated to finding a zonotope tightly enclosing points in $\mathbb{R}^n$. Much previous work solved the problem of finding tightly enclosing oriented boxes in three-dimensional space; see e.g. [17]. For n dimensions, enclosures of points have been investigated in [158].

Two different techniques for finding the matrix Λ are suggested: One which takes the direction of the flow vector $\mathtt{f}(y, x, u, \rho)$ of the hybrid automaton into account and the one in [158], which is purely based on the distribution of the vertices of the polytopes $\mathcal{P}_i$. First, the approach considering the flow vector is presented, where the flow vector after the transition is taken, is considered. The other possibility of choosing the flow vector before the transition is taken is discussed later. The direction $\hat{\rho}$ of the flow vector is uncertain within the reachable set, the input set and the parameter set. As a good estimate, the flow vector is computed at the centers of the input, the parameter set, and the enclosing box of the polytopes $\mathcal{P}_i$:

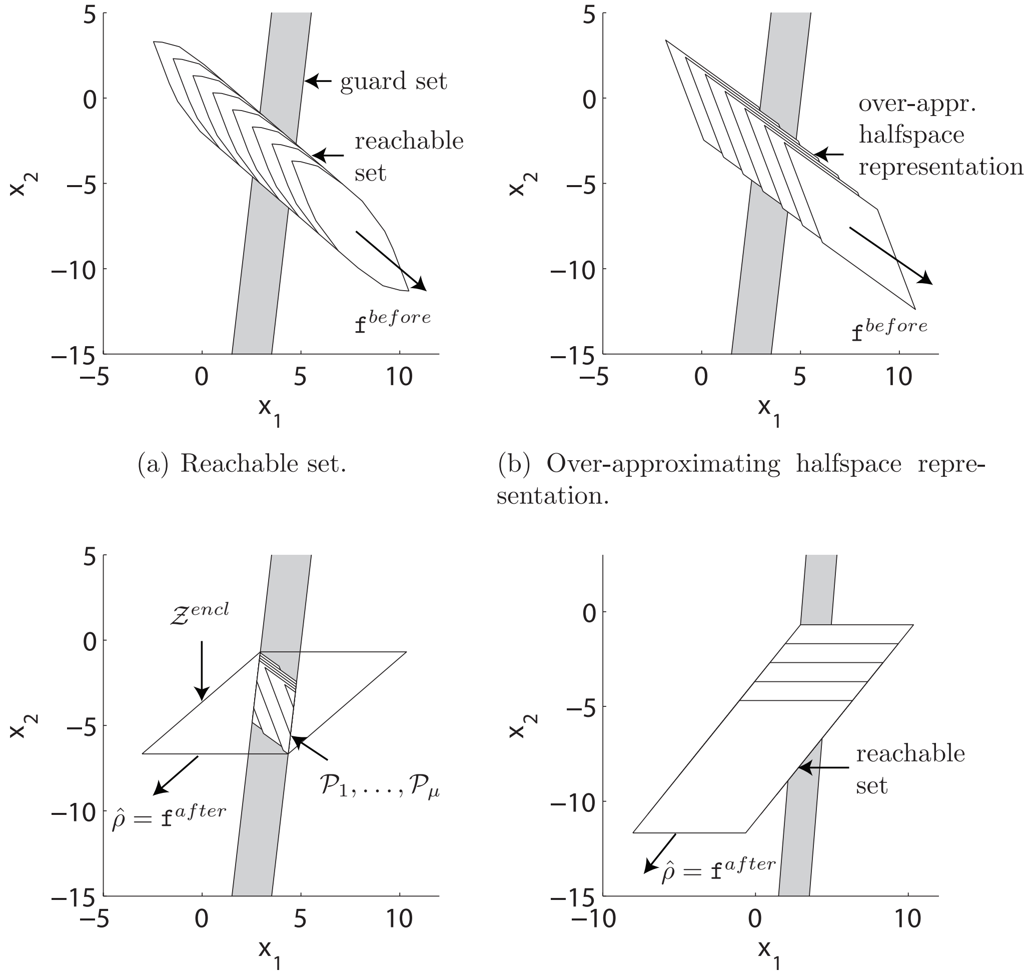
$$\hat{\rho} = \frac{\mathtt{f}(y_i, x^*, u^*, \rho^*)}{\|\mathtt{f}(y_i, x^*, u^*, \rho^*)\|_2},$$
$$x^* = \mathtt{center}\left(\mathtt{box}(\mathcal{P}_1, \ldots, \mathcal{P}_\mu)\right),\ u^* = \mathtt{center}(\mathcal{U}_{y_i}),\ \rho^* = \mathtt{center}(\mathcal{P}_{y_i}),$$

where $\mathtt{box}()$ is defined as in Prop. 2.3 and $\mathtt{center}()$ returns the volumetric center of a set. By choosing $\hat{\rho}$ as one of the column vectors of Λ, the enclosing zonotope is oriented in the direction of the flow vector. The remaining column vectors are chosen as $n-1$ unit vectors $e^{(j)}$ of the coordinate system, where $e^{(j)}_i = 1$ if $j = i$ and $e^{(j)}_i = 0$ otherwise. The replaced unit vector $e^{(j)}$ is the one that is best aligned with $\hat{\rho}$, such that:

$$\Lambda = [\ldots, e^{(j-1)}, \hat{\rho}, e^{(j+1)}, \ldots], \quad |(e^{(j)})^T\, \hat{\rho}| \geq |(e^{(m)})^T\, \hat{\rho}|,\ \forall m = 1 \ldots n.$$

The procedure for the enclosure of the polytopes is illustrated for a simple two-dimensional example, where $\mathtt{f}^{before} = [1, -1]^T$ and $\mathtt{f}^{after} = [-1, -1]^T$ are the constant flow vectors before and after the transition without a jump. The reachable sets are illustrated in Fig. 3.24.

Another method, which does not consider the flow direction, has been proposed in [158]. This method is based on applying a principal component analysis (PCA) on the set of the vertices of all polytopes $\mathcal{P}_1, \ldots, \mathcal{P}_\mu$. If one interprets the vertices as measured data points, PCA generates a new orthogonal coordinate system such that the greatest variance of the data is in the direction of the first coordinate, the second greatest variance in the direction of the second coordinate, and so on. The principal component analysis is performed as



(a) Reachable set.

(b) Over-approximating halfspace representation.

(c) Intersection with guard set.

(d) Reachable set of the next location.

Fig. 3.24.: Enclosure of the reachable set by a single zonotope.

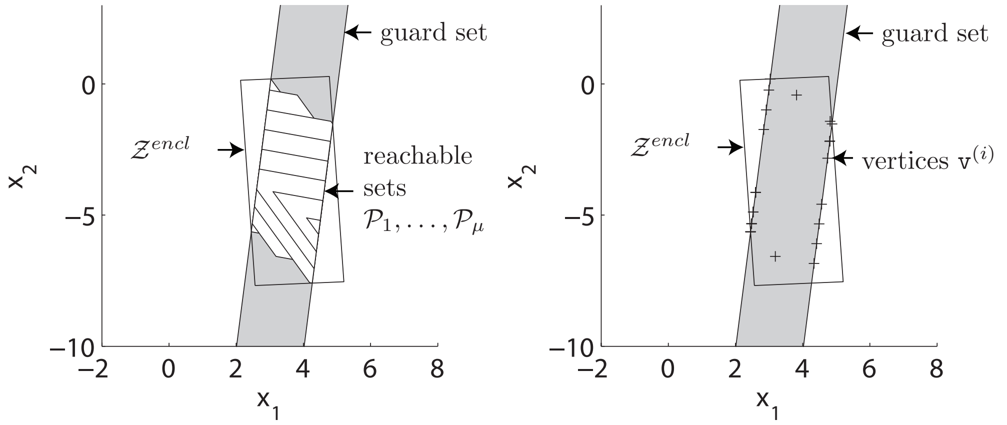
shown below.

First, the mean value $\mathtt{v}^m$ of all vertices $\mathtt{v}^{(1)}, \dots, \mathtt{v}^{(r)}$ is determined: $\mathtt{v}^m = \frac{1}{r}\sum_{i=1}^{r} \mathtt{v}^{(i)}$. The vertices are then translated by the mean value $\mathtt{v}^m$ such that $\bar{\mathtt{v}}^{(i)} = \mathtt{v}^{(i)} - \mathtt{v}^m$ and stored in a matrix: $\bar{\mathtt{V}} = \left[\bar{\mathtt{v}}^{(1)}, \dots, \bar{\mathtt{v}}^{(r)}\right]$. Next, the covariance matrix of the set of vertices is computed:

$$\mathtt{cov}(\bar{\mathtt{V}}) = \frac{1}{r-1}\,\bar{\mathtt{V}}\,\bar{\mathtt{V}}^T = U\,\Sigma\,V^T,$$

where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices that can be obtained from a singular value decomposition. The transformation matrix is $\Lambda = U = V$ since $\mathtt{cov}(\bar{\mathtt{V}})$ is symmetric. An example of the enclosure of intersected reachable sets by this method is given in Fig. 3.25.

In the next subsection, it is outlined how the alternative approaches can be integrated into an overall approach.



(a) Intersected reachable set. (b) Vertices of intersected reachable set.

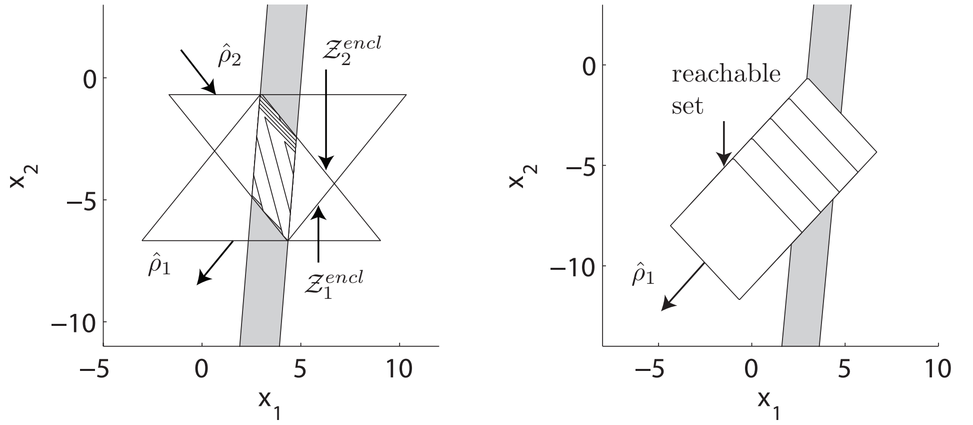
Fig. 3.25.: Enclosure of the reachable set by principal component analysis (PCA).

3.5.5. Computing with Several Enclosing Zonotopes

In order to combine the advantages of different approaches for the enclosure of points, one can compute several alternative enclosures $\mathcal{Z}_1^{encl}$, ..., $\mathcal{Z}_\omega^{encl}$. From this follows that starting from these enclosures, many reachable sets have to be computed in parallel within the active location. The parallel computation of reachable sets has already been applied to the reachability analysis of nonlinear continuous systems in Sec. 3.4. However, in this case, the reachable set is not split, but all reachable sets over-approximate the exact reachable set. From this follows that an unsafe set is only reached if all reachable sets (of a common time interval) intersect the unsafe set.

The slightly modified procedure for the computation of the intersection with guard sets is explained for the case where only two enclosing zonotopes $\mathcal{Z}_1^{encl}$, $\mathcal{Z}_2^{encl}$ are used (which can be extended in a straightforward way to more enclosing zonotopes). One difference to the non-parallel computation is that the guard set is only reached if it is intersected with both reachable sets $\mathcal{R}_1([(k-1)r, kr])$, $\mathcal{R}_2([(k-1)r, kr])$. The polytopes obtained from the intersection with a guard set are denoted by $\mathcal{P}_1([t_{\mathrm{g},1}^{j,start}, t_{\mathrm{g},1}^{j,end}])$ and $\mathcal{P}_2([t_{\mathrm{g},2}^{j,start}, t_{\mathrm{g},2}^{j,end}])$. Because of the parallel computation, the combined entry and exit time of the guard set are $t_{\mathrm{g}}^{j,start} = \max(t_{\mathrm{g},1}^{j,start}, t_{\mathrm{g},2}^{j,start})$ and $t_{\mathrm{g}}^{j,end} = \min(t_{\mathrm{g},1}^{j,end}, t_{\mathrm{g},2}^{j,end})$. Further, the combined polytopes within a guard set are: $\mathcal{P}([(k-1)r, kr]) = \mathcal{P}_1[(k-1)r, kr] \cap \mathcal{P}_2[(k-1)r, kr]$. After obtaining the polytopes $\mathcal{P}([(k-1)r, kr])$, the enclosure of the reachable sets is performed as shown in the previous subsection.

The tighter enclosure of the reachable set when using two enclosing zonotopes is demonstrated for the previous example shown in Fig. 3.24. The additional zonotope is obtained by the same method, but the flow vector of the previous instead of the next location is used. Both enclosing zonotopes, as well as the intersected reachable sets $\mathcal{R}_1([(k-1)r, kr]) \cap \mathcal{R}_2([(k-1)r, kr])$, are illustrated in Fig. 3.26.



(a) Intersection with guard set.

(b) Reachable set of the next location.

Fig. 3.26.: Enclosure of the reachable set by two zonotopes.

3.5.6. Numerical Example

The presented techniques are applied to a benchmark example proposed in [62]. It considers a room heating scenario, where in each room there is one or no heater. In contrast to the proposed benchmark example, it is not considered that heaters can be moved from one room into another. Extensions of the proposed example are that the input is modeled uncertain and that the switching is not deterministic. The example considered in this work consists of 6 rooms with heaters in the rooms 1 and 6; see Fig. 3.27. The heaters are switched on if the temperature drops below $T^{low} + \Delta T$ and switched off if the temperature exceeds $T^{high} - \Delta T$ with $T^{low} = 22$°C, $T^{high} = 24$°C and $\Delta T = [0, 0.05]$°C. The temperature dynamics in room i is

$$\dot{x}_i = c \cdot h_i + b_i(u - x_i) + \sum_{i \neq j} a_{ij}(x_j - x_i)$$

with constants a_{ij}, b_i and c. The rate of heat exchange a_{ij} between two adjacent rooms is 1. The transfer rate b_i from inside the building to the outside is 0.16 for rooms at corners and 0.08 for other rooms. The outside temperature u is in the interval of $[0, 0.05]$, and the heating power is $c = 15$ for both heaters. The variable h_i is 1 if a heater is switched on in room i and 0 otherwise. The reachable sets are computed for the time interval $t \in [0, 1]$.

The over-approximation of the zonotopes to a halfspace representation has been performed using method HI_1 in Sec. 2.5.6. It is recalled that method HI_1 is based on the order reduction method C for which the following settings have been used: $\kappa = 5$ for the preselection of generators by length and $\tilde{\lambda} = 3$ for preselected combinations of generators. The enclosure of the polytopes within the guard sets has been performed by two zonotopes as explained in Sec. 3.5.5, where the flow vector of the location before and after the transition is taken, has been used. The time increment has been set to $r = 0.01$ and the computation time was 16.8 seconds on a desktop computer with an AMD Athlon64 3700+ processor (single core) for an implementation in Matlab.

In order to assess the halfspace conversion of this example, the zonotopes converted to halfspace representation have been recorded. As the uncertain input is small compared to the uncertainty arising from the switching of the system dynamics, the recorded zonotopes have only a few dominant generators, so that the halfspace conversion can be performed with high accuracy. The distribution of the generator length is shown in the histogram in Fig. 3.28. In Tab. 3.5, the mean values of the relative over-approximation index (υ/υ^C) for different methods are presented, where υ^C is the over-approximation index of method C. The reason for the relative over-approximation index is that the absolute value cannot be computed – this requires the computation of the volume of the zonotopes of dimension 6 and order 10, which is intractable.

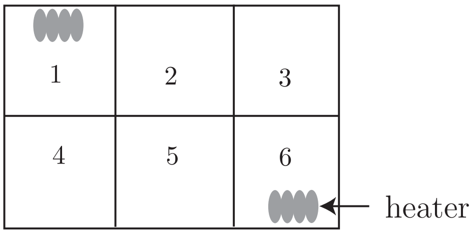


Fig. 3.27.: Room layout.

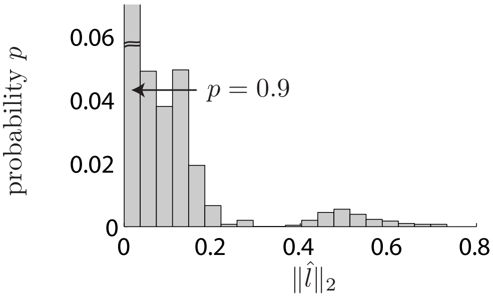


Fig. 3.28.: Histogram of generator length $\|\hat{l}\|_2$.

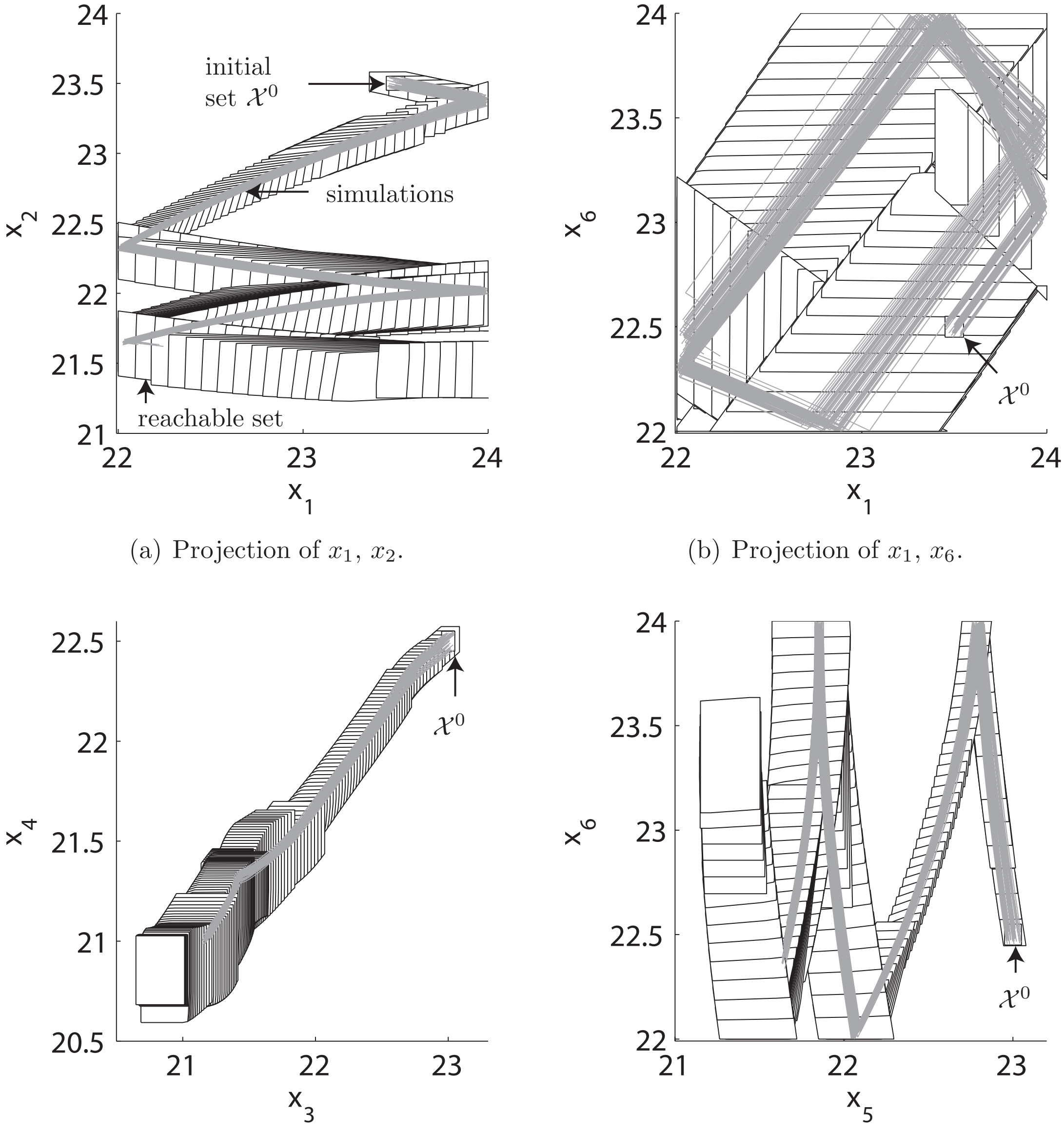
Tab. 3.5.: Results for the conversion to halfspace representation using zonotopes from the room heating example.

	method B	method HI$_1$	method HD$_2$	method HI$_2$
mean of	($\kappa = 3$)	($\kappa = 8$, $\tilde{\lambda} = 3$)	($\kappa = 8$, $\tilde{\lambda} = 3$)	($\kappa = 8$, $\tilde{\lambda} = 3$)
Θ/Θ^C:	1.0001	1.0000	1.0581	0.9836
t^{comp} [s]:	0.0528	0.0979	0.2262	0.1191

The reachable sets are shown for selected projections in Fig. 3.29. Exemplary trajectories within the reachable set are plotted in gray and were computed for randomized initial states and inputs. In comparison with the trajectories, the reachable sets seem to be computed for a longer time horizon. This is because the point in time when a transition is enabled by a guard $\mathsf{g}((y_i, y_j))$ is uncertain within $[t_{\mathsf{g}}^{j,start}, t_{\mathsf{g}}^{j,end}]$, as discussed in Sec. 3.5.3, so that the time uncertainty increases with each discrete transition. A possible verification scenario would be to analyze whether a certain combination of room temperatures is enabled (or avoided) by the switching controller.

3.6. Summary

In this chapter, reachable sets were computed for continuous and hybrid systems. All presented approaches use zonotopes as a set representation, compute with uncertain inputs, and are based on the computation of reachable sets for linear time invariant (LTI) systems.



(a) Projection of x_1, x_2.

(b) Projection of x_1, x_6.

(c) Projection of x_3, x_4.

(d) Projection of x_5, x_6.

Fig. 3.29.: Reachable sets of the room heating scenario.

Due to the importance of LTI systems, known methods have been recapitulated. It is noteworthy that reachable sets of LTI systems can be computed without the wrapping-effect when applying the algorithm presented in [72].

Known approaches for LTI systems have been extended to linear systems with constant, but uncertain parameters. Two different representations of uncertain parameters in the system matrix of the linear system have been investigated: interval matrices and matrix zonotopes. In addition, it has been shown how to abstract the more general case of matrix polytopes to interval matrices or matrix zonotopes. A possible future direction is the investigation of linear systems where the parameters are time varying.

The computation of reachable sets with zonotopes has been extended to nonlinear systems by conservative linearization, i.e. the linearization error is added as an additional uncertain input. There are two reasons that motivate the linearization of the system dynamics: The superposition principle is applicable and sets of future points in time can be obtained by linear transformation so that zonotopes are mapped to zonotopes. The set of linearization errors is obtained via interval arithmetics, which allows the determination of the errors for arbitrary nonlinear systems that are Lipschitz continuous. A future issue is to avoid the required over-approximation of zonotopes before a split. This could possibly be resolved by not splitting zonotopes, but computing with families of zonotopes such that the sets A and B enclose a zonotope $A \cup B \supseteq \mathcal{Z}$ and further define the split zonotopes by $\mathcal{Z}_1 = A \cap \mathcal{Z}$ and $\mathcal{Z}_2 = B \cap \mathcal{Z}$. The numerical results for continuous systems have shown that systems with up to 100 continuous state variables can be handled. This is not achievable for nonlinear systems when the reachable set has to be split numerous times.

The approaches for continuous systems have been extended to hybrid systems by additionally considering the intersection with guard sets. Thereby, the difficulty is that the intersection of guard sets with zonotopes does not yield a zonotope in general, even if the guard set would be represented by a zonotope. For this reason, zonotopes are over-approximated by a halfspace representation so that the intersection with guard sets can be obtained. The resulting intersections have to be over-approximated by a zonotope to continue the computation with zonotopes. In order to minimize the over-approximation by the enclosure of a single zonotope, the computation with several zonotopes in parallel has been pointed out. The results for the conversions to halfspace representation are already quite promising. There is more potential for the over-approximation of polytopes by a single zonotope. Due to this complicated task, the enclosure has been restricted to parallelotopes (zonotopes of order 1). This problem can be reformulated to enclosing a set of points in $\mathbb{R}^n$ by a parallelotope so that the volume of the parallelotope is minimized. Even for hyperrectangles (a special case of parallelotopes), this problem is unsolved in literature to the best knowledge of the author.

4. Stochastic Reachability Analysis

The previously introduced approaches for reachability analysis of continuous and hybrid systems are extended to approaches for stochastic systems. Stochastic reachability analysis not only answers the question of whether the system is safe or not, but also returns the probability for the safe operation of the system. This is especially important for applications that are inherently unsafe, such as road traffic, which will be investigated later.

4.1. Introduction and State of the Art

Continuous stochastic systems can be described by stochastic differential equations, where the derivative of random variables is computed in part by a deterministic drift term and a stochastic diffusion term [67, 128]. Discrete stochastic systems can be described by stochastic automata, where the transitions from one discrete state to the other is given by a probability which is possibly depending on a finite set of discrete inputs [36]. One of the best known examples of a stochastic automaton is the Markov chain, which will be introduced in more detail later in this chapter.

Combining continuous and discrete stochastic systems to stochastic hybrid systems results in a vast variety of possible definitions. This is because one can introduce stochastic dynamics in many parts of a standard hybrid automaton (see Def. 3.3): In transitions, guard functions, jump functions, or flow functions. Depending on which parts of the hybrid automaton are kept deterministic, many definitions are possible:

- Piecewise deterministic processes are stochastic hybrid systems with deterministic continuous evolution [53].
- Switching diffusion processes are stochastic hybrid systems with spontaneous transitions, but there exist no forced transitions and reset functions [68].
- Stochastic hybrid systems in the sense of [85] do not model spontaneous transitions.
- Generalized stochastic hybrid systems introduced in [33] is the most general class which models stochastic continuous evolution, forced transitions, continuous resets, and spontaneous transitions.

In terms of safety verification for stochastic dynamic systems, guaranteed results for large-scale systems have only been obtained for purely discrete stochastic systems [95] which extend the classical model checking approach [16, 41].

For large-scale hybrid systems, the only possible method for safety verification is Monte Carlo simulation [147]. The disadvantage of Monte Carlo simulation is that it cannot give guarantees or upper bounds for the probability that an unsafe set is hit. The advantage of Monte Carlo simulation is that the variance of the result depends on the number of simulations and not on the size of the continuous state space. This explains the applicability to large-scale stochastic hybrid systems. More details on the convergence results of Monte Carlo simulation can be found later in Sec. 5.7. It can also be shown that the probability of safety converges to the exact result when the number of simulations goes to infinity [147].

Another possibility to compute the stochastic reachable set of a hybrid system is to abstract its dynamics to a Markov chain. Different methods for the abstraction to Markov chains have been proposed. Based on the finite difference method [104], Markov chain abstractions have been obtained in [98, 140, 141]. The generation of Markov chains via Monte Carlo simulation or classical reachability analysis has been performed in [111, 150].

Stochastic reachability analysis has been reformulated to a stochastic optimal control problem in [1–3]. The optical control problem is solved via dynamic programming, and practical solutions are realized via a discretization of time and the continuous state space. From this follows that this approach as well as the approaches abstracting to Markov chains suffer from the curse of dimensionality because the state space has to be discretized, which results in an explosion of the number of discrete states.

Theoretical contributions for the reachability problem of stochastic hybrid systems make use of Dirichlet forms [34] and Bayesian statistical inference [35]. As already mentioned in the introduction of this thesis, upper bounds on the probability of reaching an unsafe set can be obtained via barrier certificates [139].

Contributions

In this thesis, two different approaches for the reachability analysis of stochastic systems are investigated. First, a novel method for linear systems with uncertain input $u \in \mathcal{U}$ and Gaussian white noise input $\boldsymbol{\xi}$ is presented. This approach is an extension of Sec. 3.2, since Gaussian white noise is additionally considered. Thus, a stochastic input $\boldsymbol{\xi}$ is mixed with an uncertain input u for which no stochastic information is given. A specialty of the proposed approach is that an upper bound for being in an unsafe set can be guaranteed. To the best knowledge of the author, upper bounds have only been computed by barrier certificates so far [139]. Note that barrier certificates compute an upper bound for *reaching* an unsafe set while the presented approach computes an upper bound for *being in* an unsafe set – this difference is explained in detail later. A further property is that the approach inherits the computational efficiency of the approach for linear systems presented in Sec. 3.2 so that it can be applied to systems with more than 100 continuous state variables. One of the reasons for the efficiency is that the approach does not require any discretization of the continuous state space. The guarantee of a probability bound is achieved by enclosing possible probability density functions by so-called enclosing probabilistic hulls. This approach is presented in detail in Sec. 4.2 and is based on the work previously published in [189].

The second approach abstracts continuous or hybrid systems to Markov chains as it has been done in [111, 150], but with certain extensions: The reachability analysis used for the abstraction is more elaborate, and the abstraction is computed for adjacent time intervals instead of points in time which allows one to consider all times instead of sampled times.

Two abstraction techniques are addressed: Abstraction via Monte Carlo simulation and via reachability analysis. Advantages as well as disadvantages are highlighted. This approach is presented in Sec. 4.3 and is later used for the safety assessment of autonomous cars in Chap. 5. Since the Markov chains have to be updated during the operation of the autonomous car, the Markov chain updates have to be efficient. In order to adjust the Markov chain to the application, the state space discretization has to be chosen independently of the time discretization. This is not possible for the finite difference method. There, the transition probabilities from one cell to a neighboring cell are chosen such that the solution converges weakly to the exact solution of the original stochastic differential equation. Since only neighboring cells are considered, the state space and time discretization are coupled [98, 140, 141], which makes this approach unsuitable for the time critical autonomous car application. The drawback of the Markov chain abstraction is that it can only be applied to hybrid systems with a few (up to $3-5$) continuous state variables. Markov chain abstraction using reachability analysis has been presented by the author in works on safety assessment of autonomous cars [179, 181–183, 185, 186, 188, 190].

In summary, the Markov chain abstraction is applicable to a general class of stochastic hybrid systems, but can only be applied to problems with up to $3-5$ continuous state variables due to the discretization. This is in contrast to the enclosing probabilistic hull approach, which is only applicable for linear continuous systems but allows results for large instances of this class to be computed. Further, the enclosing probabilistic hull approach can guarantee results, while the probabilities of the Markov chain abstraction are approximated.

Definitions of Stochastic Reachable Sets

In the works on stochastic reachability referenced so far [1–3, 34, 35, 98, 139–141], stochastic reachability analysis is defined based on the event that the state of the system reaches an unsafe set.

$$\begin{aligned} P\big(\mathrm{reach}_{t_f}\big) &= P\big(\{\omega \in \Omega | \exists t \in [0, t_f] : x(t,\omega) \in \mathcal{X}^{\mathrm{unsafe}}\}\big), \\ P\big(\mathrm{reach}_{\infty}\big) &= P\big(\{\omega \in \Omega | \exists t \geq 0 : x(t,\omega) \in \mathcal{X}^{\mathrm{unsafe}}\}\big), \end{aligned} \tag{4.1}$$

where ω is an elementary event and Ω is the set of elementary events.

The probability for the finite time horizon $P(\mathrm{reach}_{t_f})$ is computed straightforward when applying Monte Carlo simulation with importance sampling: $P(\mathrm{reach}_{t_f}) = N_{\mathcal{X}^{\mathrm{unsafe}}}/N_s$, where $N_{\mathcal{X}^{\mathrm{unsafe}}}$ is the number of simulations that reach $\mathcal{X}^{\mathrm{unsafe}}$ and N_s is the total number of simulations. However, this definition does not generalize to the definition of a reachable set in Def. 3.1. For this reason, a stochastic reachable set of a continuous system for a point in time is used as a synonym for the probability density function (PDF) of the state.

Definition 4.1 (Stochastic Reachable Set of a Point in Time): Given is a random process $\mathbf{x}(t)$ of a system. The stochastic reachable set at a certain point in time r is defined as the probability density function $f_{\mathbf{x}}(x, t = r)$ of the random state vector $\mathbf{x}$ at $t = r$. □

The stochastic reachable set for a time interval $t \in [0, r]$ is based on the definition for a point in time.

Definition 4.2 (Stochastic Reachable Set of a Time Interval): The stochastic reachable set of a time interval $[0, r]$ is defined as

$$f_{\mathbf{x}}(x, [0, r]) = \int_0^r f_{\mathbf{x}}(x, t) \cdot f_{\mathbf{t}}(t) dt, \quad f_{\mathbf{t}}(t) = \begin{cases} 1/r \text{ for } \mathbf{t} \in [0, r], \\ 0 \text{ otherwise,} \end{cases}$$

where $\mathbf{t}$ is a random variable which is uniformly distributed within the time interval. □

For hybrid system, the reachable set has to be complemented by the probability that the discrete state is in a certain location, as has been done for the classical reachable sets in Def. 3.4. The above definitions allow the following conversion.

Proposition 4.1 (Reachable Set of a Stochastic Reachable Set): The reachable set of a stochastic reachable set is the set for which the probability values of the stochastic reachable set are non-zero:

$$\mathcal{R}(t) = \{x | f_{\mathbf{x}}(x, t) > 0\}.$$ □

The probability that a state is in the set of unsafe states $\mathcal{X}^{\text{unsafe}}$ is computed as

$$P(x \in \mathcal{X}^{\text{unsafe}}, t \in [0, r]) = \int_{\mathcal{X}^{\text{unsafe}}} f_{\mathbf{x}}(x, [0, r]) \, dx. \tag{4.2}$$

This is different to the computation in (4.1), where one is interested in the probability that a state has entered the unsafe state during a certain time interval. The results of (4.1) and (4.2) are identical if the sets of unsafe states are absorbing; i.e. once a trajectory has entered a set of unsafe states, it cannot leave this set anymore. A set of unsafe states $\mathcal{X}^{\text{unsafe}}$ can be made absorbing by modeling it as an invariant of a hybrid automaton without any transitions to other locations. This can be achieved e.g. by setting the dynamics to $\dot{x} = 0$ within the unsafe set.

The definition of an over-approximative stochastic reachable set, which is also called enclosing probabilistic hull, is introduced next. It allows the computation of the over-approximated probability that the state is in an unsafe set. This concept is demonstrated for linear continuous systems.

4.2. Enclosing Hulls of Probability Density Functions for Linear Systems

Instead of computing the probability density function (PDF) of stochastic systems, the attempt to compute the enclosing hull of probability density functions is pursued in this section. A set of possible probability density functions occurs when the input of a system is uncertain, but no probabilistic distribution of this input is known. One can think of an enclosing probabilistic hull as an envelope for possible probability density functions. This allows the concept of over-approximation to be applied to stochastic reachable sets (which have been defined as probability density functions in Def. 4.1). In a first attempt, the concept of enclosing probabilistic hulls has been developed for linear stochastic systems with two different inputs. One input $u(t)$ can take values within a set $\mathcal{U}$ as it has been defined for the systems in Sec. 3.2-3.5. Note that no information about the probability of a given trajectory $u(t)$ is known. The other input $\boldsymbol{\xi}(t)$ is a Gaussian white noise signal. The system under consideration is

$$\begin{gathered}\dot{\mathbf{x}} = A\mathbf{x}(t) + u(t) + C\boldsymbol{\xi}(t), \\ \mathbf{x}(0) : \Omega \to \mathbb{R}^n,\ u(t) \in \mathcal{U} \subset \mathbb{R}^n,\ \boldsymbol{\xi} : \Omega \to \mathbb{R}^m,\end{gathered} \tag{4.3}$$

where $A \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{n\times m}$ and $\mathbf{x}$, $\boldsymbol{\xi}$ are random vectors which are functions from the set of elementary events Ω to real valued vectors. The given linear stochastic differential equation is also known as the multivariate Ornstein-Uhlenbeck process [67]. The combination of both inputs $u(t)$ and $C\boldsymbol{\xi}(t)$ can be seen as white Gaussian noise whose mean is unknown within the set $\mathcal{U}$. In order to handle this kind of system, the concept of enclosing probabilistic hulls is introduced.

Definition 4.3 (Enclosing Probabilistic Hull): The enclosing probabilistic hull of all possible probability density functions $f_{\mathbf{x}}(x, t = r)$ for inputs $u \in \mathcal{U}$ is denoted by $\bar{f}_{\mathbf{x}}(x, t = r)$ and is exemplary defined for (4.3) as

$$\bar{f}_{\mathbf{x}}(x, t = r) = \sup\big\{f_{\mathbf{x}}(x, t = r)\big|\mathbf{x}(t) \text{ is a solution of (4.3), } u(t) \in \mathcal{U},\ f_{\mathbf{x}}(x, t = 0) = f^0\big\}.$$

□

The enclosing probabilistic hull for a time interval is defined as $\bar{f}_{\mathbf{x}}(x, [0, r]) = \sup\{\bar{f}_{\mathbf{x}}(x, t)|t \in [0, r]\}$ and can be seen as an over-approximative stochastic reachable set. The enclosing probabilistic hull allows the computation of an upper bound $\bar{p}$ of the probability that the state is in an unsafe set $\mathcal{X}^{\text{unsafe}}$ for the time interval $[\underline{t}, \overline{t}]$:

$$\bar{p}([\underline{t}, \overline{t}]) := \int_{\mathcal{X}^{\text{unsafe}}} \bar{f}_{\mathbf{x}}(x, [\underline{t}, \overline{t}])\, dx.$$

The solution of the Ornstein-Uhlenbeck process is recalled in the following subsection as its solution is not as well known as the one for deterministic linear systems.

4.2.1. Solution of the Ornstein-Uhlenbeck Process

In order to derive the solution of the Ornstein-Uhlenbeck process, the Gaussian white noise $\boldsymbol{\xi}$ in (4.3) is written as the derivative of the Wiener process $\mathbf{W}$: $\boldsymbol{\xi} = \frac{d\mathbf{W}}{dt}$, $d\mathbf{W} = \mathbf{N}(0, I \cdot dt)$ where I is the identity matrix and $\mathbf{N}(\mu, \Sigma)$ denotes a random vector of Gaussian or so-called normal distribution with expected value μ and covariance matrix Σ. In analogy to deterministic linear systems, the solution of (4.3) is

$$\mathbf{x}(t) = \underbrace{e^{At}\mathbf{x}(0) + \int_0^t e^{A(t-\tau)}u(\tau)d\tau}_{\mathbf{x}_d(t)} + \underbrace{\int_0^t e^{A(t-\tau)}Cd\mathbf{W}}_{\mathbf{x}_s(t)}. \tag{4.4}$$

Note that in this case the input trajectory $u(t)$ is given and not uncertain in a set $\mathcal{U}$. Due to the superposition principle of linear systems, the above solution is computed separately for

$\mathbf{x}_d(t)$ and $\mathbf{x}_s(t)$. Assuming that the initial state has a Gaussian distribution, the solution $\mathbf{x}_d(t)$ is also Gaussian as the linear map $e^{At}\mathbf{x}(0)$ preserves the Gaussian distribution and $u(t)$ is deterministic. Thus, the solution of $\mathbf{x}_d(t)$ is $\mathbf{x}_d(t) = \mathbf{N}(\mu_d(t), \Sigma_d(t))$ with

$$\begin{aligned}\mu_d(t) &= e^{At}\mu(0) + \int_0^t e^{A(t-\tau)}u(\tau)d\tau \\ \Sigma_d(t) &= e^{At}\Sigma(0)e^{At^T}\end{aligned}$$

using

$$\alpha + D\mathbf{N}(\mu, \Sigma) = \mathbf{N}(\alpha + D\mu, D\Sigma D^T), \quad \alpha \in \mathbb{R},\, D \in \mathbb{R}^{n\times n} \tag{4.5}$$

for the solution of $\Sigma_d(t)$. The random variable $\mathbf{x}_s(t)$ also has a Gaussian distribution which follows directly from $\mathbf{x}_s(t) = \int_0^t e^{A(t-\tau)}Cd\mathbf{W}$ ($d\mathbf{W} = \mathbf{N}(0, I \cdot dt)$), the multiplication rule in (4.5), the addition rule

$$\mathbf{N}(\mu_1, \Sigma_1) + \mathbf{N}(\mu_2, \Sigma_2) = \mathbf{N}(\mu_1 + \mu_2, \Sigma_1 + \Sigma_2) \tag{4.6}$$

and the fact that the integral of $\mathbf{x}_s(t)$ can be written as an infinite sum using the Riemann integral. Thus, the probability distribution of $\mathbf{x}_s(t)$ can be fully characterized by its mean value and covariance:

$$\begin{aligned}\mathbf{x}_s(t) &= \mathbf{N}(0, \Sigma_s(t)) \\ \Sigma_s(t) &= E(\mathbf{x}_s(t)\mathbf{x}_s(t)^T) - \underbrace{E(\mathbf{x}_s(t))E(\mathbf{x}_s(t))^T}_{=0},\end{aligned}$$

where $\Sigma_s(t)$ can be obtained as shown in [67, p. 87] by

$$\Sigma_s(t) = \int_0^t e^{A(t-\tau)}CC^Te^{A^T(t-\tau)}d\tau.$$

The integral can be explicitly evaluated if $AA^T = A^TA$; see [67, p. 109]. On that account, the system equation (4.3) is diagonalized by defining $\mathbf{x}^* = Q^{-1}\mathbf{x}$, where Q is the matrix of eigenvectors of A such that $A^* = Q^{-1}AQ = \mathtt{diag}(\lambda)$, $C^* = Q^{-1}C$, and λ is the vector of eigenvalues. Hence, the solution of $\Sigma_s^*(t)$ [67, p. 109] is:

$$[\Sigma_s^*(t)]_{ij} = \frac{(C^*C^{*T})_{ij}}{\lambda_i + \lambda_j}(1 - e^{(-\lambda_i - \lambda_j)t}), \quad \Sigma_s(t) = Q\Sigma_s^*(t)Q^T. \tag{4.7}$$

The result of $\mathbf{x}(t)$ is the basis for the representation of enclosing probabilistic hulls of the Ornstein-Uhlenbeck process.

4.2.2. Representation of Enclosing Probabilistic Hulls

The Gaussian distribution of $\mathbf{x}(t)$ for a given input trajectory $u(t)$ motivates the use of multivariate Gaussian distributions with uncertain mean as a representation for enclosing probabilistic hulls. The uncertain mean originates from the set of possible inputs $\mathcal{U}$. The representation of enclosing probabilistic hulls is introduced step-by-step. First, the representation of sets with zonotopes is recalled. Next, it is shown that random variables with

Gaussian distribution can be represented by probabilistic zonotopes. Finally, probabilistic zonotopes with uncertain mean are defined which serve as the representation for enclosing probabilistic hulls.

Zonotopes $\mathcal{Z}$ have been defined in Def. 2.3 and are recalled for better readability:

$$\mathcal{Z} = \Big\{ x \in \mathbb{R}^n \big| x = c + \sum_{i=1}^{e} \beta_i \cdot g^{(i)}, \quad -1 \leq \beta_i \leq 1 \Big\}$$

where $c, g^{(1)}, \ldots, g^{e} \in \mathbb{R}^n$, c is referred to as the *center* and $g^{(i)}$ are referred to as the *generators* of $\mathcal{Z}$. Zonotopes are centrally symmetric; the order of a zonotope is $\hat{\varrho} = \frac{e}{n}$, and zonotopes are denoted by $\mathcal{Z} = (c, g^{(1)}, \ldots, g^{(e)})$. A zonotope can also be seen as the Minkowski sum of a finite set of line segments $\hat{l}_i = [-1, 1] \cdot g^{(i)}$; see Fig. 2.2.

By replacing the intervals $\beta_i \in [-1, 1]$ with pairwise independent Gaussian distributed random variables $\mathbf{N}_i(0, 1)$, one can define the following probabilistic zonotope with certain mean:

Definition 4.4 (G-Zonotope): A Gaussian zonotope (G-zonotope) with certain mean is defined as a random variable $\mathbf{Z}$:

$$\mathbf{Z} = c + \sum_{i=1}^{o} \mathbf{N}_i(0, 1) \cdot \mathscr{g}^{(i)}$$

where $\mathscr{g}^{(1)}, \ldots, \mathscr{g}^{(o)}$ are the generators which are represented by a different font in order to distinguish them from generators of regular zonotopes. G-zonotopes are denoted by $\mathbf{Z} = (c, \mathscr{g}^{(1)}, \ldots, \mathscr{g}^{(o)})$. □

For further derivations, it is advantageous to show that G-zonotopes with certain mean have a multivariate Gaussian probability density function. This result also shows that the restriction to zero mean Gaussian distributions with variance 1 in Def. 4.4 is no loss of generality.

Proposition 4.2 (Gaussian Distribution of G-Zonotopes): The probability density function of a G-zonotope with certain mean and of order greater or equal 1 ($\varrho \geq n$) can be formulated as a multivariate Gaussian distribution with

$$f_{\mathbf{Z}}(x) = (2\pi)^{-\frac{\varrho}{2}} \det(\Sigma)^{-\frac{1}{2}} \exp(-0.5(x - c)^T \Sigma^{-1} (x - c)),$$
$$\Sigma = \mathcal{G}\mathcal{G}^T$$

where Σ is the covariance matrix, c is the center and $\mathcal{G} = \begin{bmatrix} \mathscr{g}^{(1)} & \ldots & \mathscr{g}^{(o)} \end{bmatrix}$ is the matrix of probabilistic generators. □

Proof: Due to the independence of the random variables $\mathbf{N}_i$ of the generators, the joint distribution is computed as the product of the PDFs of each random variable:

$$\begin{aligned} f_{\mathbf{N}}(y_1, \ldots, y_o) &:= \prod_{l=1}^{o} f_{\mathbf{N}_l}(y_l) = \prod_{l=1}^{o} (\sqrt{2\pi})^{-1} \exp(-0.5 {y_l}^2) \\ &\overset{!}{=} (2\pi)^{-\frac{\varrho}{2}} \det(\Sigma)^{-\frac{1}{2}} \exp(-0.5 y^T \Sigma^{-1} y) \end{aligned} \tag{4.8}$$

from which follows that $\Sigma = I$ has to be the identity matrix. Next, the random variables $\mathbf{N}_i$ of the generators are mapped to the random vector $\mathbf{Z}$ of the zonotope: $\mathbf{Z} = c + \mathcal{G} \cdot \mathbf{N}$ where $\mathbf{N} = \begin{bmatrix}\mathbf{N}_1 & \dots & \mathbf{N}_o\end{bmatrix}^T$, such that $\mu^* = c$ and $\Sigma^* = \mathcal{G} \cdot I \cdot \mathcal{G}^T = \mathcal{G} \cdot \mathcal{G}^T$; see (4.5). □

Conversely, one can show that a multivariate Gaussian distribution can always be represented by n generators:

Proposition 4.3 (Probabilistic Generators of a Gaussian Distribution): A possible G-zonotope $\mathbf{Z}$ with n probabilistic generators representing a zero-mean multivariate Gaussian distribution is:

$$\mathbf{Z} = \sum_{i=1}^{n} \mathbf{N}_i(0,1)\, \sqrt{\lambda_i} \hat{q}^{(i)}$$

where λ_i are the eigenvalues and $\hat{q}^{(i)}$ are the eigenvectors of the covariance matrix Σ of the multivariate Gaussian distribution $f_{\mathbf{N}}$. □

Proof: Σ is diagonalized using the eigenvector matrix $\hat{Q}$ and the vector of eigenvalues λ:

$$\Sigma = \hat{Q}\, \mathtt{diag}(\lambda)\, \hat{Q}^T = \hat{Q}\, \mathtt{diag}(\lambda)^{0.5} (\mathtt{diag}(\lambda)^{0.5}\, \hat{Q})^T$$

and $\hat{Q}^T = \hat{Q}^{-1}$ as $\hat{Q}$ is orthogonal. Since the converse of Prop. 4.2 is satisfied, one can see that $\mathcal{G} = \hat{Q}\, \mathtt{diag}(\lambda)^{0.5} = [\sqrt{\lambda_1} \hat{q}^{(1)}, \dots, \sqrt{\lambda_n} \hat{q}^{(n)}]$ is the new generator matrix. □

In order to represent enclosing probabilistic hulls $\bar{f}_{\mathbf{x}}$, the multivariate Gaussian distribution is extended by an uncertain mean:

Definition 4.5 (EH-Zonotope): An enclosing hull zonotope (EH-zonotope) denoted by $\mathscr{Z}$ is defined as a G-zonotope $\mathbf{Z}$ where the center is uncertain and can have any value within a zonotope $\mathcal{Z}$. The combination of the random vector $\mathbf{Z}$ with the set $\mathcal{Z}$ is indicated by the $\boxplus$ operator:

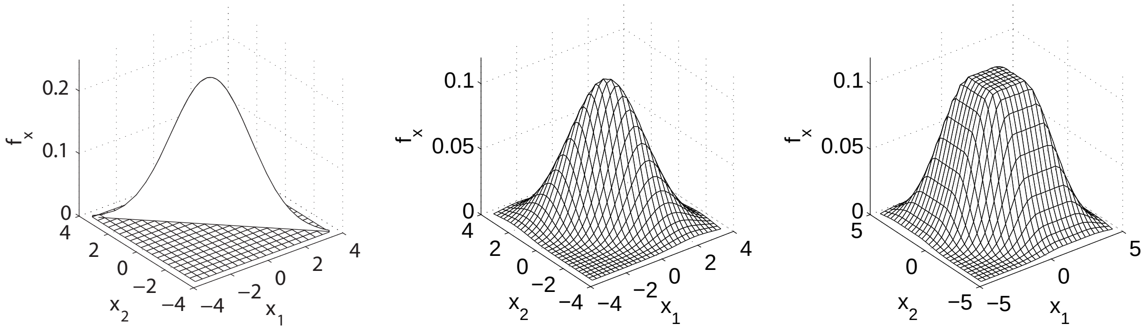
$$\mathscr{Z} := \mathcal{Z} \boxplus \mathbf{Z}, \quad \mathcal{Z} = (c, g^{(1)}, \dots, g^{(e)}), \mathbf{Z} = (0, \mathfrak{g}^{(1)}, \dots, \mathfrak{g}^{(o)}).$$

G-zonotopes with uncertain mean are also denoted by the mixed list of generators of the zonotope and the G-zonotope: $\mathscr{Z} = (c, g^{(1)}, \dots, g^{(e)}, \mathfrak{g}^{(1)} \dots, \mathfrak{g}^{(o)})$. If $o \geq n$, the probabilistic generators can be represented by the covariance matrix Σ according to Prop. 4.2: $\mathscr{Z} = (c, g^{(1)}, \dots, g^{(e)}, \Sigma)$. □

As $\mathscr{Z} = \mathcal{Z} \boxplus \mathbf{Z}$ is not a random vector, there exists no probability density function, but only an enclosing probabilistic hull which is similarly defined as in Def. 4.3:

$$\bar{f}_{\mathscr{Z}} = \sup \left\{ f_{\mathbf{Z}^*} \middle| \mathbf{Z}^* = \mathbf{Z} + a,\, a \in \mathcal{Z} \right\},$$

where $\mathcal{Z}$, $\mathbf{Z}$, $\mathscr{Z}$ are defined as in Def. 4.5. Combinations of sets with random vectors have also been investigated in [20, 21]. From now on, variables such as $\mathscr{Z}$ representing enclosing probabilistic hulls are denoted by *enclosing hull variables*. In Fig. 4.1 it is shown how an enclosing probabilistic hull (EPH) determined by two non-probabilistic and two probabilistic generators is built step-by-step from left to right. Operations on EH-zonotopes are introduced next.



(a) PDF of $(0, \mathscr{g}^{(1)})$.

(b) PDF of $(0, \mathscr{g}^{(1)}, \mathscr{g}^{(2)})$.

(c) EPH of $(0, g^{(1)}, g^{(2)}, \mathscr{g}^{(1)}, \mathscr{g}^{(2)})$.

Fig. 4.1.: Construction of a G-zonotope with uncertain mean.

4.2.3. Operations on Probabilistic Zonotopes

One of the main motivations to represent the uncertain mean of the Gaussian distributions by zonotopes is that they are closed under Minkowski addition and linear transformation, as shown in (2.1). The same rules can be applied to G-zonotopes since the definition of G-zonotopes in Def. 4.4 has the same structure as for zonotopes. After introducing two G-zonotopes $\mathbf{Z}_1 = (0, \mathscr{g}^1, \dots, \mathscr{g}^{(o)})$, $\mathbf{Z}_2 = (0, \mathscr{f}^{(1)}, \dots, \mathscr{f}^{(u)})$, it follows that

$$\begin{aligned} L\,\mathbf{Z}_1 &= (0, L\mathscr{g}^{(1)}, \dots, L\mathscr{g}^{(o)}), \quad L \in \mathbb{R}^{n\times n} \\ \mathbf{Z}_1 + \mathbf{Z}_2 &= (0, \mathscr{g}^{(1)}, \dots, \mathscr{g}^{(o)}, \mathscr{f}^{(1)}, \dots, \mathscr{f}^{(u)}). \end{aligned}$$

Since G-zonotopes can always be represented by n generators, one can reduce the number of generators after the addition of $\mathbf{Z}_1$ and $\mathbf{Z}_2$ as shown in Prop. 4.3. However, a much simpler approach is to directly compute with the covariance matrices. With the equality of the generator and covariance representation $(0, \Sigma) = (0, \mathscr{g}^1, \dots, \mathscr{g}^{(o)})$, the preferred computations for $\mathbf{Z}_1 = (0, \Sigma_1)$ and $\mathbf{Z}_2 = (0, \Sigma_2)$ are according to (4.5) and (4.6):

$$L\,\mathbf{Z}_1 = (0, L\Sigma_1 L^T), \quad \mathbf{Z}_1 + \mathbf{Z}_2 = (0, \Sigma_1 + \Sigma_2).$$

Combining the results of zonotopes with the ones of G-zonotopes yields the addition and multiplication rule for the EH-zonotopes $\mathscr{Z}_1 = (c^{(1)}, g^1, \dots, g^{(e)}, \Sigma_1)$ and $\mathscr{Z}_2 = (c^{(2)}, f^{(1)}, \dots, f^{(u)}, \Sigma_2)$:

$$\begin{aligned} L\,\mathscr{Z}_1 &= (Lc^{(1)}, Lg^{(1)}, \dots, Lg^{(e)}, L\Sigma_1 L^T), \quad L \in \mathbb{R}^{n\times n} \\ \mathscr{Z}_1 + \mathscr{Z}_2 &= (c^{(1)} + c^{(2)}, g^{(1)}, \dots, g^{(e)}, f^{(1)}, \dots, f^{(u)}, \Sigma_1 + \Sigma_2). \end{aligned}$$

A further operator that is important for the computation with G-zonotopes is the confidence set operator. This operator computes a zonotope in which the values of a G-zonotope lie with a certain probability. It is later used to concentrate on the states within a confidence set while neglecting the states outside this set. This is important for some computations since the Gaussian distribution is nonzero everywhere.

Proposition 4.4 (Confidence Set Operator): The confidence set operator $\mathtt{conf}(\mathbf{Z}, m)$ transforms a zero-mean G-zonotope $\mathbf{Z} = (0, \mathscr{g}^1, \dots, \mathscr{g}^{(n)})$ with n probabilistic generators to a zonotope $\mathcal{Z}$ whose generators are obtained by stretching the probabilistic generators by the factor m:

$$\mathtt{conf}(\mathbf{Z}, m) = (0, g^{(1)}, \dots, g^{(n)}), \quad g^{(i)} = m \cdot \mathscr{g}^{(i)},\, m \in \mathbb{R}^+ . \tag{4.9}$$

The choice of n generators is no loss of generality since a G-zonotope can always be represented by n generators; see Prop. 4.3. The obtained set encloses realizations of $\mathbf{Z}$ by a probability of $\mathtt{erf}(\frac{m}{\sqrt{2}})^n$ where $\mathtt{erf}()$ is the error function and n is the dimension of the state space. □

Proof: The probability that a realization of a one-dimensional random variable $\mathbf{x}$ with normalized Gaussian distribution is in an interval $[-m, m]$ is well known to be $P[-m < \mathbf{x} < m] = \mathtt{erf}(\frac{m}{\sqrt{2}})$. The interval $[-m, m]$ is referred to as a confidence interval from now on. Having n instead of one generator, the event that a value lies in the set spanned by all $[-m, m]$ confidence intervals is $\mathtt{erf}(\frac{m}{\sqrt{2}})^n$. □

The definition of a confidence set operator is simply extended for an EH-zonotope by $\mathtt{conf}(\mathscr{Z}, m) := \mathcal{Z} + \mathtt{conf}(\mathbf{Z}, m)$ and $\mathscr{Z} = \mathcal{Z} \boxplus \mathbf{Z}$. A special confidence set is the γ-confidence set ($m = \gamma$). Computations for states outside the γ-confidence set in $\mathcal{Y} = \mathbb{R}^n \backslash \mathtt{conf}(\mathscr{Z}, \gamma)$ are not regarded for enclosing probabilistic hull computations. Instead of computing with probability distributions, only the probability that a state is in $\mathcal{Y}$ is preserved: $\int_{\mathcal{Y}} f_{\mathbf{Z}} dx = 1 - \mathtt{erf}(\frac{\gamma}{\sqrt{2}})^n$. This procedure simplifies the computations of enclosing probabilistic hulls, as shown later. The presented operations are applied to computations of enclosing probabilistic hulls in the following subsection.

4.2.4. Enclosing Probabilistic Hulls of Systems without Input

First, the reachable set of the homogeneous solution is presented as has been done for deterministic dynamic systems in Sec. 3.2. This is possible since the superposition principle can also be applied to stochastic linear systems; see Sec. 4.2.1. The reachable set of the inhomogeneous solution due to the input $u(t) + C\boldsymbol{\xi}(t)$ is added later. The main steps for the computation of enclosing probabilistic hulls are adapted from the reachability analysis of deterministic systems in Chap. 3.

Basic Procedure

The basic steps for computing the enclosing probabilistic hull of the homogeneous solution for a time interval $t \in [0, r]$ are:

1. Computation of the enclosing hull variable $\mathscr{H}(r)$ of the homogeneous solution for $t = r$.
2. Computation of the set $\Delta\mathcal{H}^{\mathcal{R}}$, which encloses all homogeneous solutions starting from the gamma confidence set $\mathtt{conf}(\mathscr{X}^0, \gamma)$ of the initial enclosing hull variable.

3. Selection of the point in time t^* from $[0,r]$ for which the enclosing probabilistic hull has the highest values. Enlargement of the enclosing probabilistic hull $\bar{f}_{\mathscr{H}}(t^*)$ by the addition of the set $\Delta\mathcal{H}^{\mathcal{R}}$ to $\bar{f}_{\mathscr{H}}([0,r])$, which ensures the enclosure of all probability density functions for $t \in [0,r]$.

The mentioned steps are illustrated in Fig. 4.2.

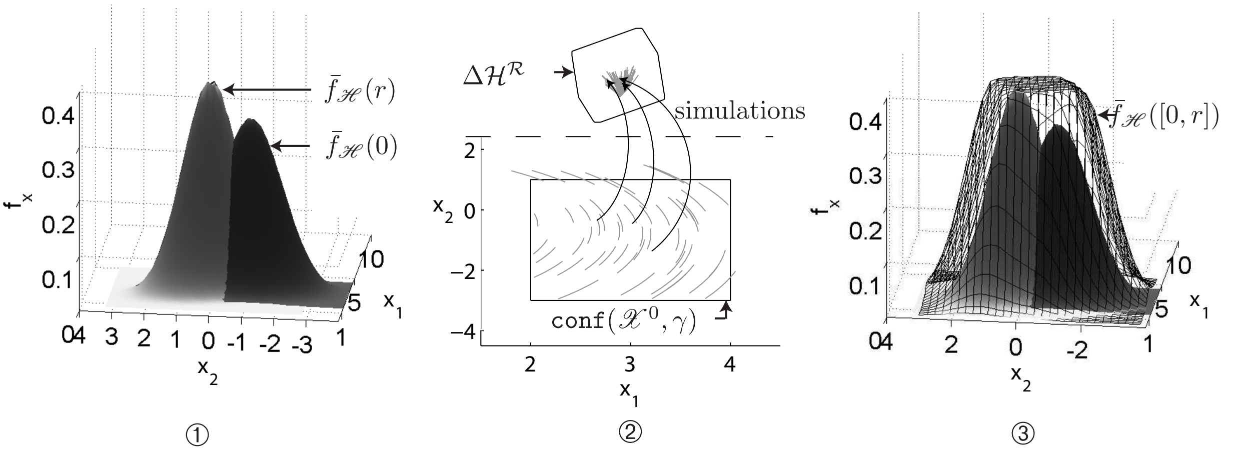


Fig. 4.2.: Computation of the enclosing probabilistic hull for a time interval.

Time Point Solution

The enclosing hull variable of the homogeneous solution $\mathscr{H}(r)$ follows directly from the solution of the Ornstein-Uhlenbeck process (4.4):

$$\mathscr{H}(r) = e^{Ar}\mathscr{X}^0,$$

where $\mathscr{X}^0$ is the initial enclosing hull variable. The computation of the enclosing probabilistic hull for a time interval is a more complicated issue. This is because an enclosing probabilistic hull has to be found for Gaussian probability density functions which are non-zero everywhere. Note that the computation of the enclosing probabilistic hull for the time interval $[0,r]$ differs from the previously published work in [189].

Time Interval Solution

As already mentioned, the enclosing probabilistic hull is computed in two steps. First, the trajectories of the homogeneous solution starting from the gamma confidence set $\mathtt{conf}(\mathscr{X}^0,\gamma)$ are bounded by a set $\Delta\mathcal{H}^{\mathcal{R}}$. This set is not defined as the union of trajectories but the union of trajectories translated such that they start in the origin:

$$\Delta\mathcal{H}^{\mathcal{R}}(t) = \{x^h(t) - x(0) | t \in [0,r], x(0) \in \mathtt{conf}(\mathscr{X}^0,\gamma)\}. \tag{4.10}$$

This is also illustrated in the middle figure of Fig. 4.2. The gamma confidence set has to be applied in order to bound the values $x^h(t)-x(0)$ since $x(0)$ can possibly take values within

$\mathbb{R}^n$ due to the Gaussian distribution of the initial set. According to (3.4), the trajectory for a time interval can be over-approximated by

$$x^h(t) \in x(0) + \frac{t}{r}(e^{Ar}x(0) - x(0)) + \mathcal{F}\, x(0), \quad t \in [0, r].$$

Inserting the over-approximation of $x^h(t)$ in (4.10) results in

$$\begin{aligned} \Delta\mathcal{H}^{\mathcal{R}}([0,r]) \subseteq \quad & \left\{ \frac{t}{r}(e^{Ar}x(0) - x(0)) + \mathcal{F}\, x(0) \middle| t \in [0,r], x(0) \in \mathtt{conf}(\mathscr{X}^0, \gamma) \right\} \\ \subseteq \quad & \left\{ \frac{t}{r}(e^{Ar}x(0) - x(0)) \middle| t \in [0,r], x(0) \in \mathtt{conf}(\mathscr{X}^0, \gamma) \right\} \\ & + \left\{ \mathcal{F}\, x(0) \middle| x(0) \in \mathtt{conf}(\mathscr{X}^0, \gamma) \right\} \\ = \quad & \mathtt{CH}(0, (e^{Ar} - I)\mathtt{conf}(\mathscr{X}^0, \gamma)) + \mathcal{F}\,\mathtt{conf}(\mathscr{X}^0, \gamma). \end{aligned} \tag{4.11}$$

The above formula allows the bounding of the trajectories $\{x^h(t)|t \in [0, r]\}$ by $x(0) + \Delta\mathcal{H}^{\mathcal{R}}([0, r])$ with a probability of $\mathtt{erf}(\frac{\gamma}{\sqrt{2}})^n$ since the enclosure holds only if $x(0) \in \mathtt{conf}(\mathscr{X}^0, \gamma)$; see Prop. 4.4. An alternative computation of $\{x^h(t)|t \in [0, r]\}$ is $x(r) + (-\Delta\mathcal{H}^{\mathcal{R}}([0, r]))$, where the set $\Delta\mathcal{H}^{\mathcal{R}}([0, r])$ is multiplied by -1 and added by Minkowski sum to $x(r)$. This is possible since the homogeneous solution is reversible.

In the second step, it is determined what the highest value of the Gaussian distribution is within the considered time interval. This is done by observing the evolution of the PDF value of a state when following a trajectory of the homogeneous solution. For this problem, the homogeneous solution without uncertain mean $\mathbf{x}^h(t)$ is considered:

Proposition 4.5 (PDF-value Evolution of a State): The PDF value of the random state $\mathbf{x}(t)$ is changed after the linear transformation with e^{At} from its initial value by a factor $e^{-\mathtt{tr}(A)t}$, where $\mathtt{tr}()$ is the trace of a matrix, such that

$$f_{\mathbf{x}^h}(x', t) = e^{-\mathtt{tr}(A)t} f_{\mathbf{x}^h}(x, 0), \quad x' = e^{At}x. \qquad \square$$

Proof: Given is an infinitesimal element around the initial state $x(0)$ which is axis-aligned with the generators of the Gaussian distribution; see Fig. 4.3. The volume of the infinitesimal element, which is a parallelotope, can be computed by the determinant of the vectors of which it is spanned: $V = \mathtt{det}(ds \cdot \mathscr{g}^{(1)}, \ldots, ds \cdot \mathscr{g}^{(n)}) = ds^n \mathtt{det}(\mathcal{G})$, where $\mathcal{G}$ is the matrix of probabilistic generators. The volume of the infinitesimal element after the mapping with e^{At} is $V' = ds^n \mathtt{det}(e^{At}\mathcal{G}) = ds^n \mathtt{det}(e^{At})\mathtt{det}(\mathcal{G})$.

The ratio of the value of the probability density function before and after the mapping is determined by the ratio of the volumes of the infinitesimal elements; see e.g. [153, p. 145]:

$$\frac{f_{\mathbf{x}^h}(x', t)}{f_{\mathbf{x}^h}(x, 0)} = \frac{V}{V'} = \frac{1}{\mathtt{det}(e^{At})}$$

Furthermore, it is well known that $\det(e^{At}) = e^{\mathtt{tr}(At)}$ such that $\mathtt{det}(e^{At})^{-1} = e^{-\mathtt{tr}(A)t}$. $\square$

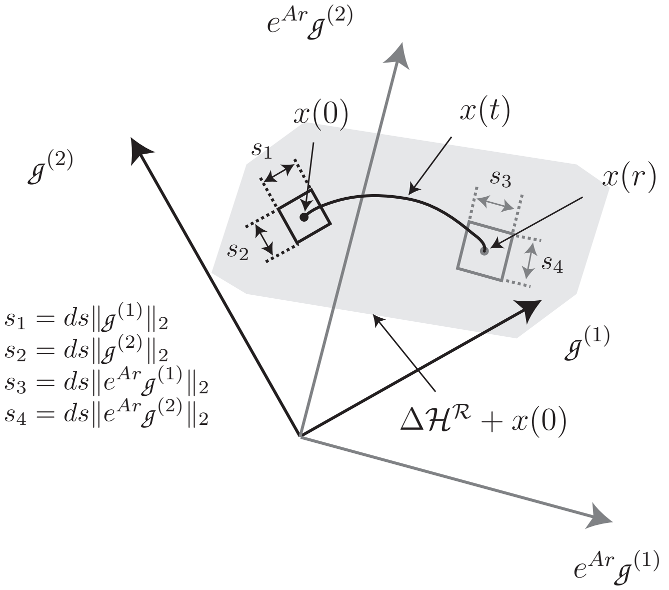

$e^{Ar}\mathscr{g}^{(2)}$
$x(0)$
$x(t)$
$\mathscr{g}^{(2)}$
s_1
s_3
$x(r)$
s_2
s_4
$s_1 = ds\|\mathscr{g}^{(1)}\|_2$
$s_2 = ds\|\mathscr{g}^{(2)}\|_2$
$s_3 = ds\|e^{Ar}\mathscr{g}^{(1)}\|_2$
$s_4 = ds\|e^{Ar}\mathscr{g}^{(2)}\|_2$
$\mathscr{g}^{(1)}$
$\Delta\mathcal{H}^{\mathcal{R}} + x(0)$
$e^{Ar}\mathscr{g}^{(1)}$

Fig. 4.3.: Mapping of an infinitesimal area around a state.

This result shows that the PDF values have a maximum for $t = 0$ or $t = r$ if $t \in [0, r]$ since $e^{-\mathtt{tr}(A)t}$ is monotone. The same result is obtained when adding the uncertain mean so that the enclosing probabilistic hull values are either maximal for $t = 0$ or $t = r$. The idea for the computation of an enclosing probabilistic hull is: Pick the enclosing probabilistic hull of the point in time where its values have a maximum. Then, assume this maximal value for the whole time interval $t \in [0, r]$. This is done by adding the uncertain set $\Delta\mathcal{H}^{\mathcal{R}}([0, r])$ with appropriate sign to $\mathscr{X}^0$ or $\mathscr{H}(r)$:

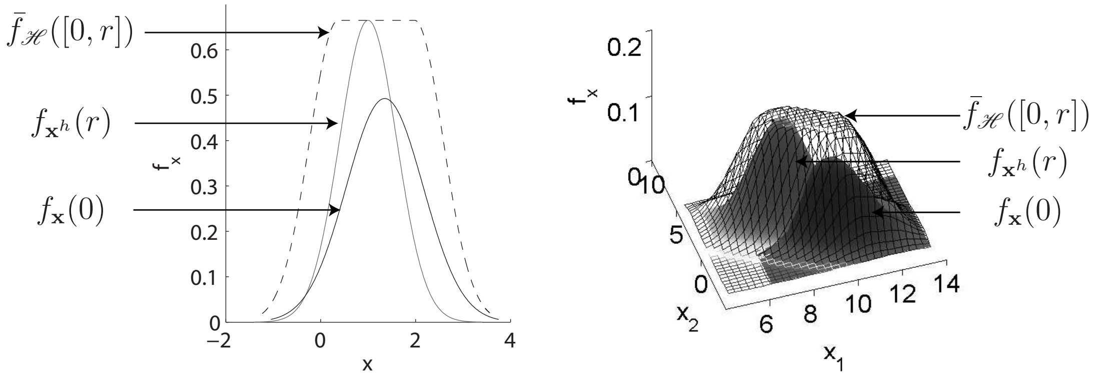
$$\begin{aligned} \mathscr{H}([0,r]) &= \begin{cases} \mathscr{X}^0 + \Delta\mathcal{H}^{\mathcal{R}}, \text{ if } \mathtt{tr}(A) > 0 \\ \mathscr{H}(r) + (-\Delta\mathcal{H}^{\mathcal{R}}), \text{ otherwise.} \end{cases} \\ \Delta\mathcal{H}^{\mathcal{R}} &= \mathtt{CH}(0, (e^{Ar} - I)\mathtt{conf}(\mathscr{X}^0, \gamma)) + \mathcal{F}\mathtt{conf}(\mathscr{X}^0, \gamma). \end{aligned} \tag{4.12}$$

Note that the proposed enclosing hull variable is only valid by a probability of $P(\mathbf{x}(0) \in \mathtt{conf}(\mathscr{X}^0, \gamma)) = \mathtt{erf}(\frac{\gamma}{\sqrt{2}})^n$; see Prop. 4.4. The remaining probability is added to the probability that the system intersects the unsafe set $\mathcal{X}^{\text{unsafe}}$, as shown later. The over-approximation introduced above cannot be quantified analytically. However, for a numerical example, the probability of hitting an unsafe set is later compared to the solution for points in time in Fig. 4.10. Numerical examples of the computation of $\bar{f}_{\mathscr{H}}([0, r])$ can be found in Fig. 4.4 for the one- and two-dimensional case.

For further time intervals $[kr, (k + 1)r]$, $k \in \mathbb{N}^+$, the enclosing hull variable is computed as

$$\mathscr{H}([kr, (k + 1)r]) = e^{Ar}\mathscr{H}([(k - 1)r, kr]).$$

Next, the uncertain input is additionally considered.



(a) One-dimensional example.

(b) Two-dimensional example.

Fig. 4.4.: Enclosure of two Gaussian distributions.

4.2.5. Enclosing Probabilistic Hulls of Systems with Input

Due to the linearity of the Ornstein-Uhlenbeck process, the enclosing hull variable of the input $v = u + C\xi$ can be computed as in Prop. 3.2 to

$$\mathscr{P}([kr,(k+1)r]) = e^{Ar}\mathscr{P}([(k-1)r,kr]) + \mathscr{P}(r) \tag{4.13}$$

which follows directly after substituting $\mathcal{U}$ by $\mathcal{U} \boxplus C\xi$. The solution of $\mathscr{P}(r)$ is computed as

$$\mathscr{P}(r) = \mathcal{P}^{\mathcal{R}}(r) \boxplus \mathbf{x}^p(r),$$

where $\mathcal{P}^{\mathcal{R}}(r)$ is the reachable set due to the uncertain set $\mathcal{U}$ as presented in Sec. 3.2.2 and $\mathbf{x}^p(r) = (0, \Sigma_s(r))$ is the solution due to Gaussian white noise as shown in (4.7).

It remains to compute the enclosing probabilistic hull for the first time interval in order to apply the recursive computation in (4.13). The enclosing hull variable $\mathscr{P}([0,r])$ is over-approximated by its γ-confidence set. Without loss of generality, one can assume that the set of possible inputs $u \in \mathcal{U}$ contains the origin. In the event that the origin is not contained, a new set of uncertain inputs $\tilde{\mathcal{U}} = \mathcal{U} - \tilde{u}$ is defined such that $0 \in \tilde{\mathcal{U}}$ and the effect of the constant input $\tilde{u}$ is computed separately as presented in Sec. 3.2.2. Since besides $0 \in \tilde{\mathcal{U}}$ the inhomogeneous solution due to the Gaussian white noise $\boldsymbol{\xi}(t)$ has zero-mean, the origin is an element of the γ-confidence set $\texttt{conf}(\mathscr{P}(t),\gamma)$, $\forall t$. From this follows that the reachable set $\texttt{conf}(\mathscr{P}(t),\gamma)$ is growing from the origin, which has been shown for deterministic linear systems in Prop. 3.3. Hence, the input solution can be over-approximated by

$$\texttt{conf}(\mathscr{P}([0,r]),\gamma) = \texttt{conf}(\mathscr{P}(r),\gamma) = \mathcal{P}^{\mathcal{R}}(r) \boxplus \texttt{conf}(\mathbf{x}^p(r),\gamma). \tag{4.14}$$

The constant input $\tilde{u}$ is considered similarly as for deterministic linear systems in (3.10):

$$\begin{array}{llll} \Delta x^h(t) + \tilde{x}^p(t) \subseteq & 0 & +\frac{t}{r}\left[e^{Ar}x(0) - x(0)\right] & +\mathcal{F}\,x(0) + \\ & 0 & +\frac{t}{r}\left[\tilde{x}^p(r) - 0\right] & +A^{-1}\mathcal{F}\,\tilde{u}, \quad t \in [0, r] \end{array}$$

From this follows that

$$\begin{aligned} \Delta\mathcal{H}^{\mathcal{R}} :=& \Delta\mathcal{H}^{\mathcal{R}}(t) + \tilde{\mathcal{P}}(t) \\ =& \mathtt{CH}(0, (e^{Ar} - I)\mathcal{X}^0 + \tilde{\mathcal{P}}(r)) + \mathcal{F}\mathcal{X}^0 + A^{-1}\mathcal{F}\,\tilde{u}, \quad \mathcal{X}^0 = \mathtt{conf}(\mathscr{X}^0, \gamma). \end{aligned}$$

Using the obtained results, Alg. 3 for reachable sets of deterministic linear systems is extended to the computation of enclosing hull variables in Alg. 6. In order to compute with the input $\mathcal{U}$ instead of $\tilde{\mathcal{U}}$ as in Alg. 3, the inhomogeneous solution $\tilde{\mathcal{P}}$ is subtracted in the computation of $\Delta\mathcal{H}^{\mathcal{R}}$. Note that for notation reasons, the enclosing hull variables at time intervals are indicated by an index only ($\mathscr{R}_k \hat{=} \mathscr{R}([kr, (k+1)r])$).

Algorithm 6 Compute $\mathscr{R}([0, t_f])$

Input: Initial enclosing hull variable $\mathscr{X}^0$, matrix exponential e^{Ar}, noise matrix C, input set $\mathcal{U}$, correction matrix $\mathcal{F}$, time horizon t_f
Output: $\mathscr{R}([0, t_f])$

```
$\tilde{\mathcal{P}}(r) \rightarrow$ Alg. 3
$\Delta\mathcal{H}^{\mathcal{R}} = \mathtt{CH}(0, (e^{Ar} - I)\mathcal{X}^0 + \tilde{\mathcal{P}}(r)) - \tilde{\mathcal{P}}(r) + \mathcal{F}\mathcal{X}^0 + A^{-1}\mathcal{F}\,\tilde{u}, \quad \mathcal{X}^0 = \mathtt{conf}(\mathscr{X}^0, \gamma)$
$\tilde{\mathscr{R}}_0 = \begin{cases} \mathscr{X}^0 + \Delta\mathcal{H}^{\mathcal{R}}, \text{ if } \mathtt{tr}(A) > 0 \\ \mathscr{H}(r) + (-\Delta\mathcal{H}^{\mathcal{R}}), \text{ otherwise.} \end{cases}$
$\mathcal{P}_0^{\mathcal{R}} \rightarrow$ Alg. 3
$\mathbf{x}_0^{p*} = (0, \Sigma_s(r))$
$\mathbf{x}_0^p = \mathtt{conf}(\mathbf{x}_0^{p*}, \gamma)$
$\mathscr{R}_0 = \tilde{\mathscr{R}}_0 + (\mathcal{P}_0^{\mathcal{R}} \boxplus \mathbf{x}_0^p)$
for $k = 1 \ldots t_f/r - 1$ do
  $\tilde{\mathscr{R}}_k = e^{Ar}\,\tilde{\mathscr{R}}_{k-1}$
  $\mathcal{P}_k^{\mathcal{R}} \rightarrow$ Alg. 3
  $\mathbf{x}_k^p = e^{Ar}\,\mathbf{x}_{k-1}^p + \mathbf{x}_0^{p*}$
  $\mathscr{R}_k = \tilde{\mathscr{R}}_k + (\mathcal{P}_k^{\mathcal{R}} \boxplus \mathbf{x}_k^p)$
end for
$\bar{f}_{\mathscr{R}}([0, t_f]) = \sup(\bar{f}_{\mathscr{R}}([0, r]), \ldots, \bar{f}_{\mathscr{R}}([t_f - r, t_f])$
```

The proposed algorithm differs from the algorithm proposed in [189] in three major points: Alg. 6 does not suffer from the wrapping-effect, the possibility that the input $\mathcal{U}$ does not contain the origin is considered, and the computation of the enclosing hull variable $\tilde{\mathscr{R}}_0$ has been changed. In the next subsection, a method is presented on how to over-approximate the probability that the state is in an unsafe set when the enclosing probabilistic hulls are given.

4.2.6. Probability of Entering an Unsafe Set

The enclosing probabilistic hulls allow the probability to be computed that the system state is in an unsafe set within a certain time interval. The over-approximated probability $\bar{p}([kr,(k+1)r])$ of hitting an unsafe set $\mathcal{X}^{\text{unsafe}}$ is computed in general as

$$\bar{p}([kr,(k+1)r]) = \int_{\mathcal{X}^{\text{unsafe}}} \bar{f}_{\mathscr{R}}(x,[kr,(k+1)r])\,dx.$$

In order to be able to efficiently over-approximate the above integral, the enclosing probabilistic hulls are over-approximated by polytopes; see Fig. 4.6. In addition, the sets of unsafe states $\mathcal{X}^{\text{unsafe}}$ are over-approximated by polytopes, too. Note that the enclosing polytopes of the unsafe sets have dimension n whereas the dimension is $n+1$ for the polytopes over-approximating the enclosing probabilistic hulls[1].

The polytopes over-approximating the enclosing probabilistic hulls $\mathscr{R}$ are constructed by computing the maximum values for a given sequence of confidence sets $Q_1 = \texttt{conf}(\mathscr{R},\gamma)\backslash\texttt{conf}(\mathscr{R},m_1)$, $Q_2 = \texttt{conf}(\mathscr{R},m_1)\backslash\texttt{conf}(\mathscr{R},m_2),\ldots$ and $\gamma > m_1 > m_2 > \ldots > 0$; see Fig. 4.5. The maximum values $h_1^{max}, h_2^{max},\ldots$ of the enclosing probabilistic hulls within the corresponding sets $Q_1, Q_2, \ldots$ are determined as follows:

Proposition 4.6 (Maximum Probability Value within Confidence Sets):
The maximum probability value of an EH-zonotope $\mathscr{Z}$ within a confidence set $Q_i = \texttt{conf}(\mathscr{Z},m_{i-1})\backslash\texttt{conf}(\mathscr{Z},m_i)$ ($\gamma > m_1 > m_2 > \ldots > 0$) is

$$h_i^{max} := \texttt{max}\{\bar{f}_{\mathscr{Z}}(x)|x \in Q_i\} = (2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-0.5(m_i)^2). \qquad \square$$

Proof: The maximum probability value of an EH-zonotope $\mathscr{Z} = \mathcal{Z} \boxplus \mathbf{Z}$ can be reformulated to the problem of finding the maximum probability value of a G-zonotope:

$$\texttt{max}(\{\bar{f}_{\mathscr{Z}}(x)|x \in Q_i\}) = \texttt{max}(\{f_{\mathbf{Z}}(x)|x \in \mathbb{R}^n\backslash\texttt{conf}(\mathbf{Z},m_i)\})$$

The PDF-value of $\mathbf{Z}$ is computed as presented in (4.8):

$$f_{\mathbf{Z}}(x) = (2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-0.5x^T\Sigma^{-1}x) = (2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-0.5y^Ty), \qquad \square$$

where $y = \mathcal{G}x$, $\Sigma = \mathcal{G}\mathcal{G}^T$, and $\mathcal{G}^T = \mathcal{G}^{-1}$. The latter equality holds because the set of generators $\mathcal{G}$ is obtained from Prop. 4.3 without loss of generality. The value of $f_{\mathbf{Z}}(y)$ is maximal under the condition that $x \in \mathbb{R}^n\backslash\texttt{conf}(\mathbf{Z},m_i)$ for $\mathbf{y}_j^* = m_i$ and $\mathbf{y}_i^* = 0$ where $i \neq j$. Thus, $\texttt{max}(\{\bar{f}_{\mathscr{Z}}(x)|x \in Q_i\}) = f_{\mathbf{Z}}(y^*) = (2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-0.5(m_i)^2)$.

The values $h_1^{max}, h_2^{max},\ldots$ are also illustrated for an enclosing probabilistic hull of a two-dimensional state space in Fig. 4.6.

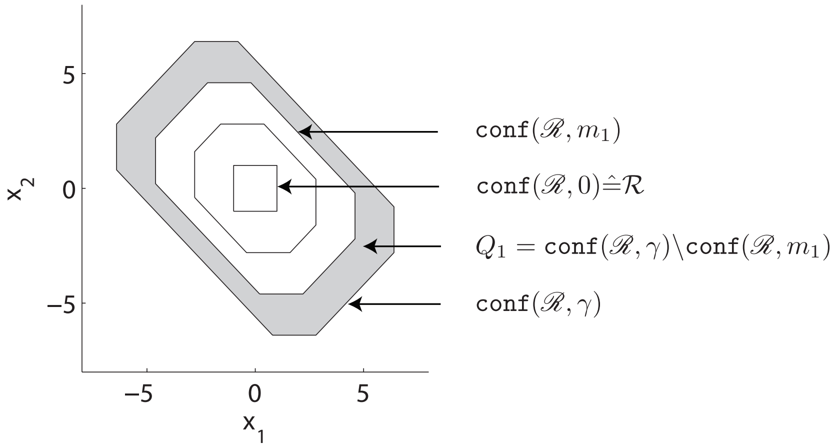

5
0
−5
−5
0
5
x_2
x_1
conf($\mathscr{R}$, m_1)
conf($\mathscr{R}$, 0)≙$\mathcal{R}$
Q_1 = conf($\mathscr{R}$, γ)\conf($\mathscr{R}$, m_1)
conf($\mathscr{R}$, γ)

Fig. 4.5.: Confidence sets of $\mathscr{R}$.

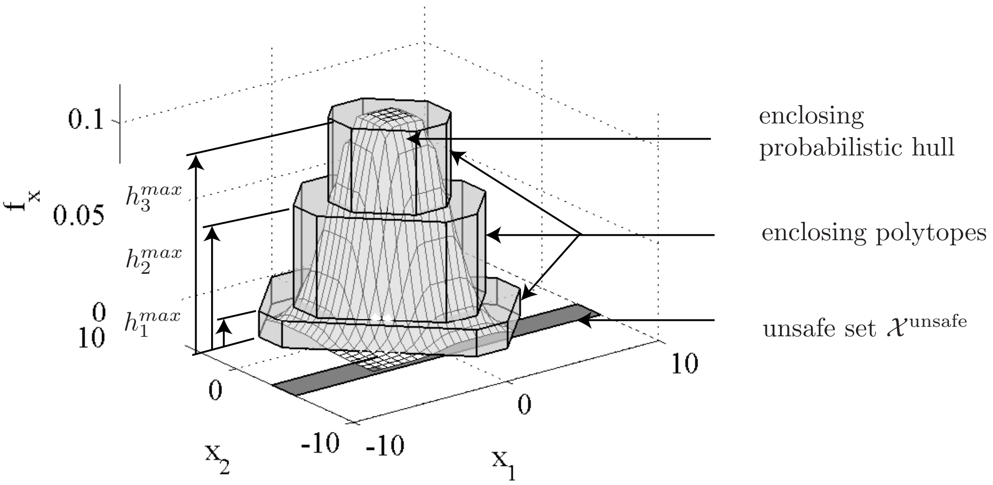


Fig. 4.6.: Over-approximation of an EH-zonotope by piecewise uniform distributions.

The enclosure by polytopes allows an over-approximated probability for being in the unsafe set $\mathcal{X}^{\text{unsafe}}$ to be computed.

Proposition 4.7 (Over-approximation of the Probability that x ∈ $\mathcal{X}^{\text{unsafe}}$): The probability $\bar{p}$ that a state x is within an unsafe set $\mathcal{X}^{\text{unsafe}}$ is computed as

$$\bar{p} = 1 - \texttt{erf}(\frac{\gamma}{\sqrt{2}})^{2n} + \sum_i h_i^{max} \cdot \texttt{V}(Q_i \cap \mathcal{X}^{\text{unsafe}}),$$

where $\texttt{V}()$ is an operator returning the volume of a geometric object and Q_i are the confidence sets introduced in Prop. 4.6. □

[1] One additional dimension is needed for the probability values of the enclosing probabilistic hulls.

Proof: The expression $1-\mathtt{erf}(\frac{\gamma}{\sqrt{2}})^{2n}$ accounts for the probability that an unsafe set may be reached due to neglecting the probability values outside the γ-confidence set; see Prop. 4.4. In contrast to Prop. 4.4, the value $\mathtt{erf}(\ldots)^n$ is squared because the γ-confidence set has been used for the homogeneous and the inhomogeneous solution in Alg. 6.

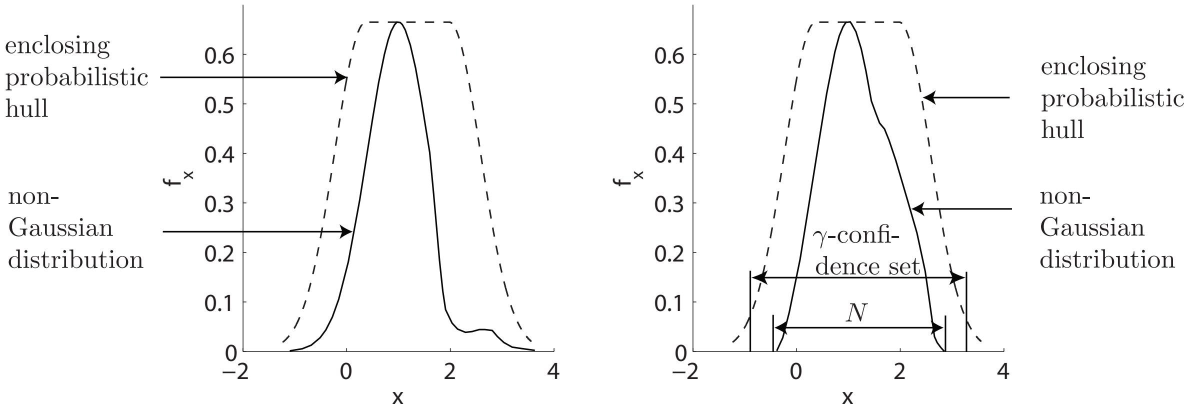
The other term accounts for the intersection of the enclosing polytopes with the unsafe set as shown in Fig. 4.6. □

Because of the over-approximated computations, it is possible that $\bar{p} > 1$ which is then limited to $\bar{p} = 1$ since a probability cannot be greater than 1. The probabilities for specific time intervals $\bar{p}([kr,(k+1)r])$ are obtained according to Prop. 4.7. It is obvious that the average probability within a larger time interval $t \in [kr,(k+h)r]$ ($h \in \mathbb{N}^+$) is $\frac{1}{h-k}\sum_{k=k}^{k+h}\bar{p}([kr,(k+1)r])$. Before the probabilities $\bar{p}([kr,(k+1)r])$ are computed for a numerical example, the following extension is presented.

4.2.7. Extension to non-Gaussian Initial and Input Distribution

The presented approach can be extended to non-Gaussian initial states and non-Gaussian white noise. This is possible by enclosing hull variables $\mathscr{X}^0$, $\mathscr{P}(r)$ which enclose the non-Gaussian probability density functions of the initial state and the input solution; see Fig. 4.7(a).

Additionally, if the initial state and input distribution are non-zero in a bounded set, one can possibly guarantee the safety of the system. For this, the set of non-zero values N has to lie within the γ-confidence sets of the EH-zonotopes representing the initial state and the input solution, which is exemplarily illustrated in Fig. 4.7(b). In this case, the probability that a state is outside the γ-confidence set is 0 for all times. From this follows that $\bar{p}$ in Prop. 4.7 changes to $\bar{p} = \sum_i h_i^{max} \cdot \mathtt{V}(Q_i \cap \mathcal{X}^{\text{unsafe}})$ which is possibly 0.



(a) Distribution with unbounded non-zero values. (b) Distribution with bounded non-zero values.

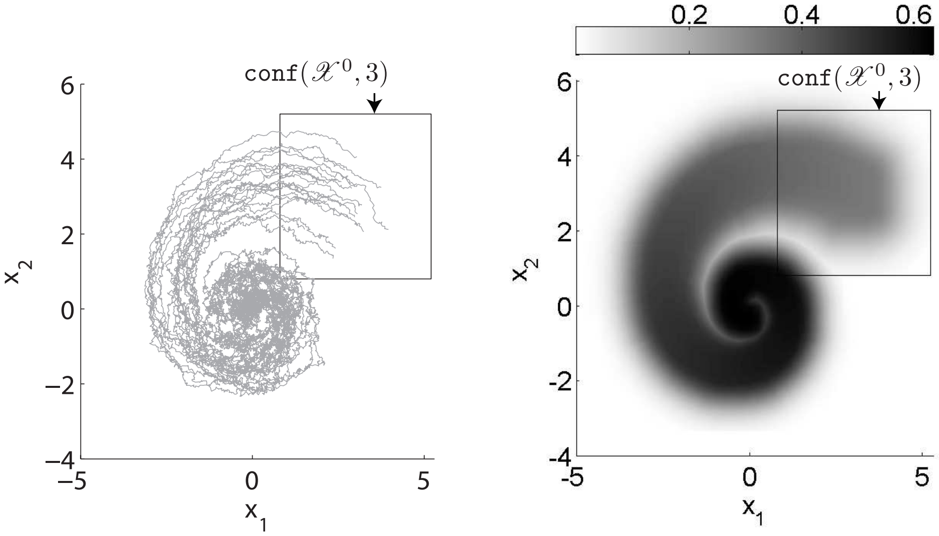
Fig. 4.7.: Non-Gaussian distributions.

4.2.8. Numerical Examples

For the illustration of the presented techniques, the exemplary deterministic linear systems in Sec. 3.2.3 are enhanced with Gaussian white noise. The first example is two-dimensional:

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & -4 \\ 4 & -1 \end{bmatrix} \mathbf{x}(t) + u(t) + \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix} \boldsymbol{\xi}(t), \quad u(t) \in \begin{bmatrix} [-0.1, 0.1] \\ [-0.1, 0.1] \end{bmatrix}.$$

Several simulations of the specified system are illustrated in Fig. 4.8(a). Note that these simulations differ from the ones in [189] since an error in the simulation of linear stochastic systems has been corrected. The corresponding enclosing probabilistic hulls are computed for 250 time steps with a time increment of $r = 0.01$ such that the time horizon is $t_f = 2.5$. The values of the overall enclosing probabilistic hull $\bar{f}_{\mathscr{R}}([0, t_f]) = \sup(\bar{f}_{\mathscr{R}}([0, r]), \dots, \bar{f}_{\mathscr{R}}([t_f - r, t_f])$ are visualized in Fig. 4.8(b), where the color bar on top of the plot relates the probability values to the gray values of the plot.



(a) Simulation results.

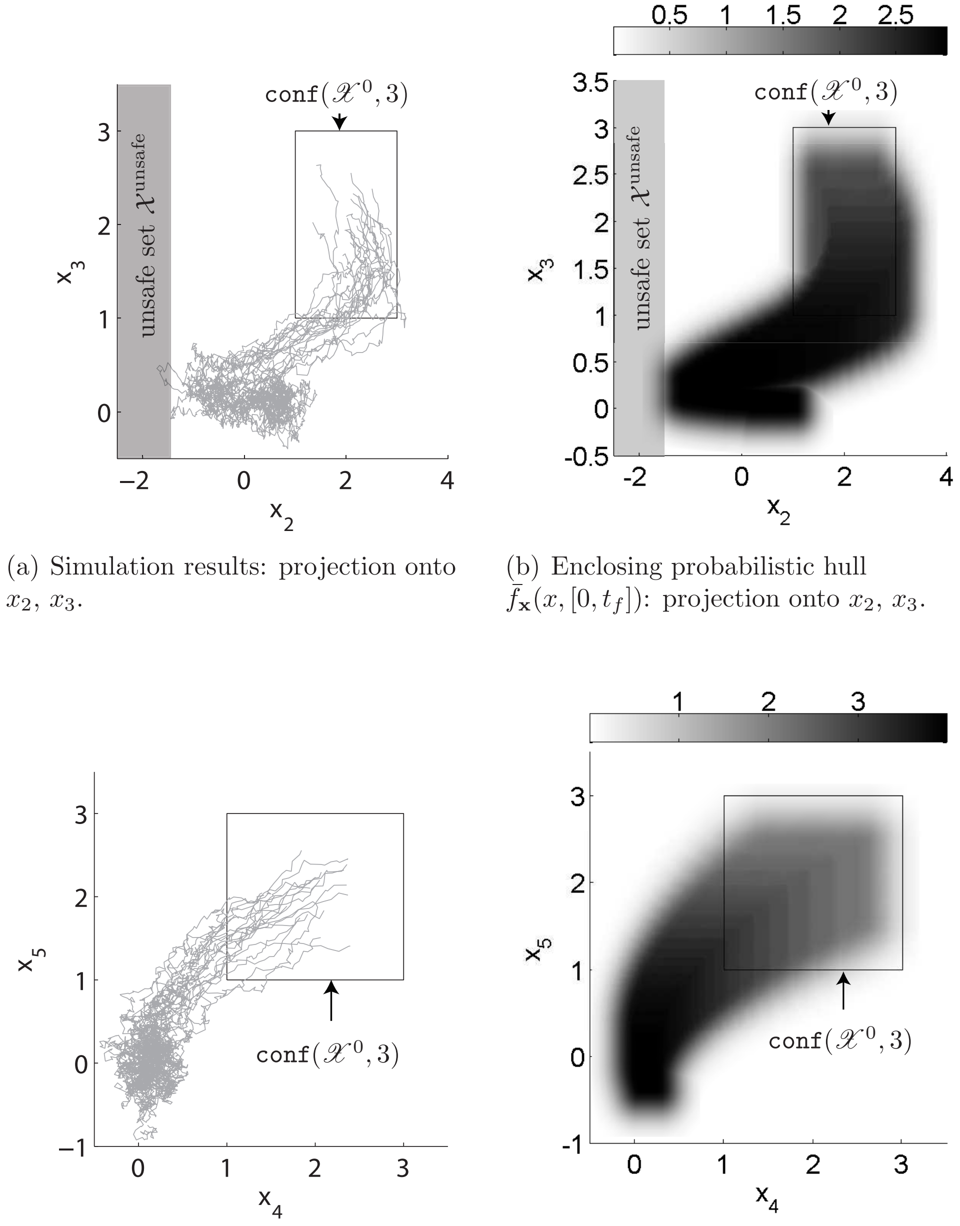
(b) Enclosing probabilistic hull $\bar{f}_{\mathbf{x}}(x, [0, t_f])$.

Fig. 4.8.: Simulation and enclosing probabilistic hulls of the two-dimensional system.

The second example is of dimension 5, and with a system matrix as specified in (3.11):

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & -4 & 0 & 0 & 0 \\ 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & -3 & 1 & 0 \\ 0 & 0 & -1 & -3 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{bmatrix} \mathbf{x}(t) + u(t) + 0.4 \cdot I \cdot \boldsymbol{\xi}(t), \quad u(t) \in \begin{bmatrix} [0.9, 1.1] \\ [-0.25, 0.25] \\ [-0.1, 0.1] \\ [0.25, 0.75] \\ [-0.75, -0.25] \end{bmatrix}.$$

The enclosing probabilistic hulls of two-dimensional projections ($\mathscr{R}' = P\mathscr{R}$, $P \in \mathbb{R}^{2\times n}$) for a sampling time of $r = 0.04$ are presented in Fig. 4.9 together with the unsafe set $x_2 < -1.5$. Note that the input probability distribution can be arbitrary as long as it is enclosed by the Gaussian distribution and the uncertain mean $\mathcal{U}$.



(a) Simulation results: projection onto x_2, x_3.

(b) Enclosing probabilistic hull $\bar{f}_{\mathbf{x}}(x, [0, t_f])$: projection onto x_2, x_3.

(c) Simulation results: projection onto x_4, x_5.

(d) Enclosing probabilistic hull $\bar{f}_{\mathbf{x}}(x, [0, t_f])$: projection onto x_4, x_5.

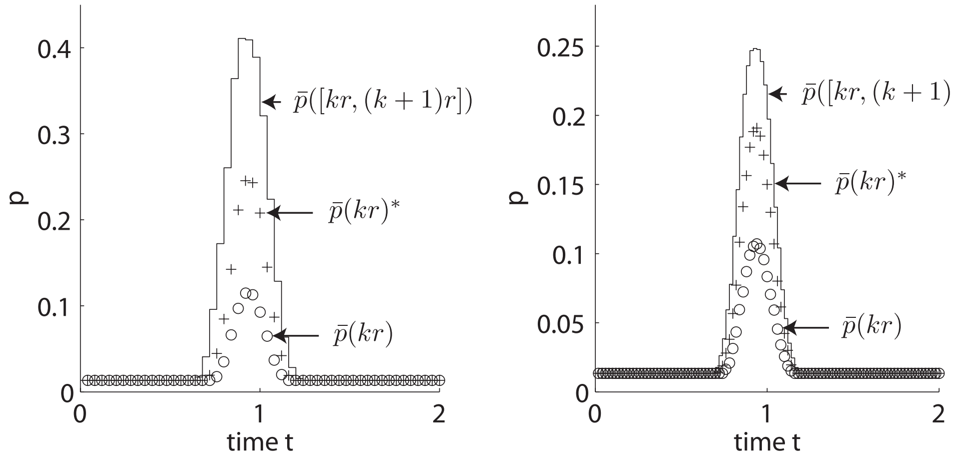
Fig. 4.9.: Enclosing probabilistic hulls of the five-dimensional system.

The over-approximated probability that the state is in an unsafe set is shown in Fig. 4.10 for different time increments and over-approximations:

- $\bar{p}([kr,(k+1)r])$ computes the over-approximated probability of the state being in an unsafe set within time intervals. The over-approximation is caused by the enclosure of the homogeneous solution (see (4.12)) and the inhomogeneous solution (see (4.14)) for time intervals .
- $\bar{p}(kr)^*$ is the time point solution for which the over-approximated time interval solution of the homogeneous solution in (4.14) is not applied while the over-approximation in (4.14) is kept.
- $\bar{p}(kr)$ computes the probability of hitting the set of unsafe states at points in time without applying (4.14) and (4.14).

The different over-approximations reveal the quantity of the over-approximation caused by the enclosure of the homogeneous and the inhomogeneous solution for time intervals.

Additionally, EH-zonotopes $\mathscr{R}([kr,(k+1)r])$ for 500 time intervals and larger systems with randomly generated matrices A and C were computed. The computation times are presented in Tab. 4.1 and were obtained by a desktop computer with an AMD Athlon64 3700+ processor (single core) in Matlab. Note that enclosing probabilistic hulls can be computed for systems of dimension 100 in reasonable time.



(a) Time increment $r = 0.04$.

(b) Time increment $r = 0.02$.

Fig. 4.10.: Over-approximated probability that the state enters the unsafe set: Time interval and time point solution.

Tab. 4.1.: Computational times.

Dimension n	5	10	20	50	100
CPU-time [s]	0.34	0.39	0.55	2.89	13.5

Next, Markov chain abstraction is presented as an alternative method for stochastic reachability analysis.

4.3. Markov Chain Abstraction

The main idea of the Markov chain abstraction is to compute the probability distribution by a Markov chain instead of making use of the original system dynamics. The Markov chain has to be generated such that it approximates the behavior of the original system with appropriate accuracy. The abstraction can be applied to both, continuous and hybrid systems. Since Markov chains are stochastic systems with a discrete state space, the continuous state space $\mathbb{R}^n$ of the original system has to be discretized for the abstraction. The discretization is performed by partitioning the continuous space into cells. This implies that the number of states of the Markov chain grows exponentially with the dimension of the continuous state space. Thus, the presented abstraction is only applicable to low dimensional systems of typically up to $3-5$ continuous state variables.

Although the abstraction to Markov chains is possible for hybrid systems, the remainder of this section focuses on purely continuous systems for simplicity. All steps undertaken for the abstraction remain the same for hybrid systems – the only difference is that the combined continuous and discrete state space has to be mapped to the discrete state space of the Markov chain instead of only mapping the continuous state space. For example, for a hybrid system with l discrete states (locations), each of the invariant sets (continuous state space region of a discrete state) is discretized into m discrete states, which results in $l \cdot m$ discrete states of the Markov chain. Where the invariants of the hybrid system do not intersect, there exists a unique mapping from the continuous to the discrete state of the hybrid system. In this case it is sufficient to map the continuous state space of the hybrid system to the discrete one of the Markov chain.

The nonlinear continuous system considered for the abstraction to Markov chains is similar to the one in (3.27) for reachability analysis. The initial state $x(0)$ can take values from a set $\mathcal{X}^0 \subset \mathbb{R}^n$, the dynamic depends on a set of parameters $\mathcal{P} \subseteq \mathcal{I}^p$ ($\mathcal{I}$ is the set of possible intervals), and the input u can take values from a bounded set $\mathcal{U} \subset \mathbb{R}^m$. In addition to the nonlinear system in (3.27), this definition has an additional input $\hat{u}(t)$, which is typically a noise term and cannot be changed after the system has been abstracted to a Markov chain. The other input u remains an input after the abstraction and can be changed in between the updates of the Markov chain. The evolution of the state x is given by the following differential equation:

$$\begin{gathered}\dot{x} = f(x(t), u(t), \hat{u}(t), \rho), \qquad (4.15)\\ x(0) \in \mathcal{X}^0 \subset \mathbb{R}^n, \quad \rho \in \mathcal{P} \subseteq \mathcal{I}^p, \quad u(t) \in \mathcal{U} \subset \mathbb{R}^m, \quad \hat{u}(t) \in \mathbb{R}^m.\end{gathered}$$

It is remarked that some vectors such as the state x or the input $\hat{u}$ might be random vectors. However, since these vectors might also be deterministic (depending on the considered system), they are not written with bold letters as is done for random vectors. Next, the definition of Markov chains is presented, which is adapted from [36]:

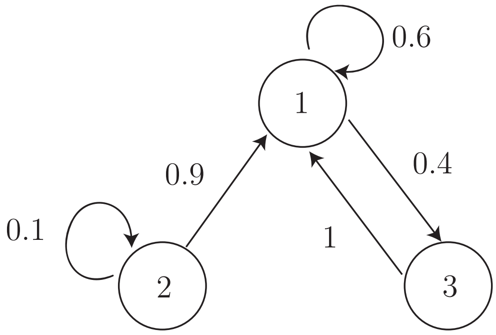
Definition 4.6 (Discrete Time Markov Chain): A discrete time Markov chain $MC = (Y, \hat{p}^0, \Phi)$ consists of

- The countable set of locations $Y \subset \mathbb{N}^+$.
- The initial probability $\hat{p}_i^0 = P(\mathbf{z}(0) = i)$, with random state $\mathbf{z} : \Omega \to Y$, where Ω is

the set of elementary events and $P()$ is an operator determining the probability of an event.

- the transition matrix $\Phi_{ij} = P(\mathbf{z}(k+1) = i | \mathbf{z}(k) = j)$ so that $\hat{p}(k+1) = \Phi\hat{p}(k)$. □

Clearly, the Markov chain fulfills the Markov property, i.e. the probability distribution of the future time step $\hat{p}(k+1)$ depends only on the probability distribution of the current time step $\hat{p}(k)$. If a process does not fulfill this property, one can always augment the discrete state space by states of previous time steps, allowing the construction of a Markov chain with the new state $\mathbf{z}^*(k)^T = \left[\mathbf{z}(k)^T, \mathbf{z}(k-1)^T, \mathbf{z}(k-2)^T, \ldots\right]$. An example of a Markov chain is visualized in Fig. 4.11 by a graph whose nodes represent the states $1, 2, 3$ and whose labeled arrows represent the transition probabilities Φ_{ij} from state j to i.



$$\Phi = \begin{bmatrix} 0.6 & 0.9 & 1 \\ 0 & 0.1 & 0 \\ 0.4 & 0 & 0 \end{bmatrix}$$

Fig. 4.11.: Exemplary Markov chain with 3 states.

The relation of the discrete time step k and the continuous time is established by introducing the time increment $\tau \in \mathbb{R}^+$ after which the Markov chain is updated according to the transition matrix Φ. Thus, the continuous time at time step k is $t_k = k \cdot \tau$.

The generation of a Markov chain from a continuous dynamics in (4.15) can be divided into two steps. First, the state space of the original continuous system is discretized into cells representing the discrete states. Second, the transition probabilities from one cell to another cell, which are stored in the transition matrix of the Markov chain, have to be determined. These steps are detailed below.

4.3.1. Discretization of the State and Input Space

In order to obtain a finite number of discrete states, a subset of the continuous state and input space is discretized. Note that only the input space of the input $u(t)$ is discretized while the input $\hat{u}(t)$ is unaffected by discretization. From now on, the discretized state space is denoted by $\mathcal{X} \subset \mathbb{R}^n$ and the discretized input space is denoted by $\mathcal{U} \subset \mathbb{R}^m$. In this work, each coordinate of $\mathcal{X}$ and $\mathcal{U}$ is partitioned into equidistant intervals such that each cell is a hyperrectangle of equal size, as shown in Fig. 4.12 for the two-dimensional case. Thus, each cell $\mathcal{X}_i$ with cell index i can be described by an n-dimensional interval: $\mathcal{X}_i =]\underline{x}_i, \overline{x}_i]$, $\underline{x}_i, \overline{x}_i \in \mathbb{R}^n$. Analogously, the input cell $\mathcal{U}^\alpha$ with cell index α is an m-dimensional interval:

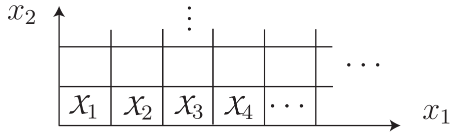


Fig. 4.12.: Discretization of the state space.

$\mathcal{U}^\alpha =]\underline{u}^\alpha, \overline{u}^\alpha]$, $\underline{u}^\alpha, \overline{u}^\alpha \in \mathbb{R}^m$. In order to easily distinguish between state indices and input indices, state indices are subscripted Latin letters and input indices are superscripted Greek letters. The set without indices refer to the union of the cells such that $\mathcal{X} = \bigcup_{i=1}^d \mathcal{X}_i$ and $\mathcal{U} = \bigcup_{\alpha=1}^c \mathcal{U}^\alpha$, where d and c are the numbers of discrete states and inputs, respectively. Other works use a more flexible discretization; see e.g. [112, 150, 157].

The state and input indices have values from 1 onwards. The region outside the discretized state space $\mathbb{R}^n \backslash \mathcal{X}$ is referred to as the outside cell. This outside cell has the index 0 and is of importance when computing the transition probabilities of the Markov chain.

After the discretization, the event that the continuous state x is in a cell $\mathcal{X}_i$ $(x \in \mathcal{X}_i)$ is equivalent to the event that the value of the discrete state is $\mathbf{z} = i$. Besides the discrete state $\mathbf{z}$, the discrete input $\mathbf{y}$ is defined and $u \in \mathcal{U}^\alpha$ is represented by $\mathbf{y} = \alpha$. The parameter space is not discretized in this work, but it could be done analogously to the state and input space.

4.3.2. Transition Probabilities

Based on the discretization of the state space, the transition probabilities $\Phi_{ij} = P(\mathbf{z}(k+1) = i|\mathbf{z}(k) = j)$ of the Markov chain have to be computed from the continuous dynamics of the original system in (4.15). Where the input is also discretized, the transition probability has to be obtained for a given input $\mathbf{y}(k) = \alpha$ so that $\Phi_{ij}^\alpha = P(\mathbf{z}(k+1) = i, \mathbf{y}(k) = \alpha|\mathbf{z}(k) = j, \mathbf{y}(k) = \alpha)$ is computed. In this thesis, the transition probabilities are computed by reachability analysis and simulations, where the latter is presented first.

Transition Probabilities Using Monte Carlo Simulation

The method of computing transition probabilities from a cell $\mathcal{X}_j$ using simulations is described. First, a finite set of initial states is randomly generated from the cell $\mathcal{X}_j$. For each initial state, an input value $u([0,\tau])$ is randomly generated from the input cell $\mathcal{U}^\alpha$ which is constant within $[0,\tau]$. In addition, an input trajectory $\hat{u}(t)$ and a parameter vector ρ are computed according to their probability distribution and the specified input dynamics. Each initial state is then simulated under the inputs $u([0,\tau])$, $\hat{u}(t)$ and parameter vector ρ according to the system dynamics (4.15) for a time interval $[0,\tau]$. This procedure is visualized in Fig. 4.13(a). Simulations generated from random sampling are also called Monte Carlo simulations. Alternatively, the initial states as well as the input values can be generated by a predefined grid on the initial cell $\mathcal{X}_j$ and the input cell $\mathcal{U}^\alpha$.

In contrast to the probability distributions of the input $\hat{u}(t)$ and the parameter vector ρ, the probability distribution within all state and input cells is strictly uniform. This assumption allows the reconstruction of a piecewise constant probability distribution of the continuous state x from the discrete distribution of the Markov chain.

The number of simulations starting from the initial cell $\mathcal{X}_j$ under input $u \in \mathcal{U}^\alpha$ is denoted by $n_j^{\text{sim},\alpha}$ and the number of those simulations located in cell $\mathcal{X}_i$ at time τ is denoted by $n_{i,j}^{\text{sim},\alpha}$. Transition probabilities are computed by the relative number of trajectories reaching a goal cell with index i when starting in the initial cell with index j:

$$\Phi_{ij}^\alpha(\tau) = \frac{n_{i,j}^{\text{sim},\alpha}}{n_j^{\text{sim},\alpha}}. \tag{4.16}$$

The probability that another state is reached within a time interval $[0, \tau]$ is approximated by computing the transition probabilities for a finite set of equidistant intermediate points in time $\tilde{t}_0, \tilde{t}_1, \ldots, \tilde{t}_{\tilde{n}} \in [0, \tau]$, where $\tilde{n}$ is the number of intermediate points in time. The transition probability for the time interval is obtained by the arithmetic mean of the intermediate transition probabilities.

$$\Phi_{ij}^\alpha([0, \tau]) = \frac{1}{\tilde{n}} \sum_{k=1}^{\tilde{n}} \Phi_{ij}^\alpha(\tilde{t}_k).$$

One disadvantage when computing the transition probabilities from Monte Carlo simulation is that the resulting model is not complete, i.e. there might exist a non-zero probability from cell j to cell i when the abstraction is exact in the sense of the transition probabilities. However, due to the finite number of Monte Carlo simulations, a feasible simulation run from cell j to i might be missed. Thus, the transition probability is computed to 0 even though it is non-zero.

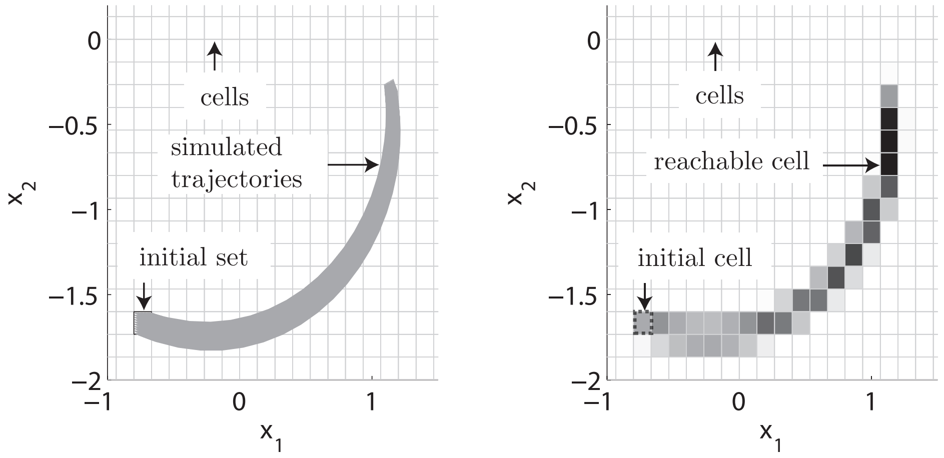
The computation of the transition probability from a cell j is exemplarily shown in Fig. 4.13. Generated trajectories are shown in Fig. 4.13(a) and the corresponding stochastic reachable set when starting in cell j with probability 1 is illustrated in Fig. 4.13(b). A transition to a cell is the more likely, the darker the color of the cell is.

This approach is applicable to all continuous and hybrid systems that can be numerically simulated, which is the case for continuous dynamics fulfilling the Lipschitz continuity. For hybrid systems, it is possible that their behavior is non-deterministic, e.g. when transitions are enabled but not enforced. In such a case one has to properly assign probabilities to possible executions as is done for stochastic hybrid systems, such that the numerical simulation is clearly defined in a stochastic sense (see e.g. [33]).

Finally, it is again remarked that the abstraction can be applied to deterministic and stochastic systems. In either case, the result is a discrete stochastic system. The stochasticity is introduced for originally deterministic systems because of the uncertain location of the continuous state within a cell. Next, the abstraction via reachability analysis is presented.

Transition Probabilities Using Reachability Analysis

Another option for computing transition probabilities is the use of reachability analysis. In order to apply reachability analysis as presented in Chap. 3, the inputs and parameters in (4.15) have to be bounded ($\rho \in \mathcal{P} \subset \mathcal{I}^p$, $(u(t) + \hat{u}(t)) \in \mathcal{U} \subset \mathbb{R}^m$); see (3.27). This is



(a) Simulated trajectories for a time interval.

(b) Reachable cells for a time interval.

Fig. 4.13.: Simulated trajectories of the original system and the corresponding stochastic reachable set of the abstracting Markov chain.

not required when using Monte Carlo simulation because samples can be generated from unbounded probability distributions.

In order to obtain the transition probabilities from a cell $\mathcal{X}_j$, the reachable set starting from this cell under the set of possible inputs $\mathcal{U}^\alpha$ is computed. The reachable set is denoted by $\mathcal{R}_j^\alpha(\tau)$, where the indices refer to the cells of the input and the initial state. The fraction of the reachable set intersecting with other cells determines the transition probability. After introducing the volume operator $\mathtt{V}()$ returning the volume of a set, the transition probabilities are obtained as

$$\Phi_{ij}^\alpha(\tau) = \frac{\mathtt{V}(\mathcal{R}_j^\alpha(\tau) \cap \mathcal{X}_i)}{\mathtt{V}(\mathcal{R}_j^\alpha(\tau))}. \tag{4.17}$$

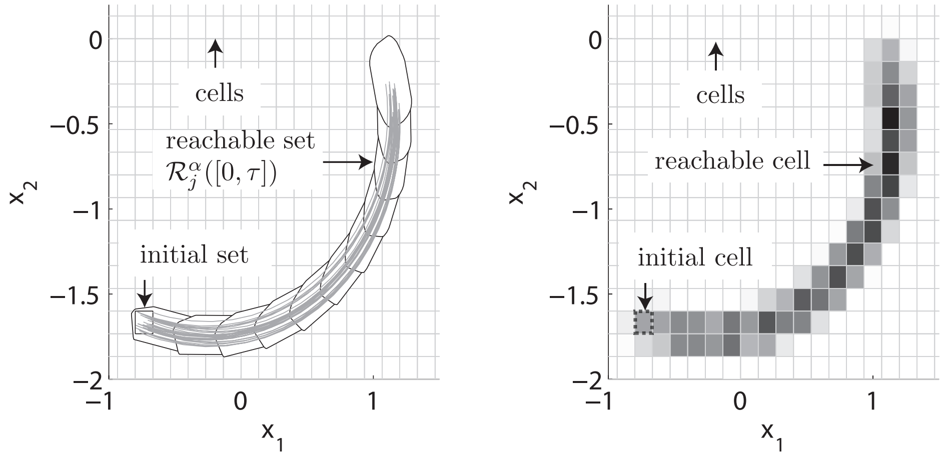
The above formula assumes that the states are uniformly distributed within $\mathcal{R}_j^\alpha(\tau)$. Similarly to the case when computing transition probabilities from simulations, the transition probabilities of the time interval $[0, \tau]$ are computed as the arithmetic mean from transition probabilities of intermediate steps. In contrast to the simulative approach, the intermediate reachable sets are computed from adjacent time intervals $[0, r], [r, 2r], \dots [\tau - r, \tau]$ (τ is a multiple of r) instead of points in time. This follows naturally from the computation of reachable sets using subintervals $[kr, (k+1)r]$; see Chap. 3. The transition probability for a subinterval is computed by substituting $\mathcal{R}_j^\alpha(\tau)$ in (4.17) by $\mathcal{R}_j^\alpha([kr, (k+1)r])$. These transition probabilities are finally averaged to the ones of the complete time interval:

$$\Phi_{ij}^\alpha([0, \tau]) = \frac{r}{\tau} \sum_{k=0}^{\tau/r-1} \Phi_{ij}^\alpha([kr, (k+1)r]). \tag{4.18}$$

Note that this transition matrix differs from the one obtained when first computing $\mathcal{R}_j^\alpha([0, \tau]) = \bigcup_{k=0}^{\tau/r-1} \mathcal{R}_j^\alpha([kr, (k+1)r]$ and then substituting $\mathcal{R}_j^\alpha(\tau)$ by $\mathcal{R}_j^\alpha([0, \tau])$ in (4.17).

The reason is that the reachable sets $\mathcal{R}_j^\alpha([kr,(k+1)r])$ are overlapping, which shows that the probability distribution of the states is not homogeneous. When $k \to \infty$, the computation in (4.18) returns the exact result under the assumption that the states are uniformly distributed for each point in time t within $\mathcal{R}_j^\alpha(t)$, whereas the computation error of the transition probabilities when using $\mathcal{R}_j^\alpha([0,\tau])$ remains for $k \to \infty$.

The abstraction using reachability analysis is visualized in Fig. 4.14. In Fig. 4.14(a), the reachable set $\mathcal{R}_j^\alpha([0,\tau])$ computed for the transitions from cell $\mathcal{X}_j$ is shown. The corresponding stochastic reachable set when starting from the same cell with probability 1 is illustrated in Fig. 4.14(b). A transition to a cell is the more likely, the darker the color of the cell is.



(a) Reachable set for a time interval.

(b) Reachable cells for a time interval.

Fig. 4.14.: Reachable set of the original system and the corresponding stochastic reachable set of the abstracting Markov chain.

The accuracy of the transition probabilities depends on the validity of the assumption that the state is uniformly distributed within the reachable sets. One source of violation is the over-approximation of the reachable sets so that the probability density value is 0 close to the borders of the reachable set. Another source is the consideration of system inputs. The effect of the input on the probability distribution of the state strongly depends on the system dynamics, the time step size τ, and the quantity of the input uncertainty. This effect is demonstrated for a simple scenario for which an analytical solution exists.

Example 4.1 (Exact Probability Distribution of an Integrator): Given is the simple system $\dot{\mathbf{x}} = a \cdot \mathbf{u}$ ($a \in \mathbb{R}$, $\mathbf{x}, \mathbf{u} : \Omega \to \mathbb{R}$), where $\mathbf{x}$ is the random state variable, $\mathbf{u}$ is the random input variable, and a is a constant parameter. The input $\mathbf{u}$ is constant for the time interval $[0,\tau]$. The exact solution at time τ is

$$\mathbf{x}(\tau) = \mathbf{x}(0) + a \cdot \mathbf{u} \cdot \tau = \mathbf{x}(0) + \Delta\mathbf{x}.$$

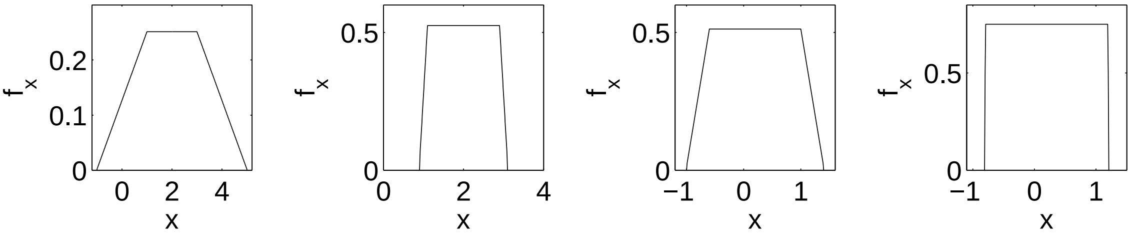
At $t = 0$ the random state $\mathbf{x}$ and the input $\mathbf{u}$ are uniformly distributed within the set of initial states $\mathcal{X}^0$ and the set of inputs $\mathcal{U}$:

$$f_{\mathbf{x}}(x,t=0)=\begin{cases}\frac{1}{\mathtt{V}(\mathcal{X}^0)}\ ,\text{ if } x\in\mathcal{X}^0\\ 0\ ,\text{ otherwise}\end{cases}\quad ,\quad f_{\mathbf{u}}(u)=\begin{cases}\frac{1}{\mathtt{V}(\mathcal{U})}\ ,\text{ if } u\in\mathcal{U}\\ 0\ ,\text{ otherwise}\end{cases}\quad .$$

Thus, the probability distribution of the change of the state $\Delta\mathbf{x}$ is also uniform, where $f_{\Delta\mathbf{x}}(x)=1/\mathtt{V}(a\cdot\mathcal{U}\cdot\tau)$ if $x\in a\cdot\mathcal{U}\cdot\tau$ and 0 otherwise. Further, x and u are statistically independent, so that the distribution of the state at time τ can be computed analytically using convolution (see [153]):

$$f_{\mathbf{x}}(x,t=\tau)=\int_{-\infty}^{\infty}f_{\mathbf{x}}(\chi,t=0)f_{\Delta\mathbf{x}}(x-\chi)\,d\chi. \qquad \square$$

The resemblance with a uniform distribution depends on the uncertainty of the input and the time step size τ, as can be seen in Fig. 4.15 for different combinations of time steps τ and input sets $\mathcal{U}$.



(a) $\tau=1,\mathcal{U}=[0,4]$. (b) $\tau=1,\mathcal{U}=[1.9,2.1]$. (c) $\tau=0.1,\mathcal{U}=[0,4]$. (d) $\tau=0.1,\ \mathcal{U}=[1.9,2.1]$.

Fig. 4.15.: Probability distribution of the state $f_{\mathbf{x}}$ after convolution.

These mentioned effects can be more or less accepted. However, there are cases in which one has to advise against the use of reachability analysis. One of these cases is when there exists a noisy and dominant input $\hat{u}(t)$ which has a probability distribution that is strongly non-uniform. It is also not advisable to abstract hybrid systems by reachability analysis, since the probability distribution is far from being uniform when the continuous dynamics is switched as shown in Fig. 4.16 for a two-dimensional example.

Hence, due to the more or less violated assumption of uniform probability distributions within reachable sets, the abstraction with Monte-Carlo simulation is more exact. This disadvantage is compensated by the property that the Markov chain obtained from reachability analysis is complete. This means that for each transition of an exactly generated Markov chain, there exists a corresponding transition of the Markov chain generated by reachability analysis. Thus, the reachable cells (cells with non-zero probability) over-approximate the reachable set of the original system.

Since the transition probabilities obtained from reachability analysis are generally not as accurate as the ones obtained from Monte Carlo simulation, one can combine Monte Carlo simulation with reachability analysis, as pointed out next.

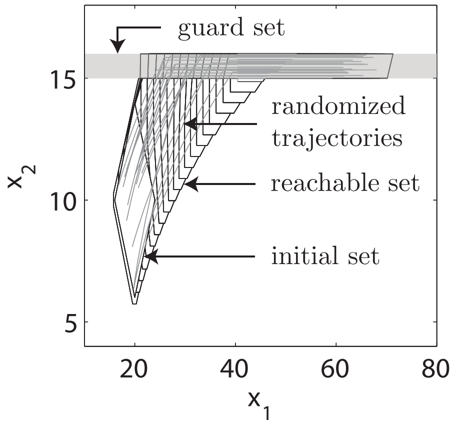


Fig. 4.16.: Change of the density of trajectories after a discrete transition. The dynamics changes to $\dot{x}_1 = v$ ($v = \texttt{const} \in \mathbb{R}$), $\dot{x}_2 = 0$.

Combination of Monte Carlo Simulation and Reachability Analysis

A possible way of combining Monte Carlo simulation and reachability analysis is to use the transition probabilities $\Phi^{\alpha}_{ij}([0,\tau])$ from Monte Carlo simulation, since they are more accurate. Next, transitions with zero probability of the Monte Carlo approach are replaced by a probability of ϵ when a non-zero probability is computed by the reachable set approach. The value of ϵ should be chosen small, e.g. $1/n_j^{\mathrm{sim},\alpha}$ and $n_j^{\mathrm{sim},\alpha}$ is the number of simulations; see (4.16). After inserting N_j probabilities with value ϵ into the j-th column, the transition probabilities of this column have to be normalized such that the sum is 1. The computation of a combined transition matrix $\widetilde{\Phi}^{\alpha}([0,\tau])$ is illustrated for the j-th column denoted by $\widetilde{\Phi}^{\alpha}_j([0,\tau])$ in the following example.

$$\Phi^{\alpha}_j([0,\tau]) = \begin{bmatrix} \vdots \\ 0 \\ \vdots \\ 0.4 \\ 0.6 \end{bmatrix}, \quad \widetilde{\Phi}^{\alpha}_j([0,\tau]) = \frac{1}{1 + N_j \cdot \epsilon} \begin{bmatrix} \vdots \\ 0 \\ \epsilon \\ 0.4 \\ 0.6 \end{bmatrix}.$$

This method can be applied to continuous and hybrid systems (deterministic and stochastic) with bounded parameter and input values ($\rho \in \mathcal{P} \subset \mathcal{I}^p, \quad (u(t) + \hat{u}(t)) \in \mathcal{U} \subset \mathbb{R}^m$). In contrast to the pure use of reachability analysis for hybrid systems, the combination with Monte Carlo simulation makes sense since the inaccurate transition probabilities are corrected. The advantages and disadvantages of the proposed methods for the abstraction of continuous/hybrid systems to Markov chains is summarized next.

Discussion

One of the advantages of the Monte Carlo approach is that the transition probabilities can be computed arbitrarily exact when the number of simulations tends to infinity. This is not the case for the reachability approach, since the assumption that the probability distribution is uniform is generally violated. Further, Monte Carlo simulation allows the use of arbitrary noise models such as Gaussian white noise as additional inputs, which is not possible for reachability analysis when the inputs are unbounded. If the noise was bounded, only the worst case effect could be considered, which causes inaccurate transition probabilities. On the other hand, only the reachability approach allows complete abstractions to be computed.

Both approaches can be combined as previously shown, which results in complete abstractions with accurate transition probabilities. Note that this combination is not possible if unbounded disturbances are considered. For hybrid systems, reasonable results are only obtained when using Monte Carlo simulation or its combination with reachability analysis.

The combination of simulation with advanced reachability analysis is an extension to previous work on abstraction to Markov chains; see e.g. [111, 113, 150]. Especially uncertain inputs are not considered in many works. The additional abstraction for time intervals is a novelty for which no other work is known to the best knowledge of the author.

4.3.3. Stochastic Reachable Sets from Markov Chains

The obtained transition probabilities from Monte Carlo simulation or reachability analysis make it possible to compute the probability distribution of the discrete states. A single Markov chain is updated according to $\hat{p}(k+1) = \Phi \hat{p}(k)$; see Def. 4.6. However, in this work, Markov chains were computed for different input cells $\mathcal{U}^\alpha$ and for points in time $t = \tau$ as well as time intervals $t \in [0, \tau]$. Hence, the update of the Markov chain probabilities $\hat{p}(k)$ has to be extended. First, the case is considered when the input for each time interval $[t_k, t_{k+1}]$ is known. Note that the input $\hat{u}(t)$ does not have to be considered anymore since its effect is already incorporated in the state transition probabilities of the Markov chain.

Deterministic Input

In order to introduce the extensions for the update of the Markov chain step-by-step, known inputs $u(t)$ are considered first. In addition, it is required that the input stays within the input cells $\mathcal{U}^\alpha$ for a time interval $[t_k, t_{k+1}]$, but may change the cell for the next time interval.

The first extension compared to the update of a single Markov chain is that not only the probability distribution at points in time, but also at time intervals is considered. The update for points in time $t_k = k \cdot \tau$ is done as for a regular Markov chain resulting in the probability vectors $\hat{p}(t_k)$. The probability vector of time intervals is computed from the probability vector at the beginning of the time interval $\hat{p}(t_k)$ to $\hat{p}([t_k, t_{k+1}]) = \Phi^\alpha([0, \tau])\, \hat{p}(t_k)$. This is because the transition probabilities for a time interval $\Phi^\alpha([0, \tau])$ were computed from the initial cell at the beginning of the time interval $[0, \tau]$. Thus, the solution at points in time is an auxiliary result for the time interval computation. After introducing the state transition matrix Φ^α containing all values of Φ^α_{ij} when the input value α is fixed, the updates can be formulated as

$$\begin{aligned} \hat{p}(t_{k+1}) &= \Phi^\alpha(\tau)\, \hat{p}(t_k), \\ \hat{p}([t_k, t_{k+1}]) &= \Phi^\alpha([0, \tau])\, \hat{p}(t_k), \end{aligned} \tag{4.19}$$

where α is chosen according to the actual discrete input. It is remarked that by the iterative multiplication of the probability distribution with the transition matrices, a further error is introduced: The probability distribution within one cell is treated as if it is replaced by a uniform distribution in the next time step. This error can be decreased by refining the discretization while increasing computational costs.

As a next extension, it is no longer assumed that the input $u(t)$ is exactly known.

Stochastic Input

Often, not only the state of a system, but also its input is uncertain. In order to update the Markov chain subject to uncertain inputs, the conditional probability $q_i^\alpha := P(\mathbf{y}=\alpha|\mathbf{z}=i)$ of the input for a given state $\mathbf{z}=i$ is introduced. Further, $\hat{p}_i = \sum_\alpha p_i^\alpha$ is the total probability of the state, where $\sum_\alpha$ denotes the sum over all possible values of α. The joint probability of the state and input is

$$p_i^\alpha := P(\mathbf{z}=i,\mathbf{y}=\alpha) = P(\mathbf{y}=\alpha|\mathbf{z}=i)\cdot P(\mathbf{z}=i) = q_i^\alpha \cdot \hat{p}_i. \tag{4.20}$$

Due to the uncertainty of the inputs, the Markov chains have to be updated for each possible value of α. After introducing the vector p^α containing all possible values of p_j^α for a fixed value of α, the joint probability vector p^α is updated to

$$\begin{aligned} p^\alpha(t_{k+1}) &= \Phi^\alpha(\tau)\, p^\alpha(t_k),\\ p^\alpha([t_k,t_{k+1}]) &= \Phi^\alpha([0,\tau])\, p^\alpha(t_k). \end{aligned} \tag{4.21}$$

The above formula updates the state distribution according to the state transition matrices. However, the conditional input probabilities q_i^α remain unchanged, since $\Phi^\alpha(\tau)$ and $\Phi^\alpha([0,\tau])$ only describe the mapping of the state probabilities. Hence, for changing input probabilities, the conditional input probabilities are updated instantaneously at times t_k so that $q_i^{\alpha\prime}(t_k) = \sum_\beta \Gamma_i^{\alpha\beta}(t_k) q_i^\beta(t_k)$. The values of $\Gamma_i^{\alpha\beta}(t_k)$ depend on the state i and for a fixed state, $\Gamma_i(t_k)$ is a time varying transition matrix for inputs. It is remarked that instead of updating the conditional probabilities q_i^α, one can also update the joint probabilities p_i^α because the state probability $\hat{p}_i$ does not change instantaneously:

$$q_i^\alpha(t_k)' = \sum_\beta \Gamma_i^{\alpha\beta}(t_k)\cdot q_i^\beta(t_k) \overset{\cdot \hat{p}_i(t_k) \text{ and } (4.20)}{\longrightarrow} p_i^\alpha(t_k)' = \sum_\beta \Gamma_i^{\alpha\beta}(t_k)\cdot p_i^\beta(t_k). \tag{4.22}$$

In order to simplify the notation and elegantly combine the state transition values Φ_{ij}^α with the input transition values $\Gamma_i^{\alpha\beta}$, the joint probabilities p_i^α are combined to a new probability vector

$$\tilde{p}^T = \left[\begin{array}{ccccccccccc} p_1^1 & p_1^2 & \dots & p_1^c & p_2^1 & p_2^2 & \dots & p_2^c & p_3^1 & \dots & p_d^c \end{array}\right]. \tag{4.23}$$

The values d and c refer to the number of discrete states and inputs, respectively. The construction of the new probability vector $\tilde{p}$ requires the state transition and input transition values to be rearranged, too.

$$\begin{aligned}
\tilde{\Phi} &= \left[\begin{array}{ccccccccccccccc}
\Phi^1_{11} & 0 & 0 & \dots & 0 & \Phi^1_{12} & 0 & 0 & \dots & 0 & \Phi^1_{13} & 0 & 0 & \dots & 0 \\
0 & \Phi^2_{11} & 0 & \dots & 0 & 0 & \Phi^2_{12} & 0 & \dots & 0 & 0 & \Phi^2_{13} & 0 & \dots & 0 \\
\vdots & & & & & & & & & & & & & & \vdots \\
0 & 0 & 0 & \dots & \Phi^c_{d1} & 0 & 0 & 0 & \dots & \Phi^c_{d2} & 0 & 0 & 0 & \dots & \Phi^c_{dd}
\end{array}\right], \\
\tilde{\Gamma} &= \left[\begin{array}{cccccccccc}
\Gamma^{11}_1 & \Gamma^{12}_1 & \dots & \Gamma^{1c}_1 & 0 & \dots & 0 & 0 & \dots & 0 \\
\Gamma^{21}_1 & \Gamma^{22}_1 & \dots & \Gamma^{2c}_1 & 0 & \dots & 0 & 0 & \dots & 0 \\
\vdots & & & & & & & \vdots & & \vdots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & \Gamma^{c1}_d & \dots & \Gamma^{cc}_d
\end{array}\right] = \left[\begin{array}{ccccc}
\Gamma_1 & \mathbf{0} & \dots & \dots & \mathbf{0} \\
\mathbf{0} & \Gamma_2 & \mathbf{0} & \dots & \mathbf{0} \\
\vdots & & \ddots & & \vdots \\
\mathbf{0} & \dots & \dots & \mathbf{0} & \Gamma_d
\end{array}\right],
\end{aligned} \tag{4.24}$$

and $\mathbf{0}$ is a matrix of zeros. The rewriting allows the state update in (4.21) to be formulated as $\tilde{p}(t_{k+1}) = \tilde{\Phi}(\tau)\tilde{p}(t_k)$ (for points in time) and the input update in (4.22) as $\tilde{p}(t_k)' = \tilde{\Gamma}(t_k)\tilde{p}(t_k)$. Combining both results yields the overall update equations

$$\begin{aligned}
\tilde{p}(t_{k+1}) &= \tilde{\Gamma}(t_k)\,\tilde{\Phi}(\tau)\,\tilde{p}(t_k), \\
\tilde{p}([t_k, t_{k+1}]) &= \tilde{\Phi}([0,\tau])\,\tilde{p}(t_k).
\end{aligned} \tag{4.25}$$

The reason why $\tilde{\Gamma}(t_k)\,\tilde{\Phi}(\tau)$ is not combined to a new matrix is that $\tilde{\Gamma}(t_k)$ is possibly time varying and thus has to be multiplied with the state transition matrices for each time step anyway. Note that the multiplication with $\tilde{\Gamma}(t_k)$ is not required for the time interval solution since it is based on the time point solution.

Once the abstraction of the original system has been performed, it is only left to perform the matrix multiplications in (4.25) for the computation of the probability distribution. It is remarked that the matrices have dimension $d \cdot c \times d \cdot c$, which might become large. However, the obtained matrices are very sparse (only a few non-zero entries), such that the multiplication can be accelerated by using sparse matrix multiplication algorithms [175]. Thus, the abstraction of the original system to the Markov chain is the computationally most expensive part. For this reason, the Markov chain abstraction is especially useful when one has to predict the safety of a system during its operation. In this scenario, one can compute the abstraction offline without constraints on the computational time. During the online operation of the system, the probability distribution can be computed efficiently using matrix multiplications in (4.25). The advantage that expensive computations can be performed beforehand will be of use when applying this approach to the safety analysis of road traffic scenes.

4.3.4. Numerical Examples

The abstraction to a Markov chain is exemplarily performed for the same two-dimensional system that has been used throughout this thesis in Chap. 3 and 4. The five-dimensional example that has been used in previous examples cannot be abstracted since the number of discrete states would exceed computationally manageable dimensions. For better readability, the two-dimensional example is displayed again, where the interval of the uncertain input varies from previous examples:

$$\dot{x} = \begin{bmatrix} -1 & -4 \\ 4 & -1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t), \quad u(t) \in [-1, 1]$$

For the abstraction to a Markov chain, the state space region $\mathcal{X}$ to be discretized is chosen as $[-2, 2]$ for both dimensions and each of them is partitioned into 30 segments resulting in 900 discrete states. The input is discretized into 5 equidistant intervals from -1 to 1. For the Markov chain abstraction, the Monte Carlo simulation approach and the reachable set approach have been used and combined as described in Sec. 4.3.2. The ϵ value for combining both approaches has been selected to $\epsilon = 1/500$ for points in time and $\epsilon = 1/50000$ for time intervals where 500 and 50000 are the number of simulations for the simulative abstraction of the Markov chain. The time step size for the abstracted Markov chain in $\tau = 0.5$. Examples of the computation of transition probabilities for the initial cell $\mathcal{X}_{70}$ and the input cell $\mathcal{U}_1$ can be found in Fig. 4.13 and 4.14 using the Monte Carlo approach and the reachable set approach, respectively.

The input transition matrix is not dependent on the discrete state and is given as

$$\Gamma = \begin{bmatrix} 0.7 & 0.2 & 0.1 & 0 & 0 \\ 0.2 & 0.5 & 0.2 & 0.1 & 0 \\ 0.1 & 0.2 & 0.4 & 0.2 & 0.1 \\ 0 & 0.1 & 0.2 & 0.5 & 0.2 \\ 0 & 0 & 0.1 & 0.2 & 0.7 \end{bmatrix}.$$

The initial probability distribution of the inputs is chosen independently of the state distribution and is set to $\begin{bmatrix} 0 & 0.2 & 0.6 & 0.2 & 0 \end{bmatrix}$. The initial set is chosen such that it is uniformly distributed in the interval $x_1 \in [0.9, 1.1]$ and $x_2 \in [0.9, 1.1]$, which is then mapped to the distribution of the state space cells.

The average probability distribution for different time intervals is computed as

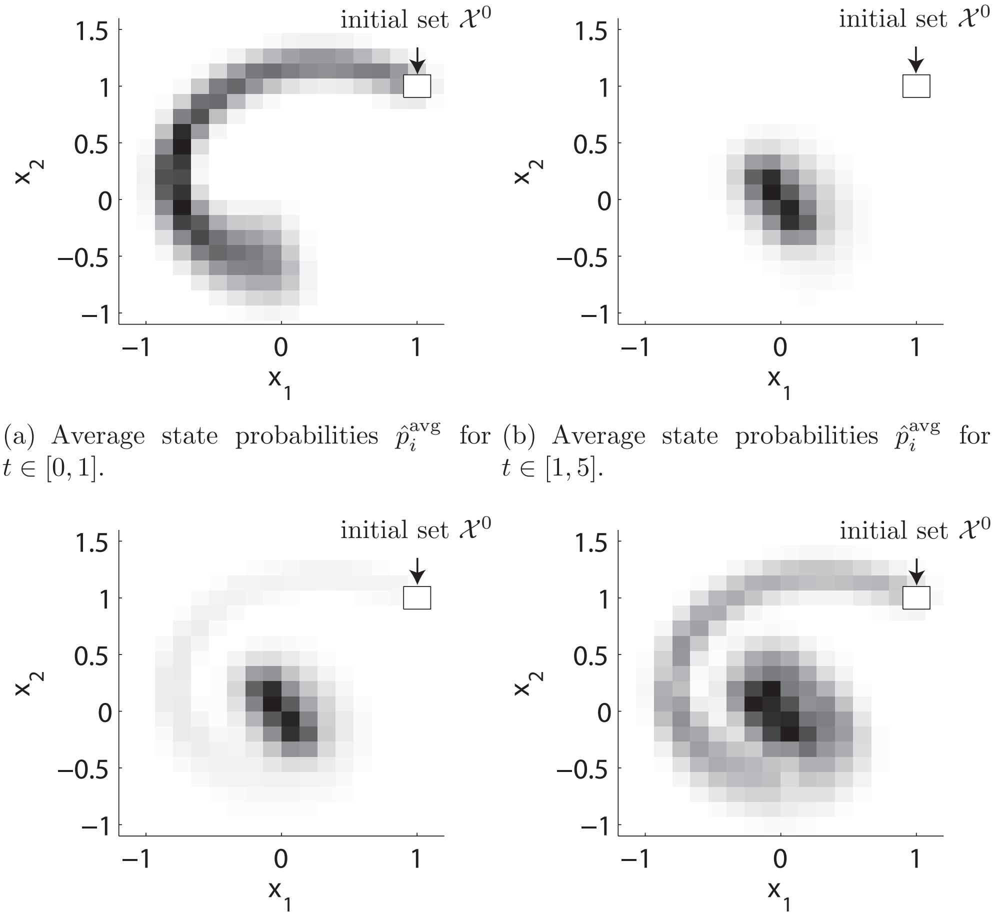
$$\tilde{p}^{\mathrm{avg}} = \frac{1}{n_{\mathrm{end}} - n_{\mathrm{start}}} \sum_{k=n_{\mathrm{start}}}^{n_{\mathrm{end}}} \tilde{p}(t_{k+1}),$$

where $n_{\mathrm{start}}, n_{\mathrm{end}} \in \mathbb{N}^+$ and $n_{\mathrm{end}} > n_{\mathrm{start}}$. For the visualization, the maximum probability values of state space cells are also of interest. This distribution is computed as

$$\tilde{p}^{\max} = \mathtt{max}(\tilde{p}(t_{n_{\mathrm{start}}}), \ldots, \tilde{p}(t_{n_{\mathrm{end}}})).$$

Note that $\tilde{p}$ contains the combined probabilities of the state and the input. In order to display the state probability only, the total probabilities of the state have to be computed. Rewriting the combined probabilities $\tilde{p}$ to p_i^α by reverse use of (4.23), the total probability of the state is obtained as $\hat{p}_i = \sum_\alpha p_i^\alpha$. The average state probability $\hat{p}_i^{\mathrm{avg}}$ and maximum state probability $\hat{p}_i^{\max}$ are visualized in Fig. 4.17 for different time intervals.

The computational time is 0.06 s for 10 time intervals resulting in a prediction horizon of $t \in [0, 5]$. The computations were performed on an AMD Athlon64 3700+ processor (single core) in Matlab.



(a) Average state probabilities $\hat{p}_i^{\text{avg}}$ for $t \in [0, 1]$.

(b) Average state probabilities $\hat{p}_i^{\text{avg}}$ for $t \in [1, 5]$.

(c) Average state probabilities $\hat{p}_i^{\text{avg}}$ for $t \in [0, 5]$.

(d) Maximum state probabilities $\hat{p}_i^{\max}$ for $t \in [0, 5]$.

Fig. 4.17.: Average and maximum probability distribution for different time intervals.

4.4. Summary

The concept of reachability analysis has been extended to stochastic reachability analysis in this chapter. The definition of stochastic reachability analysis introduced in this thesis differs from the established concept of determining the probability that the state reaches an unsafe set. According to the definition used in this thesis, the stochastic reachable set of a point in time is used as a synonym for the probability density function of the state. An extension to time intervals is obtained by averaging the probability density function over all points in time of the considered time interval. This definition is a natural extension of reachable sets since a reachable set is the set for which the stochastic reachable set has non-zero probability values. The probability that the state is in an unsafe set is obtained by integrating the stochastic reachable set within the unsafe set. If one is interested in the probability that the state has reached an unsafe set, instead of being in an unsafe set, one has to ensure that the unsafe set is absorbing, i.e. the state cannot leave the unsafe set once it has entered it. This can be done by specifying an absorbing dynamics within the unsafe set such that the unsafe set can be interpreted as an invariant of a hybrid system with no transitions to other locations.

One of the biggest challenges in stochastic reachability analysis is the design of scalable algorithms. The curse of dimensionality is an even bigger issue for stochastic reachability than for classical reachability analysis since most algorithms rely on a discretization of the state space causing exponential complexity with respect to the number of continuous state variables. Another issue is that only barrier certificates can compute upper bounds on the probability of reaching unsafe states.

These problems have been tackled for linear systems with enclosing probabilistic hulls. The input of the linear system is modeled as Gaussian white noise with uncertain mean which allows non-Gaussian white noise to be modeled in an over-approximative way. Due to the use of enclosing probabilistic hulls, which enclose all possible probability density functions, the probability of being in an unsafe set can be over-approximated. Note that this probability differs from computing the probability of reaching an unsafe set. As discussed above, both results are identical when ensuring that the unsafe set is absorbing. Algorithms for this case using enclosing probabilistic hulls are part of future work. An advantage of the presented approach is that it can be applied to large systems with more than 100 continuous state variables.

The second considered approach deals with the abstraction of continuous or hybrid systems to Markov chains. This concept can be applied to a large class of systems, but only small-scale problems can be handled due to the required discretization of the continuous state space. Two different methods for the abstraction process have been presented: Abstraction by Monte Carlo simulation and by reachability analysis. Monte Carlo simulation is accurate, but the resulting Markov chain is not complete, i.e. does not cover all possible behaviors of the original system. On the other hand, abstraction by reachability analysis is less accurate, but the resulting Markov chain is complete. The combination of both methods yields good results since accuracy and completeness can be unified.

An unfavorable property of the Markov chain abstraction is that input probabilities can only be changed by the update rate of the Markov chain. If frequent changes of the input probability are necessary, the time step size of the Markov chain has to be shortened. An interesting property of the presented Markov chain abstraction techniques is that most of the computation time is spent on the abstraction so that they can be used for the online safety assessment of time critical applications. Markov chains can run especially efficiently on dedicated hardware such as DSPs, since only multiplication of sparse matrices have to be computed.

5. Safety Assessment of Autonomous Cars

In this chapter, the generic methods on stochastic reachability analysis are applied to the safety assessment of autonomous vehicles. Ultimately, the presented safety verification module should assess various alternatively planned driving actions of autonomous cars according to their safety.

5.1. Introduction and State of the Art

First, some challenges of autonomous driving, i.e. driving without a human driver, are presented.

Autonomous Driving

A prerequisite of autonomous driving is the equipment of vehicles with sensors for the detection of their environment. More importantly, relevant information such as the position and velocity of other traffic participants has to be correctly extracted from the raw data streams of the sensors. This has to work properly in different weather conditions and even when unknown or unexpected objects are present. Besides the enormous challenge of detecting objects such as other vehicles or lane markings, it is also desirable to estimate the intentions of other traffic participants in order to derive the optimal behavior for the autonomous vehicle.

Human drivers are very good at recognizing other traffic participants, estimating their intention, and planning almost optimal driving actions. Besides these capabilities, humans can focus on the relevant information, handle unexpected situations, and automatically learn from previously unknown situations. It is obvious that researchers aim to partly implement those cognitive capabilities into an autonomously driving car. This has been tried in many research projects, including the collaborative research center *Cognitive Automobiles* [154], in which this work has been carried out.

One of the main objectives of the research on autonomous vehicles is the vision of accident-free driving by exclusion of human errors. Worldwide, the number of people killed in road traffic each year is estimated at almost 1.2 million, while the number of injured is estimated at 50 million [134, chap. 1]. However, autonomous cars will not be market-ready soon, such that mature driving capabilities of autonomous prototype vehicles will be incorporated into intelligent driver assistant systems of market-ready vehicles. It is e.g. desirable that a driver assistant system fully controls a vehicle when a crash is almost inevitable. Predicting the probability of a crash is subject of the safety assessment developed for autonomous cars in this thesis.

Safety Assessment in Road Traffic

In order to assess the safety of a planned maneuver, the predicted motion of other traffic participants is vital for the identification of future threats. For this reason, the prediction of other traffic participants is one of the main objectives in this chapter. In contrast to this approach, non-predictive methods are based on the recording and evaluation of traffic situations that have resulted in dangerous situations; see e.g. [4]. However, such an approach is only suitable for driver warnings. Planned trajectories of autonomous cars cannot be evaluated with non-predictive methods since the consequences when following these trajectories have to be predicted.

Behavior prediction has been mainly limited to human drivers within the *ego vehicle* (i.e. the vehicle for which the safety assessment is performed). This is motivated by research on driver assistant systems which tries to warn drivers when dangerous situations are ignored. The majority of works on this topic use learning mechanisms (e.g. neural networks, autoregressive exogenous models [151, 174]), or filter techniques (e.g. Kalman filters [109, 136]). Another line of research is to detect traffic participants on selected road sections and predict their behavior for anomaly detection. Such automatic surveillance system has been realized with learning techniques such as clustering methods [86] or hidden Markov models [123]. The disadvantage of a prediction at fixed locations is that the predictions are specialized to this particular road segment and probably not generalizable to other traffic situations.

For the prediction of arbitrary traffic situations, simulations of traffic participants have been used [18, 84]. Due to the efficiency of single simulations, these approaches are already widely implemented in cars, e.g. to initiate an emergency braking maneuver based on measures like *time to collision* or *predicted minimum distance*. Simulations of traffic participants are also computed in microscopic traffic simulations [119, 162]. However, single simulations do not consider uncertainties in the measurements and actions of other traffic participants, which may lead to unsatisfactory collision predictions [108]. A more sophisticated threat assessment considers multiple simulations of other vehicles, considering different initial states and changes in their inputs (steering angle and acceleration). These so-called Monte-Carlo methods have been studied in [15, 31, 32, 52, 59, 60] for the risk analysis of road traffic and in [25, 26, 168] for the related topic of air traffic safety. A framework for the reduction of possible future scenarios of traffic situations, using motivations for the actions of drivers, is introduced in [46].

Another method to compute possible behaviors of traffic participants is reachability analysis as presented in Chap. 3. Safe motion of two vehicles is guaranteed if their reachable sets of positions do not intersect. In traffic scenarios, the reachable positions of a vehicle define the unsafe set of another vehicle, and vice versa. Reachable sets for vehicles and mobile robots have been investigated in [149, 165]. It has been shown that planned paths of autonomous vehicles are too often evaluated as unsafe by this method. This is because the reachable sets of other vehicles rapidly cover all positions the autonomous vehicle could possibly move to, which is demonstrated in Fig. 1.8(a).

A combination of reachability analysis and stochastic reachability analysis has been investigated in [77]. The reachable sets of traffic participants are used to find out which vehicles might have a crash. Next, the stochastic reachable sets are only computed for those traffic participants that might crash in order to save computational time. Note that the reachable sets in [77] are not over-approximative and that the algorithm for the computation is rather fuzzy. The stochastic reachable sets are described by Gaussian distributions, which are obtained by several linearized models. The concept of detecting relevant traffic participants by reachability analysis can be analogously applied to the concept in this work.

Contributions

In this chapter, *stochastic reachable sets* of traffic participants are computed as previously shown in Chap. 4. The stochastic information allows not only to check if a planned path of the ego vehicle may result in a crash, but also with which probability. Consequently, possible driving strategies of autonomous cars can be evaluated according to their safety. Traffic participants are predicted by Markov chains as presented in Sec. 4.3. There are three properties which are in favor of the Markov chain approach: The approach can handle the hybrid dynamics of traffic participants, the number of continuous state variables (position and velocity) is low, and Markov chains are computationally inexpensive when they are not too large.

The contributions in more detail are: A basic concept for the safety assessment of autonomous cars in Sec. 5.2. A mathematical model of traffic participants in Sec. 5.3 and their abstraction to Markov chains in Sec. 5.4. Further, a stochastic generation of driving commands, which is addressed in Sec. 5.5 for certain driving capabilities: road following, vehicle following, intersection crossing, and lane changing. It is also discussed how driving capabilities are unified and how to handle capabilities which are not implemented, such as parking. The driving command generation and the Markov chains allow the position distribution of other traffic participants to be predicted. How to compute the crash probability for the autonomous vehicle, given the position probabilities of other traffic participants, is presented in Sec. 5.6. In order to evaluate the Markov chain approach, it is tested against Monte Carlo simulation in Sec. 5.7. The introduced Markov chain approach is also tested in the autonomous vehicle *MUCCI* on a test ground, which is documented in Sec. 5.8. Finally, the chapter is summarized in Sec. 5.9.

5.2. Basic Concept

Clearly, autonomous driving requires a control loop containing a perception and a planner module; see Fig. 5.1. The perception module detects traffic situations and extracts relevant information, such as the road geometry as well as static and dynamic obstacles. In order to fulfill the driving task, the planner module computes trajectories that the autonomous car is tracking with the use of low-level controllers [173]. A major constraint for the trajectory planner is that the generated trajectories have to be safe, i.e. static and dynamic obstacles must not be hit. The task of circumventing static obstacles can be ensured by checking whether the static obstacle intersects with the vehicle body of the autonomous car when

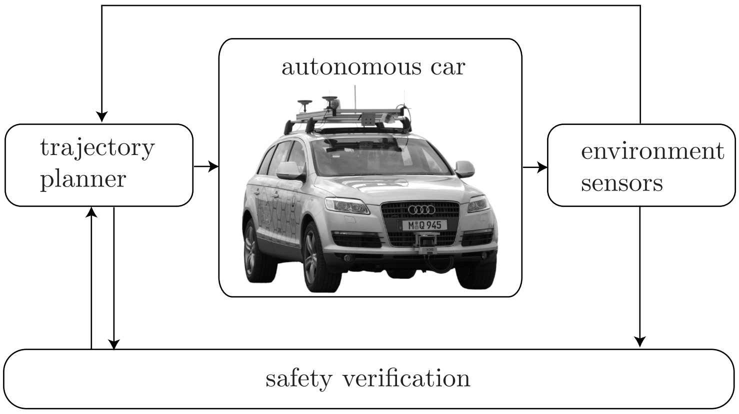


Fig. 5.1.: Conception of the safety assessment.

following the planned path. For dynamic obstacles, the safety assessment is much more intricate as their future actions are unknown. For this reason, sets of possible behaviors of other traffic participants are considered, which are checked with the planned path of the autonomous car in a dedicated safety verification module (see Fig. 5.1). Paths that fulfill the safety requirements are executed or are conservatively replanned otherwise, e.g. by braking the car.

The safety verification module which is described in this work requires the description of a traffic situation containing the following information gathered by the perception and planner modules:

- the planned trajectory of the autonomous car,
- the geometric description of relevant road sections,
- the position and geometry of static obstacles,
- as well as the position, velocity, and classification of dynamic obstacles.

Static obstacles are a special case of dynamic obstacles with zero velocity, and for that reason the discussion is continued for dynamic obstacles only. The classifier of the autonomous vehicle groups the dynamic obstacles ($\widehat{=}$ traffic participants) into *cars*, *trucks*, *motorbikes*, *bicycles*, and *pedestrians*. As the measurement of positions and velocities of other traffic participants is uncertain, the presented approach allows the measured data to be specified by a probability distribution. However, it is required that all relevant traffic participants are detected. Given the information of the perception module, the future stochastic reachable set of all traffic participants is computed, from which the probability distribution of the position can be extracted. The positions with non-zero probability value belong to the set of reachable positions, which serves as a time varying unsafe set for the autonomous car[1].

If the reachable positions of other traffic participants do not intersect with the ones of the autonomous car for a prediction horizon t_f, safety can be guaranteed within the specified horizon. Where a crash is possible, the probability of the crash is computed from the probability distribution within the reachable set. This is illustrated in Fig. 5.2, where

[1] The ego car and the autonomous car refer to the same car in this thesis. However, it is possible that other vehicles are autonomous vehicles, too.

stochastic reachable sets are shown for the time intervals $\tau_1 = [0, t_1]$, $\tau_2 = [t_1, t_2]$ (dark color indicates high probability density). Within the time interval τ_1, a crash between both cars is impossible since their stochastic reachable sets do not intersect, while for the second time interval τ_2, the crash probability is non-zero. It is obvious that all necessary computations have to be faster than real time to allow an online application. In order to update the crash probability prediction after a time interval Δt based on new sensor values, its computation has to be faster than real time by a factor of $t_f/\Delta t$. This is illustrated in Fig. 5.3 for the reachable set of a single variable $x(t)$.

The mathematical model of other traffic participants used for the computation of their stochastic reachable sets is introduced next.

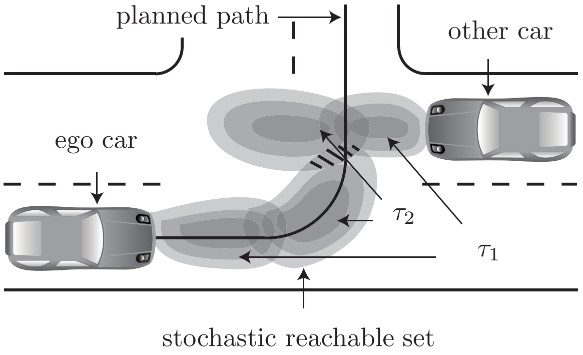


Fig. 5.2.: Stochastic reachable sets of traffic participants.

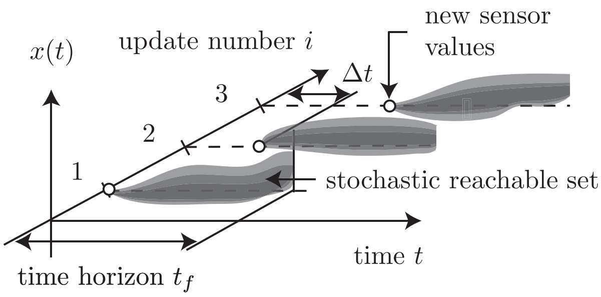


Fig. 5.3.: Repetitive computation of reachable sets.

5.3. Modeling of Traffic Participants

The presented safety assessment focuses on autonomous cars driving on a road network, i.e. the motion of traffic participants is constrained along designated lanes. On that account, the prediction of traffic participants is performed in two steps. Firstly, the lanes most likely followed by traffic participants are determined by high-level behaviors. Secondly, the dynamics of traffic participants along the corresponding paths on the lanes is considered. The same concept is applied in [77].

Possible paths of traffic participants are modeled by the finite set of high-level behaviors {*left turn*, *right turn*, *go straight*}, and {*left lane change*, *right lane change*} on a multi-lane road. Further high-level behaviors such as *parking* or *overtaking* can be included in the modeling scheme later. In the continuation of this work, it is planned to automatically derive such high-level behaviors by observation and clustering of traffic scenes. It is noted that the high-level behavior does not have to be exactly known, since one can also compute with probabilities of high-level behaviors.

In unstructured environments, such as parking spaces or pedestrian zones, the motion of vehicles/people cannot be described along paths. For these kinds of scenarios, the approach presented in [195] is suggested, which uses the same mathematical principles as presented later, but applies them to unstructured environments. The prediction in unstructured environments also serves as a fallback solution when the observed high-level behavior cannot be categorized in one of the given or additionally learned high-level behaviors.

Note that the introduced model is only used for other traffic participants. The ego car does not have to be modeled since its future behavior is already determined by its trajectory planner.

5.3.1. Lateral Dynamics

The deviation along the paths of traffic participants is modeled by a piecewise constant probability distribution $f(\delta)$, where δ is the lateral deviation from a path. Possible paths of a road network section, as well as the deviation probability distribution $f(\delta)$, are shown in Fig. 5.4. The deviation probability can be adjusted to different classes of traffic participants: Bicycle riders are more likely to be found close to the curb, whereas cars and trucks are more likely to be driving in the center of a lane. A statistical analysis of lateral displacement of vehicles on a road can be found in [54]. The deviation probabilities are normalized to the width of the lanes so that typical distributions can be applied independently of the lane width.

In this thesis, the deviation probability is held constant over time and the more complex case of dynamically changing lateral distribution is the subject of future work. Another assumption is that the deviation probability is computed independently of the probability distribution along the path. This is a reasonable assumption since the task of following the desired path is more or less independent of the task of keeping the speed or the distance to other vehicles. Additionally, this assumption drastically simplifies the computation of probabilistic reachable sets of other traffic participants, because the lateral and longitudinal probability distribution can be computed separately in low dimensional spaces. Thus, given the lateral probability distribution $f(\delta)$ and the longitudinal probability distribution $f(s)$, the overall probability distribution is computed as $f(s, \delta) = f(s) \cdot f(\delta)$; see Fig. 5.4. The combined probability distribution is described in a curved, path-aligned coordinate system as also used in e.g. [60]. It is emphasized that the lateral and the longitudinal distribution $f(\delta)$ and $f(s)$ refer to the position s and the deviation δ of the volumetric center of the vehicles. However, for visualization reasons, all figures in this work show the density of the vehicle body, which takes the vehicle size into account; see Fig. 5.5.

The longitudinal probability distribution is obtained from a dynamical model which is explained in the next subsection.

5.3.2. Longitudinal Dynamics

For the longitudinal dynamics model, the position of a vehicle along a path s, the velocity v, and the absolute acceleration a have to be introduced. The acceleration command u is normalized and varies from $[-1, 1]$, where -1 represents full braking and 1 represents full acceleration. Further, the function $\rho(s)$ is introduced which maps the path coordinate s to the radius of curvature. The radius of the path determines the tangential acceleration a_T for a given velocity v and thus limits the normal acceleration a_N, since the absolute value of the combined accelerations has to be less than the maximum absolute acceleration $a^{\max}$. In addition, the acceleration dynamics changes at the switching velocity v^{sw}. The

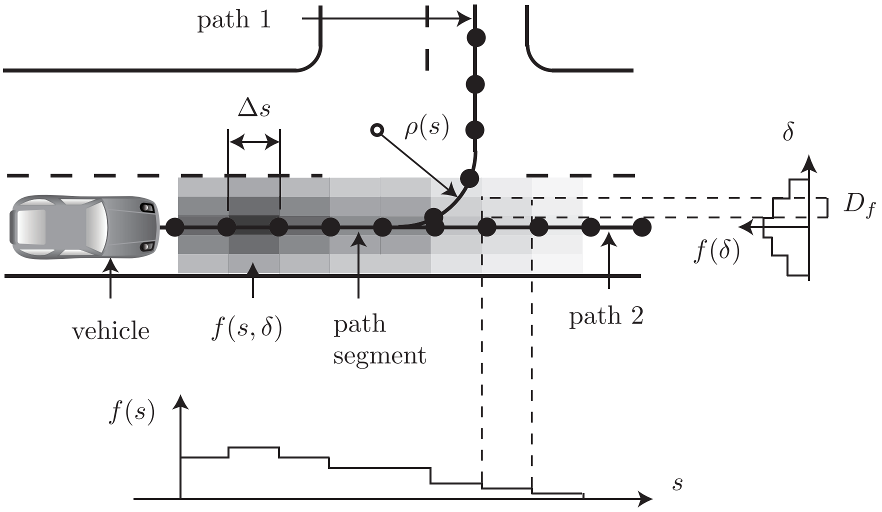


Fig. 5.4.: Position distribution $f(s, \delta) = f(s) \cdot f(\delta)$ along a path-aligned coordinate system, which is composed of the longitudinal and lateral distribution. D_f is a deviation segment and $\rho(s)$ is the radius of curvature along a path.

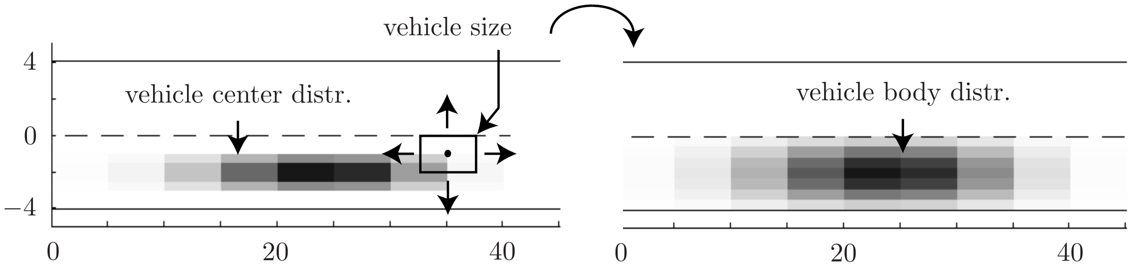


Fig. 5.5.: Probability distribution of the vehicle center and the vehicle body. The coordinate axes refer to positions in [m].

differential equations for the longitudinal dynamics are chosen as proposed in [60]:

$$\dot{s} = v, \quad \dot{v} = \begin{cases} a^{\max} u, & 0 < v \le v^{\mathrm{sw}} \vee u \le 0 \\ a^{\max} \frac{v^{\mathrm{sw}}}{v} u, & v > v^{\mathrm{sw}} \wedge u > 0 \\ 0, & v \le 0 \end{cases} \tag{5.1}$$

subject to the constraint

$$|a| \le a^{\max}, \text{ where } |a| = \sqrt{a_N^2 + a_T^2}, \; a_N = v^2/\rho(s), \; a_T = \dot{v}. \tag{5.2}$$

Backwards driving on a lane is not considered; see (5.1) ($\dot{v} = 0$, $v \le 0$). The constraint in (5.2) models that the tire friction of a vehicle only allows a limited absolute acceleration $a^{\max}$ (Kamm's circle). The constants $a^{\max}$ and v^{sw} can be chosen according to the specific properties of the different classes of traffic participants. The values used in this thesis when not stated differently are listed in Tab. 5.1.

The differential equations in (5.1) are chosen exemplarily and can be easily exchanged against a different set of equations. For example, in previous publications [182, 183, 185,

Tab. 5.1.: Vehicle parameters.

	Car	Truck	Motorbike	Bicycle
$a^{\max}$ [m/s^2]	7	7	7	7
v^{sw} [m/s]	7.3	4	8	1

186, 188, 190], the vehicle model for the case $u > 0$ is $\dot{v} = (a^{\max} - c_2\, v^2)\, u$, where c_2 is a constant. This model considers aerodynamic drag, but does not consider that the acceleration force decreases with velocity since $m\, a \leq P^{\max}/v$, where m is the mass and $P^{\max}$ is the maximum acceleration power. Another possibility is to enhance the model according to Eidehall in (5.1) with aerodynamic drag so that the model for $v > v^{\mathrm{sw}} \wedge u > 0$ is $\dot{v} = \left(a^{\max}\frac{v^{\mathrm{sw}}}{v} - c_1\, v^2\right) u$, where v^{sw} and c_1 are constants. The different models are compared in Fig. 5.6 to a highly realistic vehicle model of the Audi Q7, which serves as the platform for the experimental vehicle *MUCCI* which is presented later. Note that the peaks in acceleration occur due to the torque converter of the automatic gear box in the Audi Q7. The model proposed by Eidehall and the one enhanced with aerodynamic drag match the realistic vehicle model best. However, since the model of Eidehall is accurate enough and simpler than the enhanced version, the model proposed in (5.1) is used throughout this thesis.

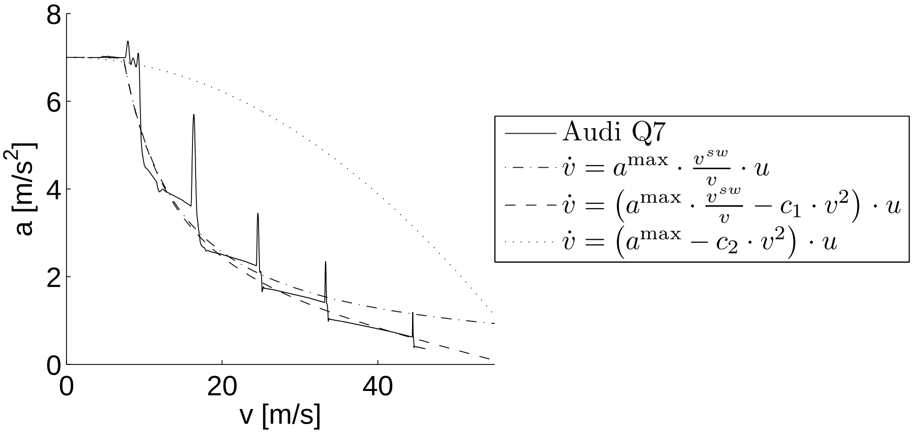


Fig. 5.6.: Maximum acceleration a of the Audi Q7 plotted over its velocity v; used parameters for the compared models: $a^{\max} = 7$ [m/s^2], $v^{\mathrm{sw}} = 7.3$ [m/s], $c_1 = 2.8e - 4$ [m], $c_2 = 1.9e - 3$ [m].

5.3.3. Violation of Traffic Regulations

The safety assessment approach presented in this thesis initially assumes that the traffic participants respect the traffic rules, e.g. they do not violate speed limits[2] or they do not drive in the wrong lanes (driving against oncoming traffic).

[2] In order to account for sporty drivers, the speed limit can be set higher than the official speed limit. The consideration of speed limits is presented in detail in Sec. 5.5.2.

The assumption that drivers stay in allowed lanes is considered by only allowing non-zero deviation probabilities within the span of the allowed lanes. Note that the determination of allowed lanes is not trivial since, for example, a vehicle may use a lane that is usually used by oncoming traffic if an obstacle has to be circumvented.

There are two strategies for considering violations of traffic regulations. One possibility is to assume that a non-conform driver is a reckless driver, e.g. the speed limit assumption, the assumption that this driver stays in allowed lanes, etc., is not considered anymore in the prediction. Another possibility is to only disable the speed limit regulation if only the speed limit is violated or only disable the allowed lanes assumption if this assumption is violated, etc. A satisfying answer for choosing the correct strategy is yet to be found.

The computation of the longitudinal probability distribution of traffic participants with Markov chains is presented next.

5.4. Abstraction of Traffic Participants to Markov Chains

The dynamic model of traffic participants (5.1) introduced in the previous section is hybrid with nonlinear continuous dynamics. Since this model is nonlinear, the enclosing hull method of Sec. 4.2 for the computation of stochastic reachable sets cannot be applied. However, due to the low dimensionality of the vehicle model, the Markov chain abstraction of Sec. 4.3 can be applied. The advantage of this approach is that the computationally intensive abstraction is computed offline. During the operation of the autonomous vehicle, the stochastic reachable sets of the Markov chains can be computed efficiently. It is again noted that the abstraction to Markov chains is only applied to other traffic participants, while the behavior of the ego vehicle is known from the trajectory planner.

In Sec. 4.3 on Markov chain abstraction, two methods were presented: Abstraction by Monte Carlo simulation and abstraction by reachability analysis. Monte Carlo simulation yields more accurate transition probabilities while reachability analysis yields a complete abstraction. It has also been shown that both methods can be combined so that both positive properties are unified. However, for the dynamics of traffic participants as specified in (5.1), there is no need for a complete abstraction. This is because the reachable position can be efficiently computed by two simulations, as shown in the next subsection. Thus, the probability that a crash may occur can be answered by the reachable positions, while the probability of a crash is answered by the stochastic reachable set. If there is contradicting information, i.e. a reachable set intersects, but the corresponding stochastic reachable set does not (due to an incomplete Markov chain abstraction), a low crash probability is assumed.

5.4.1. Reachable Set of Traffic Participants

This subsection presents an efficient computation of reachable sets for traffic participants as previously defined in Sec. 5.3. Writing the dynamics of a traffic participant as $\dot{x} = f^{TP}(x(t), u(t))$, where $x \in \mathbb{R}^2$ is the state and $u \in [-1, 1]$ is a Lipschitz continuous input, the exact reachable set $\mathcal{R}^e(r)$ at time $t = r$ is defined as

$$\mathcal{R}^e(r) = \Big\{x(r) = x(0) + \int_0^r f^{TP}(x(\tau), u(\tau))\, d\tau \big| x(0) \in \mathcal{X}^0, u(\tau) \in [-1, 1]\Big\}.$$

In general, the exact reachable set of a system cannot be computed [106]. However, one can always compute an over-approximation as presented in Chap. 3. An example of the over-approximated reachable set of a traffic participant according to (5.1) for $u = 1$ and $t \in [0, 2]$ s is shown in Fig. 5.7 for two different initial sets. Additionally, sample trajectories starting from the initial set are shown, where the states at times $k \cdot \Delta t^*$, $k = 0 \dots 4, \Delta t^* = 0.5$ s are marked by a circle. If one is only interested in the reachable interval of the position and the velocity of a vehicle driving along a straight path, the following special case can be formulated:

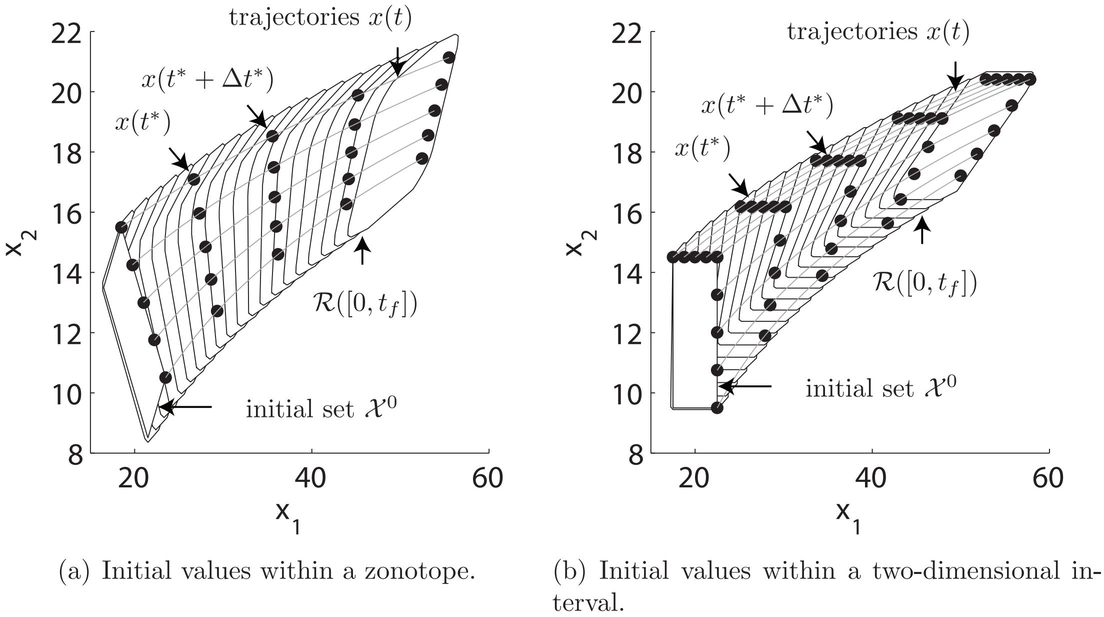
Proposition 5.1 (Reachable Two-Dimensional Interval of the Vehicle State): Given is a vehicle driving along a straight path with dynamics subject to (5.1) and the initial condition $x(0) \in \mathcal{X}^0 = \mathfrak{s}(0) \times \mathfrak{v}(0)$ where $\mathfrak{s}(0) = [\underline{s}(0), \overline{s}(0)]$ and $\mathfrak{v}(0) = [\underline{v}(0), \overline{v}(0)]$ are the position and velocity intervals. The reachable, two-dimensional interval $\mathcal{X}(t) = [\underline{x}(t), \overline{x}(t)]$ of position and velocity is given by

$$\begin{aligned} \underline{x}(t) &= \underline{x}(0) + \int_0^t f^{TP}(x(\tau), u(\tau))\, d\tau, \quad u(\tau) = -1 \\ \overline{x}(t) &= \overline{x}(0) + \int_0^t f^{TP}(x(\tau), u(\tau))\, d\tau, \quad u(\tau) = 1. \end{aligned} \qquad \square$$

The proof is omitted as it is obvious that the maximum position and velocity is obtained when the vehicle starts with the maximum initial position and velocity under full acceleration. The analogous argumentation holds for the lowest position and velocity. Note that this argumentation is only applicable if there exists an initial state that jointly contains the maximum initial position and velocity. This is always the case when the initial set is a two-dimensional interval, which is in contrast to the example of Fig. 5.7(a), for which Prop. 5.1 is not applicable. In this example, the maximum reachable position at different times is reached from trajectories starting from different initial states. However, if one is only interested in the reachable position – and the initial set is a two-dimensional interval, as shown in the example of Fig. 5.7(b), the result of Prop. 5.1 results in the exact reachable interval of the position coordinate.

If the path is curved, one has to consider the tire friction constraint in (5.2). For a given curvature profile $\rho(s)$, the minimum and maximum admissible input $\underline{u}(s)$ and $\overline{u}(s)$ can be obtained as presented e.g. in [167]. By exchanging $u(\tau) = -1$ with $u(\tau) = \underline{u}(s(\tau))$ and $u(\tau) = 1$ with $u(\tau) = \overline{u}(s(\tau))$ in Prop. 5.1, one can compute the reachable positions for a curved road. Speed limits on a road can be considered by cutting off the previously computed speed profile $v(s)$ at $v^{\max}$ by assigning $\overline{u}(s) = 0$ if $v(s) > v^{\max}$. Note that for a given input u (which is constant for a time step of the simulation), the position and velocity can be computed analytically:

Proposition 5.2 (Analytical Solution of the Longitudinal Dynamics): The analytical solution of the longitudinal dynamics of traffic participants in (5.1) for $u > 0$



(a) Initial values within a zonotope.

(b) Initial values within a two-dimensional interval.

Fig. 5.7.: Reachable sets of a vehicle for different initial sets. The vehicle model according to (5.1) with $a^{\max} = 7$ [m/s²], $v^{\text{sw}} = 7.3$ [m/s] is applied.

and $v > v^{\text{sw}}$ is

$$\begin{aligned} s(t) &= s(0) + \frac{(v(0)^2 + 2v^{\text{sw}}\, u\, t)^{\frac{3}{2}} - v(0)^3}{3v^{\text{sw}}\, u}, \\ v(t) &= \sqrt{v(0)^2 + 2v^{\text{sw}}\, u\, t}. \end{aligned}$$

The correctness can be easily verified by inserting the solution into (5.1). The analytical solution of the cases $0 < v < v^{\text{sw}} \vee u < 0$ and $v \leq 0$ is trivial. □

In order to obtain the two-dimensional reachable positions on the road and not only the reachable positions along a path from the previous computation, it is generally assumed that the vehicle can laterally cover the whole lane if not stated differently. Another simple solution of reachable positions exists for vehicles with bounded absolute acceleration $a^{\max}$ on a two-dimensional plane [149].

The previously presented deterministic computations are supported by the Markov chains abstracting the original dynamics. Their generation is summarized in the following.

5.4.2. Offline Computations

Since the Markov chain does not have to be complete, its transition probabilities are computed by Monte Carlo simulation as shown in Sec. 4.3.2. In order to represent the movement of other traffic participants for the whole prediction horizon, the discretization region $\mathcal{X} = s \times v$ has to be properly chosen for the Markov chain abstraction. The maximum necessary region is as follows: The velocity interval ranges from standstill to the maximum considered speed $v = [0, v^{\max*}]$ and the position interval is $[0, v^{\max*} \cdot t_f + s^{\text{detect}}]$, where t_f

is the fixed or maximal prediction horizon and s^{detect} the distance from which other traffic participants can be detected. However, due to efficiency reasons, smaller discretization regions $\mathcal{X}$ can be reasonable, too.

Based on the discretization, the transition matrices of the state are computed for points in time and time intervals as presented in Sec. 4.3.2. Besides different transition matrices for points in time and time intervals, transition matrices are also diversified by different choices of parameters in (5.1) which are listed in Tab. 5.1. The various transition matrices for different types of traffic participants are stored and loaded during the online procedure, which is addressed next.

5.4.3. Online Computations

During the online operation, a Markov chain for each detected traffic participant is instantiated whose state transition probability matrices $\tilde{\Phi}(\tau)$, $\tilde{\Phi}([0,\tau])$ are loaded from a database and τ is the time increment of Markov chains, see Sec. 4.3.2. For example, in a traffic scene with 2 cars and 1 bicycle, 2 Markov chains for the cars and 1 Markov chain for the bicycle are instantiated. The initial probability distributions $\tilde{p}(0)$ are generated according to the measurement uncertainties. Then, the Markov chain of each traffic participant is updated for the prediction horizon t_f according to (4.25) which is recapitulated for better readability:

$$\begin{aligned}\tilde{p}(t_{k+1}) &= \tilde{\Gamma}(t_k)\,\tilde{\Phi}(\tau)\,\tilde{p}(t_k),\\ \tilde{p}([t_k,t_{k+1}]) &= \tilde{\Phi}([0,\tau])\,\tilde{p}(t_k),\end{aligned}$$

where $\tilde{\Gamma}(t_k)$ are the time varying input transition matrices which are generated from behavior models to be introduced in the next section. The times t_k are a short notation for $t_k = k \cdot \tau$.

Since the probabilities do not have to be computed in an over-approximative fashion as previously explained, one can cancel small probabilities and normalize the probability vector afterwards such that its sum is one. This procedure has the advantage that the computational time is reduced because the transition matrix and the probability vector are stored as a sparse matrix/vector which neglects zero entries. Special algorithms for sparse matrix multiplications allow a drastic increase in the efficiency of matrix multiplications, which are the faster the more zero entries exist (see e.g. [175]).

Heuristic 5.1 (Cancelation of Small Probabilities): Probabilities in $\tilde{p}$ which have a value of less than $\underline{p}$ are replaced by zeros so that one obtains $\tilde{p}^*$, which is normalized to the new probability vector $\tilde{p}_l = \tilde{p}_l^* / \sum_l \tilde{p}_l^*$. The value of $\underline{p}$ should be chosen in relation to the number of combined input and state cells $d \cdot c$ such that

$$\underline{p} = \frac{\varpi}{d \cdot c} \tag{5.3}$$

and ϖ can be freely chosen. □

5.5. Behavior Modeling

The dynamics of traffic participants has been modeled based on physical considerations in (5.1) and abstracted by Markov chains as described in the previous section. The Markov chains allow the computation of the probability distributions of other traffic participants when their sequence of input transition probabilities $\tilde{\Gamma}(t_k)$ is known, where the input values refer to the acceleration command. This sequence is also referred to as the behavior of traffic participants from now on. Besides the acceleration command, high-level decision probabilities such as the probability of taking a left turn or the probability of changing lane is also taken into account for the prediction of traffic participants. This is done by weighting possible paths of traffic participants according to high-level decision probabilities. In general, these probabilities are provided by other prediction algorithms, which often work with alternative methods such as Bayesian networks or neural networks. However, the computation of the high-level probability for changing lane is later addressed using the methods at hand. Besides this exception, only the generation of input transition matrices $\tilde{\Gamma}(t_k)$ is discussed below, because the high-level decision probabilities are provided by other software modules within the prototype vehicle [164].

Clearly, the input transition matrices $\tilde{\Gamma}(t_k)$ cannot be derived the same way as the state transition matrices $\tilde{\Phi}(\tau)$ and $\tilde{\Phi}([0,\tau])$, because the acceleration commands of other drivers cannot be described by differential equations. There are two main approaches for creating the input transition probability matrices $\tilde{\Gamma}(t_k)$. One possibility is to generate the transition probabilities based on heuristics. The other possibility is to learn the behavior of traffic participants based on traffic observations. In most works, those traffic observations are realized with static cameras (fixed position) and computer vision for the recognition of traffic participants in the camera image. Another possibility is to observe the traffic from a camera installed in a moving vehicle. Both configurations allow the recording of the trajectories of other traffic participants for the learning algorithms.

These trajectories can be clustered, resulting in motion primitives such as *left turn*, *lane change*, or *parking* [86]. A similar work focuses more on the probability distribution of trajectories within a cluster [88]. Those motion primitives could complement the paths of the traffic participants which are automatically generated from the road geometry and the finite set of decisions {*left turn*, *right turn*, *go straight*}, and {*left lane change*, *right lane change*} as introduced in Sec. 5.3. The advantage of this automatic clustering is that it covers more typical behaviors, so that one does not have to use the prediction algorithms for unstructured environments as a fallback solution so often.

The other objective for the recording of trajectories is to learn the behavior of traffic participants when following certain motion primitives. Alternatively, one can also learn behaviors directly from their two-dimensional movements on the road without grouping them into motion primitives. However, it is believed that the two-step approach of firstly learning motion patterns and secondly learning the dynamics along these motion patterns is more promising. In literature, various models are investigated in order to learn the behavior of traffic participants: Hidden Markov Models [123, 159], growing Hidden Markov Models [166], and switched ARX models [151].

Due to the lack of recorded trajectories of other traffic participants in various real world traffic situations, learning algorithms have not been applied, and heuristics are used instead. The recording of the required trajectories by the prototype vehicles of the *Cognitive Automobiles* research project is part of future work, though. Unavailable trajectories of other traffic participants are also the reason why the applied heuristics have not yet been compared with real traffic data. The heuristic approach presented afterwards adapts the input probabilities (acceleration commands) based on the geometry of the road and the interaction with other traffic participants in *lane following*, *intersection crossing*, and *lane changing* situations. First, general properties of the input dynamics are introduced.

5.5.1. General Computation

The input transition values $\Gamma_i^{\alpha\beta}(t_k)$ are generated below, where the index i refers to the state and α, β are the final and initial value of the transition, respectively. For better readability, the update of the conditional input probabilities $q_i^{\beta}(t_k)$ according to the input transition values $\Gamma_i^{\alpha\beta}(t_k)$ is recalled from (4.22):

$$q_i^{\alpha}(t_k)' = \sum_{\beta} \Gamma_i^{\alpha\beta}(t_k) \cdot q_i^{\beta}(t_k).$$

The input transition probabilities $\Gamma_i^{\alpha\beta}$ are composed of two components. One component is a transition matrix Ψ which models the intrinsic behavior of the traffic participant when there are no priorities for certain input values. Priorities arise when e.g. a traffic participant is forced to brake due to a curve or a slower vehicle. Those priorities are modeled by a priority variable λ, which is the second component. The intrinsic transition matrix Ψ is introduced first and later combined with the priority λ.

The effect of the intrinsic transition matrix Ψ is discussed under the assumption that there are no priorities, such that $\Gamma_i^{\alpha\beta}(t_k) = \Psi^{\alpha\beta}$. Note that due to the intrinsic nature, $\Psi^{\alpha\beta}$ is independent of the time step t_k and the state value i. In order to generate a proper transition matrix Ψ, the normalization operator norm() is introduced first:

$$\Psi^{\alpha\beta} = \texttt{norm}(\hat{\Psi}^{\alpha\beta}) := \frac{\hat{\Psi}^{\alpha\beta}}{\sum_{\alpha} \hat{\Psi}^{\alpha\beta}}.$$

The column sums of the resulting transition matrix Ψ are 1 in order to ensure that the multiplication with a probability vector results in a probability vector whose sum is 1. The transition probabilities of Ψ are set according to the heuristics that the bigger the change of the input[3], the more unlikely this change is. A transition matrix that considers this aspect is

$$\Psi^{\alpha\beta}(\gamma) = \texttt{norm}\big(\hat{\Psi}^{\alpha\beta}(\gamma)\big), \quad \hat{\Psi}^{\alpha\beta}(\gamma) = \frac{1}{(\alpha - \beta)^2 + \gamma}.$$

The parameter γ allows the gradual interpolation of the extreme cases $\lim_{\gamma\to 0} \Psi(\gamma) = I$ and $\lim_{\gamma\to\infty} \Psi(\gamma) = \frac{1}{c}\mathbf{1}$, where $\mathbf{1}$ is a matrix of ones and c is the number of inputs. Informally speaking, a low value of γ models drivers that rarely change their acceleration command, whereas a high value models drivers that often change their acceleration command. The higher the value of γ, the faster the initial input distribution converges to the steady state

[3] As the discrete inputs are numbered in increasing order according to the acceleration intervals, the difference between the input numbers is a measure of the change of the acceleration interval.

distribution, which is uniform over all inputs. This is illustrated in Fig. 5.8 for 3 inputs. High input numbers represent high positive acceleration, such that the first input $\mathbf{y} = 1$ represents full braking and the last input $\mathbf{y} = 3$ full acceleration. The initial probabilities are set to $P(\mathbf{y} = 1) = 0$, $P(\mathbf{y} = 2) = 0.8$, $P(\mathbf{y} = 3) = 0.2$ and the probabilities converge to $\frac{1}{3}$ as no prioritization is specified.

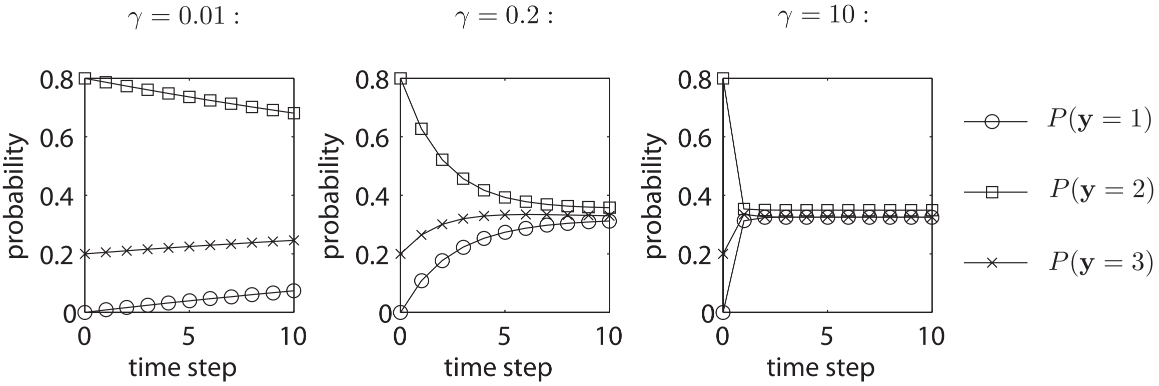


Fig. 5.8.: Input evolution for $\gamma = 0.01, 0.2, 10$.

Complementing the intrinsic transition matrix Ψ with the priority values of λ results in the input transition values $\Gamma_i^{\alpha\beta}$.

$$\begin{aligned} &\Gamma_i^{\alpha\beta} = \mathtt{norm}(\hat{\Gamma}_i^{\alpha\beta}), \\ &\hat{\Gamma}_i^{\alpha\beta} = \lambda_i^\alpha \cdot \Psi^{\alpha\beta}, \quad \forall i : \sum_\alpha \lambda_i^\alpha = 1,\, 0 \leq \lambda_i^\alpha \leq 1, \end{aligned} \tag{5.4}$$

where the state dependence of $\Gamma_i^{\alpha\beta}$ is solely modeled by the priority values λ_i^α, while the input dynamics matrix $\Psi^{\alpha\beta}$ is independent of the state. The above formula has the following special cases and properties:

- $\lambda_i^\alpha = 0$: Regardless of the intrinsic transition matrix Ψ, the input α of state i is prohibited ($q_i^\alpha = 0$).
- $\lambda_i^\alpha = \frac{1}{c}, \forall i, \alpha$ (c is the number of inputs): No input is prioritized, such that $\Gamma_i^{\alpha\beta} = \Psi^{\alpha\beta}$.
- $\Psi = I$ (I is the identity matrix): $\Gamma_i^{\alpha\beta} = I^{\alpha\beta}$, regardless of λ, such that the input probability is unchanged.
- $\Psi = \frac{1}{c}\mathbf{1}$ ($\mathbf{1}$ is a matrix of ones): The multiplication $\sum_\beta \Gamma_i^{\alpha\beta} \cdot q_i^\beta$ results in λ_i^α. Thus, a certain input probability distribution $q_i^{\alpha\prime} = \lambda_i^\alpha$ is enforced, regardless of the probability distribution of the previous time step.

Unfortunately, a formalism that generates values $\Gamma_i^{\alpha\beta}$ with the above-listed properties and additionally ensures the steady state solution $q_i^\alpha(t_\infty) = \lambda_i^\alpha$ has not been found. This solution would have the advantage that in the long run, drivers would always change their measured acceleration distribution to the one enforced by the priority values. The problem of creating a transition matrix which results in a certain steady state solution has been addressed in [117, 118]. However, there exist only solutions for so-called class $\mathcal{C}$ matrices and Γ does not belong to this class. Nevertheless, the steady state solution has values that are at least similar to the priority values of λ and approximate the priority values better for higher γ values, so that for $\gamma \to \infty$: $q_i^\alpha(t_\infty) = \lambda_i^\alpha$. In a previous work of the author [186], the intrinsic transition matrix Ψ is not introduced, so that the input probability is $q_i^{\alpha\prime} = \lambda_i^\alpha$, which is equivalent to $\gamma \to \infty$.

It remains to compute the priority values $\lambda_i^\alpha(t_k)$ for different traffic situations. One aim is to alter λ so that constraints given by other traffic participants or the road geometry are met. The other considered effect for the priority values is that traffic participants have preferences for certain input values which are stored in the state independent motivation values μ^α ($\forall i : \mu_i^\alpha = \mu^\alpha$). The motivation values are equivalent to the priority values in the absence of constraints ($\lambda_i^\alpha = \mu_i^\alpha$).

In order to consider constraints, the event C of constraint satisfaction is introduced. Due to the uncertain modeling of traffic participants, constraint satisfaction is not guaranteed, such that the conditional probability of constraint satisfaction is introduced as $\eta_i^\alpha := P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha)$, where $\mathbf{z}$ and $\mathbf{y}$ is the random state and input of the Markov chain, respectively. Since constraints have to be met, the probability values η_i^α serve as an upper bound of the motivation values μ_i^α so that the priority values become

$$\lambda_i^\alpha = \begin{cases} \mu_i^\alpha, \text{ if } \mu_i^\alpha \le \eta_i^\alpha \\ \eta_i^\alpha, \text{ otherwise.} \ \to \mu_i^{\alpha-1} := \mu_i^{\alpha-1} + \mu_i^\alpha - \eta_i^\alpha \end{cases}$$

In other words, μ_i^α is cut off at η_i^α and the cut-off probability is added to the next lower acceleration interval: $\mu_i^{\alpha-1} := \mu_i^{\alpha-1} + \mu_i^\alpha - \eta_i^\alpha$, which is also visualized in Fig. 5.9. The redistribution to lower acceleration intervals implies that a constraint is satisfied by lowering the acceleration. However, there are also situations such as overtaking maneuvers in which higher acceleration allows constraints to be satisfied. The integration of those maneuvers is the subject of future work.

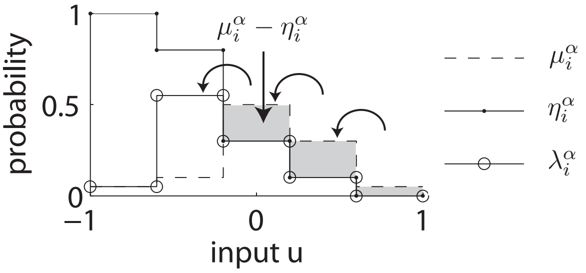


Fig. 5.9.: Combining a motivation distribution μ with a constraint distribution η.

The computation of constraint values for the capabilities *road following*, *vehicle following*, *intersection crossing* and *lane changing* is addressed in the next subsections.

5.5.2. Road Following

In this subsection, the computation of the constraint values ${\eta^{\mathrm{road}}}_i^\alpha$ with respect to acceleration limits and speed limits along curved paths is presented. For this, the possible velocity interval $[\underline{v}(s), \overline{v}(s)]$ resulting from the minimum and maximum possible acceleration in (5.2) is introduced. An additional labeling with brackets indicates the maximum absolute

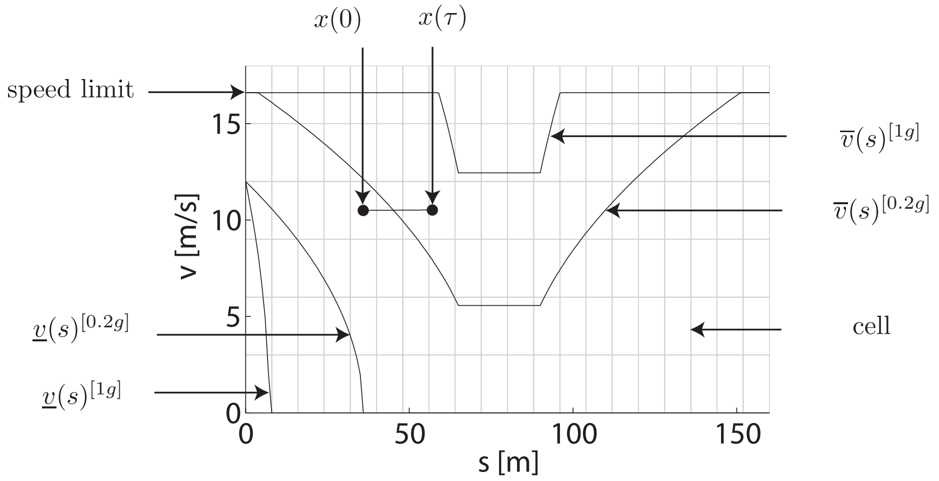


Fig. 5.10.: Velocity profiles for two straights connected by a 90° curve (15.5 m radius) and a speed limit of $v^{\max} = 16.7$ m/s $\hat{=}60$ km/h.

acceleration, e.g. $\overline{v}(s)^{[0.5g]}$ is the fastest possible velocity profile for $a^{\max} = 0.5g$ and g is the gravitational constant. Exemplary velocity profiles with a speed limit (possibly greater than the official speed limit to account for sporty drivers) are shown in Fig. 5.10.

The maximum acceleration constraint is violated when the velocity is outside the velocity profile bounds. However, this constraint may also be violated within the velocity bounds, when e.g. strongly accelerating within a curve while staying below $\overline{v}(s)$. Nevertheless, the event of constraint satisfaction C is defined such that it is true when the velocity is within the velocity profile bounds ($\underline{v}(s) \leq v \leq \overline{v}(s)$). This simplification is necessary for an efficient implementation and yields reasonable results within this framework, as shown later.

The compliance with the acceleration and the maximum velocity constraint for a state $\mathbf{z} = i$ and an input $\mathbf{y} = \alpha$ is approximately checked by a single simulation run[4]:

1. Simulate the vehicle for the time τ, starting from $x(0) = \texttt{center}(\mathcal{X}_i)$ under the effect of $u = \texttt{center}(\mathcal{U}^\alpha)$. The operator $\texttt{center}()$ returns the volumetric center of a set. Remember that $\mathcal{X}_i$ is the set of continuous states represented by the discrete state $\mathbf{z} = i$ and $\mathcal{U}^\alpha$ is the set of continuous inputs represented by the discrete input $\mathbf{y} = \alpha$.
2. Check whether the velocity is within the minimum and maximum velocity profile after one time increment τ (see also Fig. 5.10):

$$\underline{v}(s(\tau))^{[\bar{a}_d]} \leq v(\tau) \leq \overline{v}(s(\tau))^{[\bar{a}_d]}. \tag{5.5}$$

The constraint values are then obtained from the simulation results as

$$P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha, \mathbf{a} < \bar{a}_d) = \begin{cases} 1, & \text{if (5.5) holds} \\ 0, & \text{otherwise} \end{cases} .$$

Next, the probability distribution for applied accelerations $P(\mathbf{a} < \bar{a}_d)$ among all drivers is

[4] The lack of an over-approximative computation is appropriate since the reachable positions are computed separately in Prop. 5.1.

Tab. 5.2.: Discretization of the state and input space.

position interval s	$[0, 200]$ m
velocity interval v	$[0, 20]$ m/s
input interval $\mathcal{U}$	$[-1, 1]$
position segments	40
velocity segments	10
input segments	6
time increment τ	0.5 s

Tab. 5.3.: Behavior parameters.

γ	0.2
μ	$\begin{bmatrix}0.01 & 0.04 & 0.1 & 0.4 & 0.4 & 0.05\end{bmatrix}$
$q_i(0)$	$\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}$ $(\forall i)$
$v^{\max}$	16 m/s

Tab. 5.4.: Initial state: Set with uniform distribution.

$s(0) \in$	$[2, 8]$ m
$v(0) \in$	$[12, 14]$ m/s

introduced, where $0 < \bar{a}_d \leq a^{\max}$ and $a^{\max}$ is the physically possible acceleration. The index d refers to a value within a finite set of selected absolute accelerations, e.g. $\{\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_6\}$. Drivers that prefer a more comfortable ride have high probabilities for low values of $\bar{a}_d$ and the other way round for sporty drivers. The constraint values over all accelerations $\bar{a}_d$ are

$$\eta_i^\alpha = P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha) = \sum_d P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha, \mathbf{a} < \bar{a}_d) P(\mathbf{a} < \bar{a}_d).$$

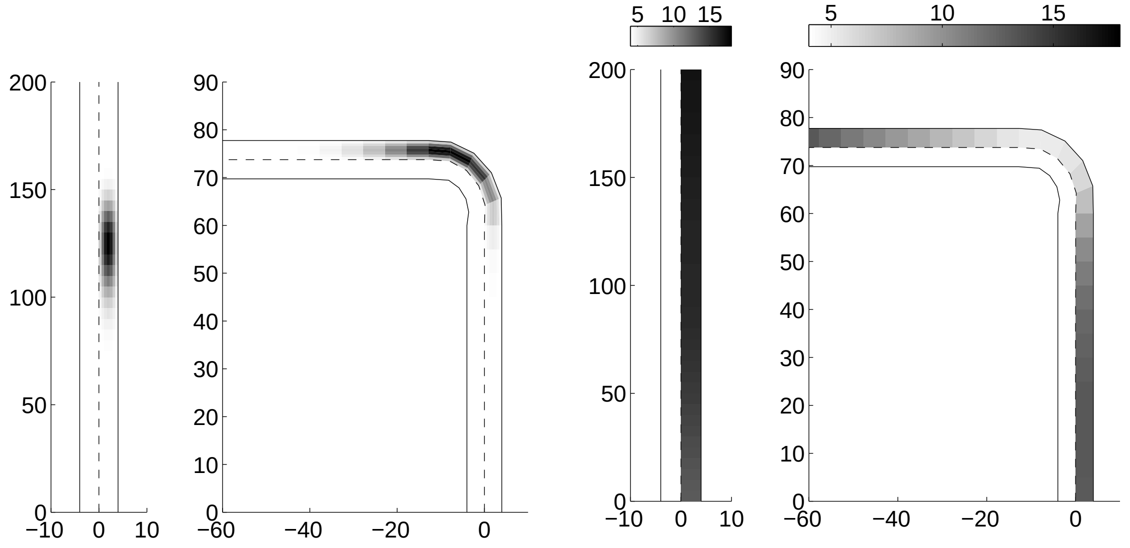
It is remarked that the probability of maximum applied acceleration $P(\mathbf{a} < \bar{a}_d)$ is not obtained online by observation of other drivers, but set according to an average distribution of all drivers. The effect of the constraint values η_i^α on road following is demonstrated for a vehicle that drives on a straight road and a curved road with the same initial states.

Example 5.1 (Straight versus Curved Road): The stochastic reachable sets of a vehicle is computed with identical initial states when following a straight and a curved road. The Markov chain for the vehicle is obtained from a discretization specified in Tab. 5.2. The parameters determining the behavior of the vehicle are shown in Tab. 5.3 and the initial state of the vehicle is listed in Tab. 5.4.

The probability distributions on the straight and curved road can be seen in Fig. 5.11(a) for $t \in [7.5, 8]$ s. The average velocity of the vehicle on different road segments in shown in Fig. 5.11(b), in which the color bar maps the gray tone to the average velocity. This figure nicely illustrates that the vehicle considers the speed limit of 16 m/s while decelerating in front of a curve and accelerating after it. The distributions of the longitudinal position, the velocity, and the input for the time interval $t \in [7.5, 8]$ s are plotted in Fig. 5.12. There, it can be observed that due to the curve, the vehicle on the straight road has traveled further.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.05 s for each scenario with a prediction horizon $t_f = 10$ s and cancelation of probabilities below $p^{\max} = 10/(d \cdot c) = 4.2e - 3$ (see Heuristic 5.1). □

The effect on the constraint values when following another vehicle is shown next.



(a) Probability distribution for $t \in [7.5, 8]$ s. The coordinate axes refer to positions in [m].

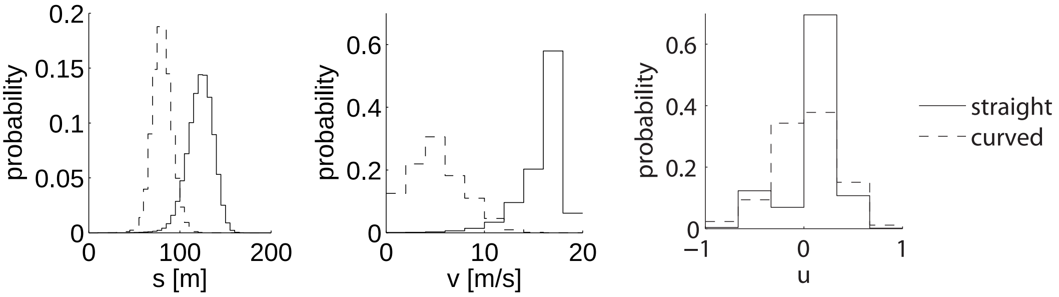
(b) Average velocity indicated by the color bar in [m/s] for $t \in [0, 10]$ s. The coordinate axes refer to positions in [m].

Fig. 5.11.: Probability distribution and average velocity on a straight and curved road.

5.5.3. Vehicle Following

Constraints not only arise from curved paths to be followed and speed limits to be respected, but also from other traffic participants. In this subsection, traffic participants following other traffic participants on the same lane are considered. The more complex cases of interaction at intersections and during lane changes are considered later.

Analogously to road following, the constraint values for vehicle following are computed based on simulations and heuristics. In order to distinguish the variables of the following vehicle from the ones of the leading vehicle, the variables of the following vehicle are denoted by a raised F (e.g. $\mathbf{z}^F$), and the variables for the leading vehicle by a raised L (e.g. $\mathbf{z}^L$). The constraint for the following vehicle is that its behavior should not cause



(a) Position histogram. (b) Velocity histogram. (c) Input histogram.

Fig. 5.12.: Histograms of position, velocity and input for $t \in [7.5, 8]$ s.

a crash. This is approximately checked by a single simulation[5], similarly as for the road following scenario:

1. Simulate both vehicles for the time interval $[0, \nu \cdot \tau]$, with constant $\nu \in \mathbb{N}^+$, starting from $x^F(0) = \texttt{center}(\mathcal{X}_i)$, $x^L(0) = \texttt{center}(\mathcal{X}_j)$ under the effect of $u^F = \texttt{center}(\mathcal{U}^\alpha)$, $u^L = \texttt{center}(\mathcal{U}^\beta)$.
2. Simulate a sudden and full brake beginning at $t = \nu \cdot \tau$ of the leading vehicle, to which the following vehicle reacts with a full brake, too. This behavior is simulated until the following vehicle has stopped at $t = t_S$.
3. Check if the following vehicle has crashed into the leading vehicle for $t \in [0, t_S]$.

The outcome of the simulation determines the conditional probability for satisfying the constraint of crashing with a probability of less than ϵ which is motivated by driver inattentiveness.

$$P(\mathrm{C}|\mathbf{z}^F = i, \mathbf{z}^L = j, \mathbf{y}^F = \alpha, \mathbf{y}^L = \beta, \Delta\mathbf{t} = \nu \cdot \tau) = \begin{cases} 1, & \text{no crash simulated} \\ \epsilon, & \text{otherwise,} \end{cases}$$

where $\Delta\mathbf{t}$ is the discrete random variable for the time interval with constant acceleration. Long time intervals model the behavior of foresighted drivers who adjust their acceleration early to changes of other drivers, while short time intervals represent sporty drivers. The probabilities $P(\Delta\mathbf{t} = \nu \cdot \tau)$ for time intervals in which the acceleration interval is unchanged, allows the computation of

$$P(\mathrm{C}|\underbrace{\mathbf{z}^F = i, \mathbf{z}^L = j, \mathbf{y}^F = \alpha, \mathbf{y}^L = \beta}_{D}) = \sum_{\nu} P(\mathrm{C}|D, \Delta\mathbf{t} = \nu \cdot \tau) \cdot P(\Delta\mathbf{t} = \nu \cdot \tau).$$

Note that the probability $P(\Delta\mathbf{t} = \nu \cdot \tau)$ is not obtained online by observation of other drivers, but set according to an average distribution of all drivers. The conditional probabilities $P(\mathrm{C}|D)$ are computed offline and stored according to the state and input indices in $\Theta_{ij}^{\alpha\beta}$ which allows the constraint values to be obtained for the vehicle following:

Proposition 5.3 (Constraint Vector for Vehicle Following): Under the assumption that $P(\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ and $P(\mathbf{z}^L = j, \mathbf{y}^L = \beta)$ are independent, one obtains

$$\eta_i^\alpha = \sum_{j,\beta} \Theta_{ij}^{\alpha\beta} p_j^{L\beta}. \qquad \square$$

Proof: After defining the events $\tilde{A} = (\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ and $\tilde{B}_j^\beta = (\mathbf{z}^L = j, \mathbf{y}^L = \beta)$, one can write:

$$\begin{aligned} P(\mathrm{C}, \tilde{A}) &= \sum_{j,\beta} P(\mathrm{C}|\tilde{A}, \tilde{B}_j^\beta) P(\tilde{A}, \tilde{B}_j^\beta) \\ &\overset{\text{independence}}{=} \sum_{j,\beta} P(\mathrm{C}|\tilde{A}, \tilde{B}_j^\beta) P(\tilde{A}) P(\tilde{B}_j^\beta) \end{aligned}$$

[5] The lack of an over-approximative computation is appropriate since the reachable positions are computed separately in Prop. 5.1.

$$\to \eta_i^\alpha = P(\mathrm{C}|\tilde{A}) = \frac{P(\mathrm{C},\tilde{A})}{P(\tilde{A})} = \sum_{j,\beta} \underbrace{P(\mathrm{C}|\tilde{A},\tilde{B}_j^\beta)}_{\Theta_{ij}^{\alpha\beta}} \underbrace{P(\tilde{B}_j^\beta)}_{p_j^{L\beta}} . \qquad \square$$

It is obvious that the independence assumption only approximates the joint probability $P(\tilde{A},\tilde{B}_j^\beta) \approx P(\tilde{A})P(\tilde{B}_j^\beta)$. Another issue is that the constraint value of the following vehicle for a single state i and input α is computed based on the complete probability distribution of the leading vehicle. The summation over all states j and inputs β of the leading vehicle in Prop. 5.3 leads to an averaging effect. The error caused by the averaging effect and the independence assumption are later evaluated in Sec. 5.7.6 when the Monte Carlo approach is compared to the Markov-chain approach.

In traffic situations with more than two vehicles on a lane, the simplification is made that each vehicle only reacts to the next vehicle driving in front and not to other vehicles.

The notation in Prop. 5.3 can be further simplified when using the probability vector $\tilde{p}$ as introduced in (4.23) containing all values p_j^β. This rewriting can be analogously applied to the constraint values of η:

$$\begin{aligned} \tilde{p}^T &= \begin{bmatrix} p_1^1 & p_1^2 & \dots & p_1^c & p_2^1 & p_2^2 & \dots & p_2^c & p_3^1 & \dots & p_d^c \end{bmatrix} \\ \tilde{\eta}^T &= \begin{bmatrix} \eta_1^1 & \eta_1^2 & \dots & \eta_1^c & \eta_2^1 & \eta_2^2 & \dots & \eta_2^c & \eta_3^1 & \dots & \eta_d^c \end{bmatrix} . \end{aligned}$$

After writing the interaction values $\Theta_{ij}^{\alpha\beta}$ into the interaction matrix

$$\tilde{\Theta} = \begin{bmatrix} \Theta_{11}^{11} & \Theta_{11}^{12} & \dots & \Theta_{11}^{1c} & \Theta_{12}^{11} & \Theta_{12}^{12} & \dots & \Theta_{12}^{1c} & \Theta_{13}^{11} & \dots & \Theta_{1d}^{1c} \\ \Theta_{11}^{21} & \Theta_{11}^{22} & \dots & \Theta_{11}^{2c} & \Theta_{12}^{21} & \Theta_{12}^{22} & \dots & \Theta_{12}^{2c} & \Theta_{13}^{21} & \dots & \Theta_{1d}^{2c} \\ \vdots & & & & & & & & & & \vdots \\ \Theta_{d1}^{c1} & \Theta_{d1}^{c2} & \dots & \Theta_{d1}^{cc} & \Theta_{d2}^{c1} & \Theta_{d2}^{c2} & \dots & \Theta_{d2}^{cc} & \Theta_{d3}^{c1} & \dots & \Theta_{dd}^{cc} \end{bmatrix} ,$$

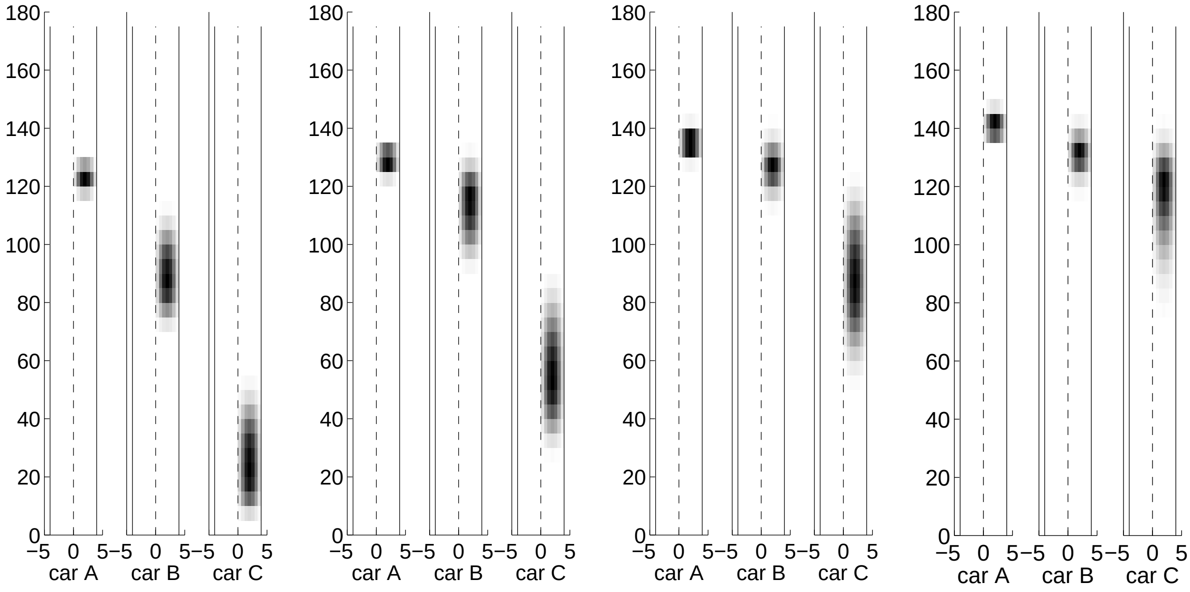
one can rewrite the computation of the constraint vector $\tilde{\eta}$ as $\tilde{\eta} = \tilde{\Theta}\, \tilde{p}^L$.

Note that the above equation is the only equation that has to be computed online in order to obtain $\tilde{\eta}$, since $\tilde{\Theta}$ is computed offline. When more than one constraint vector is active, e.g. the constraint vector for road following and vehicle following, the combined constraint vector is obtained by choosing the minimum constraint values elementwise ($\tilde{\eta} = \mathtt{min}(\tilde{\eta}^{\text{road}}, \tilde{\eta}^{\text{vehicle}})$) so that the constraints are simultaneously fulfilled in a probabilistic sense. The effect of the constraint vector is shown in the next example in which three cars drive in the same lane.

Example 5.2 (Vehicle Following): The interaction between vehicles driving in a lane is exemplarily shown for 3 cars driving one after the other. The cars are denoted by the capital letters A, B, and C, where A is the first and C the last vehicle in driving direction. Analogously to the previous example on road following, the Markov chain used for the vehicles B and C is obtained from the discretization in Tab. 5.2. Vehicle A is not computed based on a Markov chain, but predicted with a constant velocity of 3 m/s so that the faster vehicles B and C are forced to brake. The parameters determining the behavior of the vehicles B and C are as for the previous example in Tab. 5.3 and the initial state distributions of the vehicles $A - C$ are specified in Tab. 5.5.

The probability distributions for consecutive time intervals are plotted in Fig. 5.13. For visualization reasons, the position distributions are plotted in separate plots, although the vehicles drive in the same lane. Dark regions indicate high probability, while bright regions represent areas of low probability. In order to improve the visualization, the colors are separately normalized for each vehicle. The average velocity of the vehicles along the lane is shown in Fig. 5.14. The distributions of the position, velocity, and acceleration command for different vehicles and time intervals is shown in Fig. 5.15-5.17.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.21 s when canceling probabilities below $p^{\max} = 10/(d\cdot c) = 4.2e{-}3$ (see Heuristic 5.1) for a prediction horizon of $t_f = 8$ s. □



(a) $t \in [0, 2]$ s.

(b) $t \in [2, 4]$ s.

(c) $t \in [4, 6]$ s.

(d) $t \in [6, 8]$ s.

Fig. 5.13.: Position distribution for different time intervals. The coordinate axes refer to positions in [m].

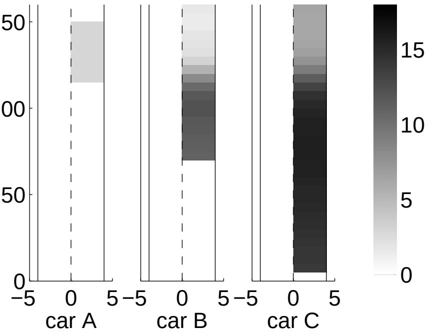
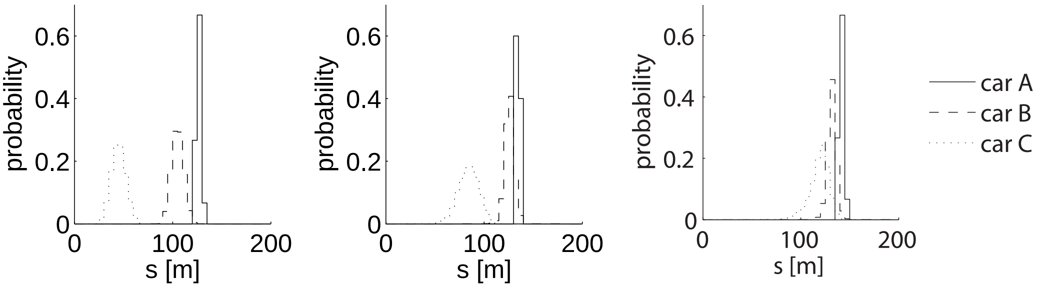


Fig. 5.14.: Average velocity indicated by the color bar in [m/s] for $t \in [0, 8]$ s. The coordinate axes refer to positions in [m].

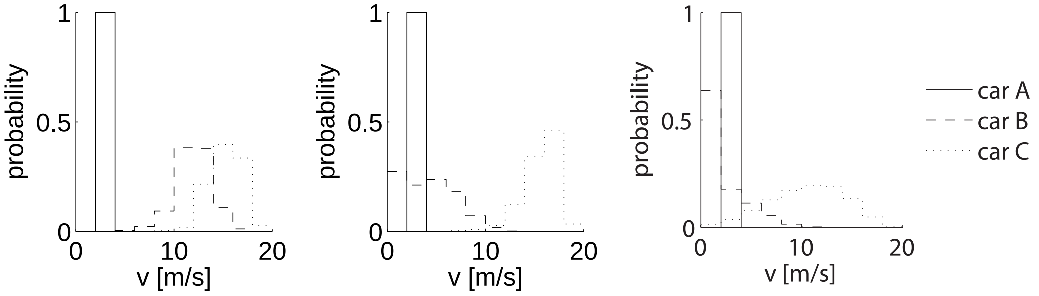
Tab. 5.5.: Initial state: Set with uniform distribution.

vehicle A	$s^A(0) \in [117, 123]$ m $v^A(0) \in [2, 4]$ m/s
vehicle B	$s^B(0) \in [72, 84]$ m $v^B(0) \in [10, 12]$ m/s
vehicle C	$s^C(0) \in [5, 17]$ m $v^C(0) \in [13, 15]$ m/s



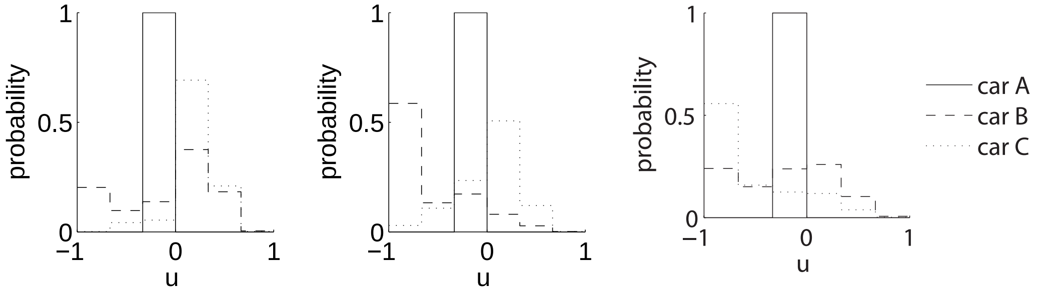
(a) $t \in [2, 2.5]$ s.
(b) $t \in [4.5, 5]$ s.
(c) $t \in [7, 7.5]$ s.

Fig. 5.15.: Position histograms for different time intervals.



(a) $t \in [2, 2.5]$ s.
(b) $t \in [4.5, 5]$ s.
(c) $t \in [7, 7.5]$ s.

Fig. 5.16.: Velocity histograms for different time intervals.



(a) $t \in [2, 2.5]$ s.
(b) $t \in [4.5, 5]$ s.
(c) $t \in [7, 7.5]$ s.

Fig. 5.17.: Input histograms for different time intervals.

The concept of vehicle following can be extended to scenarios at intersections with a few additional computations.

5.5.4. Intersection Crossing

The interaction results for two vehicles driving on a lane are extended to intersection scenarios. The behavior is differently computed for vehicles that have right of way and those that have not. The constraint vector of traffic participants having right of way are computed as previously described for road following and vehicle following, resulting in the combined constraint vector $\tilde{\eta} = \mathtt{min}(\tilde{\eta}^{\text{road}}, \tilde{\eta}^{\text{vehicle}})$. The constraint vector for vehicles that do not have right of way are computed in two phases (modes): *approaching* and *crossing*. First, the interaction at intersections is discussed for a simple example with two vehicles R (right of way) and N (no right of way) as depicted in Fig. 5.18, and extended later. Note that for each vehicle only one path through the intersection is considered for simplicity. This means no loss of generality, as further combinations of driving paths are computed the same way.

In *approaching* mode, a virtual vehicle V with zero velocity is put in front of the approaching vehicle where the path with right of way is crossing; see Fig. 5.18. This point is also referred to as the crossing position $s^{\text{cross},N}$ of vehicle N. The approaching vehicle N interacts with the virtual vehicle as previously described in Sec. 5.5.3 so that approaching a crossing is emulated by approaching a standing vehicle. As soon as the approaching vehicle N has entered the transition region to crossing mode (see Fig. 5.18), the mode *crossing* may become active. The length of the crossing region can be set by default to e.g. 10 m or be adapted to the intersection geometry if such information is available. Within the crossing region, the virtual vehicle V is replaced by a probabilistic virtual vehicle pV which has zero velocity, too, but its presence is modeled by the probability $p^{\text{cross}}(t_k)$ of crossing traffic. For each time interval $[t_k, t_{k+1}]$, the transition to *crossing* mode is activated for the fraction $1 - p^{\text{cross}}(t_k)$ of vehicle states within the crossing region. After the transition, the input probability distribution is reset in order to remove the deceleration behavior of the *approaching* phase.

The probability of crossing traffic p^{cross} is defined as the probability that a vehicle is crossing within the time span $\xi \cdot \tau$, where $\xi \in \mathbb{N}^+$ is a parameter that can be freely chosen. The crossing probability of one vehicle is computed as $p^{\text{cross}}(t_k) = \hat{p}(t_k - t_\xi) - \hat{p}(t_k)$, where $\hat{p}$ is the sum of position probabilities in front of the crossing. In the case of several crossing vehicles $1, \ldots, n$, the crossing probability is approximated by $p^{\text{cross}} = \texttt{min}(p^{\text{cross},1} + \ldots + p^{\text{cross},n}, 1)$ since the probability of crossing traffic cannot exceed 1.

It remains to extend the crossing procedure to the case when a vehicle simultaneously approaches a crossing and follows another vehicle. Analogously to the case when the vehicles are simultaneously constrained by the road and a vehicle driving ahead, it is sufficient to consider the constraint vector that is most restrictive. After introducing the constraint vector $\tilde{\eta}^{\text{inter}}$ for interaction with the virtual vehicle V, the overall constraint vector is computed as $\tilde{\eta} = \texttt{min}(\tilde{\eta}^{\text{road}}, \tilde{\eta}^{\text{vehicle}}, \tilde{\eta}^{\text{inter}})$.

The concept for intersection crossing is demonstrated by the following numerical example.

Example 5.3 (Intersection Crossing): In this example, it is assumed that each vehicle moves straight over the crossing. Additional possibilities of going left or right are neglected for simplicity, but can be computed analogously. The cars involved in this scenario are denoted by capital letters, where A, B have right of way and C, D do not have right of way; see Fig. 5.19. As for the previous examples, the Markov chains used for the vehicles are obtained according to the discretization in Tab. 5.2 and the parameters determining the behavior of the vehicles are in Tab. 5.3. The uniform distributions of initial states are specified in Tab. 5.6.

The probability distributions for selected time intervals are plotted in Fig. 5.19. Dark regions indicate high probability, while bright regions represent areas of low probability. In order to improve the visualization, the colors are separately normalized for each vehicle.

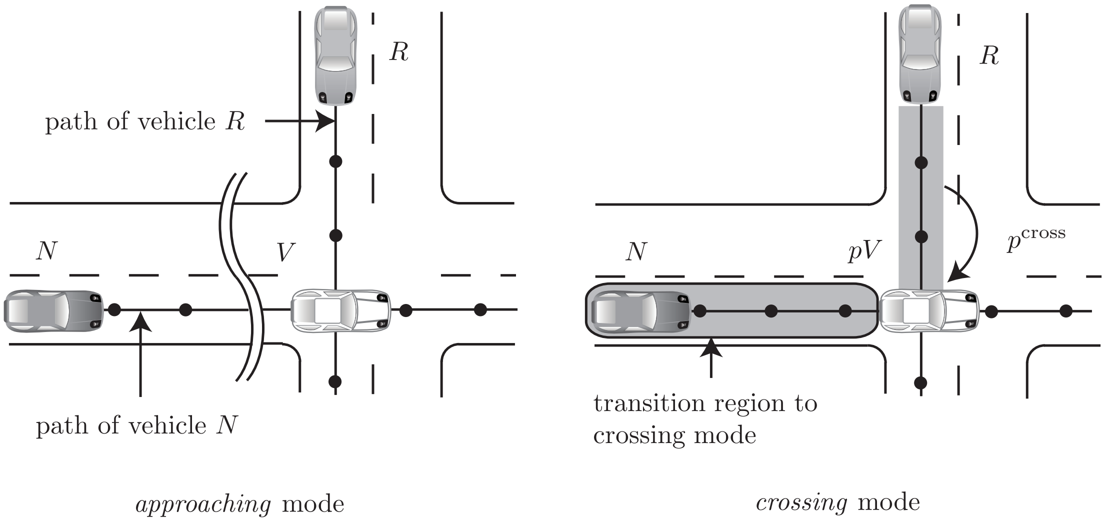
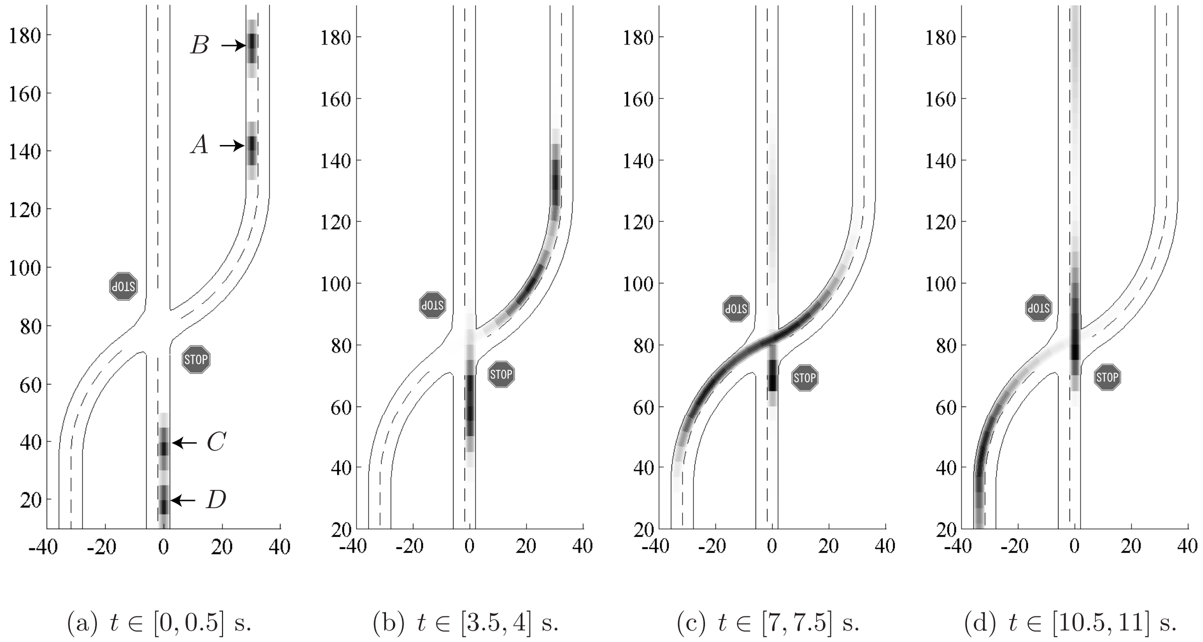


Fig. 5.18.: Intersection scenario: Approaching and crossing mode.

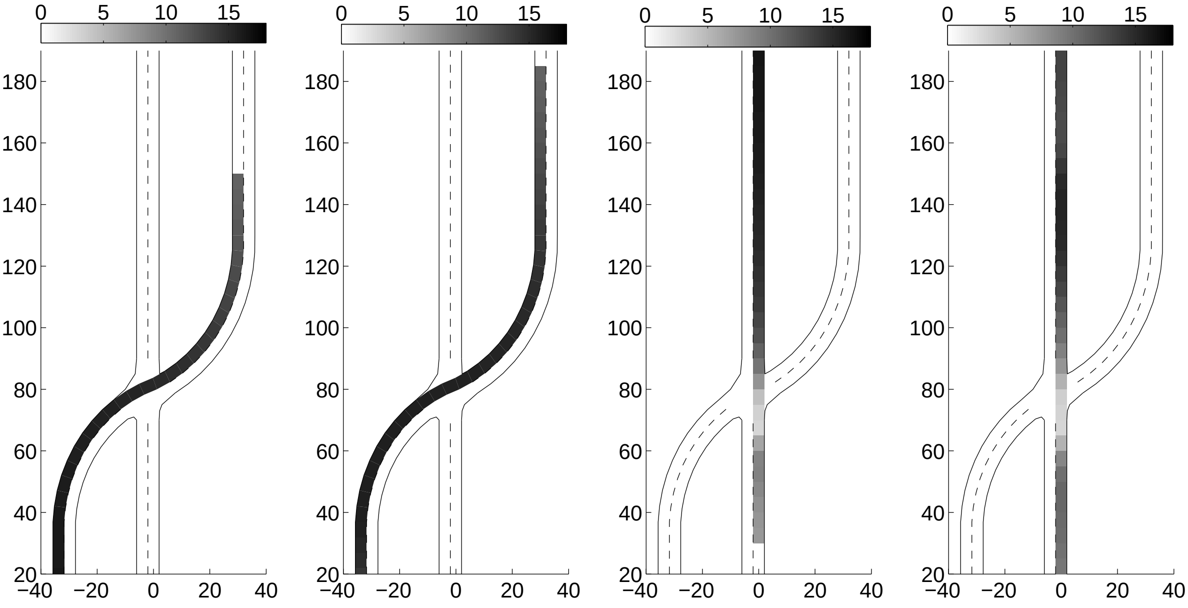
The average velocity of the vehicles along the lane is shown in Fig. 5.20. One can see that the vehicles without right of way are forced to brake. The crossing probability determined by the probability distributions of crossing vehicles A and B is plotted in Fig. 5.21.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 1.68 s when canceling probabilities below $p^{\max} = 10/(d \cdot c) = 4.2e{-}3$ (see Heuristic 5.1) for a prediction horizon of $t_f = 11$ s. □



(a) $t \in [0, 0.5]$ s.
(b) $t \in [3.5, 4]$ s.
(c) $t \in [7, 7.5]$ s.
(d) $t \in [10.5, 11]$ s.

Fig. 5.19.: Position distribution for different time intervals. The coordinate axes refer to positions in [m].



(a) vehicle A.

(b) vehicle B.

(c) vehicle C.

(d) vehicle D.

Fig. 5.20.: Average velocity indicated by the color bar in [m/s] for $t \in [0, 11]$ s. The coordinate axes refer to positions in [m].

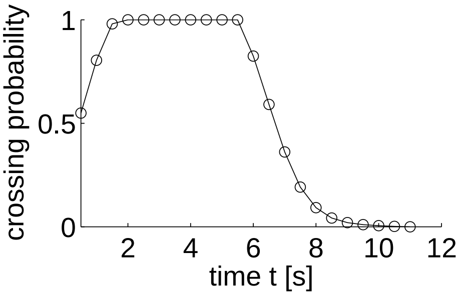


Fig. 5.21.: Probability p^{cross} of the virtual vehicle.

Tab. 5.6.: Initial state: Set with uniform distribution.

vehicle A	$s^A(0) \in [40, 52]$ m $v^A(0) \in [10, 12]$ m/s
vehicle B	$s^B(0) \in [5, 17]$ m $v^B(0) \in [10, 12]$ m/s
vehicle C	$s^C(0) \in [25, 37]$ m $v^C(0) \in [6, 8]$ m/s
vehicle D	$s^D(0) \in [5, 17]$ m $v^D(0) \in [8, 10]$ m/s

5.5.5. Lane Changing

Similar to the intersection crossing behavior, the lane change behavior requires a discrete decision. At intersections, the probability for crossing has to be modeled, and on a multi-lane road, the probability for a lane change has to be modeled. The lane change probability could be forwarded by an external algorithm. In the area of lane change recognition, most work focuses on lane change assistance for human drivers within the ego vehicle; see e.g. [120]. These systems warn the driver if a lane change is predicted and if the lane change is dangerous, where the latter problem is addressed in e.g. [89, 93]. Further, it has been shown that lane change prediction increases the acceptance of automatic cruise control (ACC) systems [65].

Lane change prediction of surrounding vehicles [45] is not so much investigated. Factors contributing to a lane change are discussed in [29]. Besides the prediction of the lane change decision, possible lane change trajectories using HMMs are generated in [126, 127].

In this work, another possibility to estimate the probability of a lane change maneuver is presented. This approach only uses data that has been computed from the Markov chain updates. In order to obtain an algorithm for traffic prediction under possible lane changes which can be computed online, some simplifications have to be made.

Simplifications and Restrictions

The consideration of lane changes adds a significant amount of complexity to the computation of stochastic reachable sets, as will be shown later. For this reason, certain restrictions for lane changes are proposed such that their consideration preserves the online capability of this approach. In order to discuss the restrictions, a scenario with 3 lanes is set up in Fig. 5.22. Note that vehicle A is the autonomous vehicle, while vehicles $B - F$ are surrounding traffic participants. The restrictions are as follows:

1. Lane changes are only considered for vehicles starting within a certain region around the autonomous vehicle: $s^A(0) - \underline{\eta} \leq s \leq s^A(0) + \overline{\eta}$, where $s^A(0)$ is the position of the autonomous vehicle at $t = 0$. Thus, lane changes are only considered for vehicles D and E in Fig. 5.22.
2. A vehicle does a lane change only once within the prediction horizon, i.e. changing two lanes or changing the lane and returning to the original lane is not considered.
3. There is no interaction between vehicles that may change to the same lane. Thus, the probability that vehicle D changes to the middle lane is computed independently of the probability that vehicle E changes to the middle lane.

From the second and third restriction follows that, in principle, the lane changes on a road with n lanes can be broken down to lane changes on several virtual roads with two lanes. For this reason, only two lanes with traffic participants $A - D$ are considered for lane changes from now on. It is remarked that it has not been checked if the assumptions comply with real traffic, which is part of future work. However, the first assumption is reasonable as human drivers also limit their prediction for possible lane changes to the vicinity of their own vehicle. The second assumption is justified as a single lane change takes about 6 seconds, which is similar to the prediction horizon t_f for lane change scenarios [129]. The last assumption does not strongly influence the overall result as the event that two vehicles change to the same lane within the prediction horizon t_f is rare. In the next subsection, the computation of the lane change probability is discussed.

Lane Change Probability Approximation

The following considerations are made for the situation in Fig. 5.22, but for two lanes with vehicles $A - D$, whereof vehicle D performs the lane change. This is no loss of generality as lane change maneuvers can be broken down to this case using the previously discussed assumptions. The probability of a lane change is heuristically obtained from three factors:

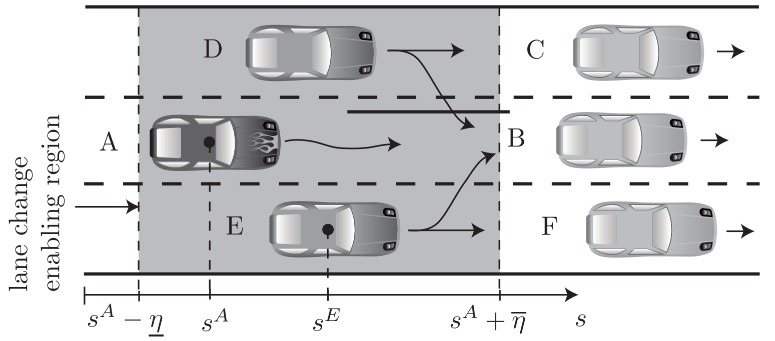


Fig. 5.22.: Considered actions for a three-lane scenario.

1. the motivation σ^{cl} for driving on the current lane,
2. the motivation σ^{nl} for driving on the neighboring lane,
3. the convenience σ^{conv} of the new following vehicle after the lane change is performed.

The motivation values σ^{cl} and σ^{nl} are obtained from the constraint values η_i^α of vehicle D following vehicle C or B, respectively. Similarly, the convenience value σ^{conv} is computed from the constraint values of vehicle A when following vehicle D after the lane change. As an intermediate step, the probabilities e^α of meeting the constraint event C for a given acceleration input are computed:

$$e^\alpha := P(\text{C}|\mathbf{y} = \alpha) = \sum_i P(\text{C}|\mathbf{z} = i, \mathbf{y} = \alpha) P(\mathbf{z} = i) = \sum_i \eta_i^\alpha \hat{p}_i,$$

where independency of $P(\mathbf{z} = i)$ and $P(\mathbf{y} = \alpha)$ is assumed as in Prop. 5.3. Next, the distributions $e^{C,\alpha}$, $e^{B,\alpha}$ and $e^{A,\alpha}$ for interaction with the vehicles A, B, and C are weighted by values stored in a vector $\overline{w}$ for the vehicles in front and a different vector $\underline{w}$ for the new following vehicle:

$$\sigma^{\text{cl}} = \sum_\alpha \overline{w}^\alpha e^{C,\alpha}, \quad \sigma^{\text{nl}} = \sum_\alpha \overline{w}^\alpha e^{B,\alpha}, \quad \sigma^{\text{conv}} = \sum_\alpha \underline{w}^\alpha e^{A,\alpha}.$$

For the motivation values σ^{cl}, σ^{nl}, the weights are chosen such that high positive acceleration has high weights, because the possibility of acceleration motivates driving in a particular lane. For the convenience value σ^{conv}, the weights are chosen such that braking has low values, since it is inconvenient for the new following vehicle to be forced to brake while all positive acceleration inputs have similar weights. Finally, the probability for a lane change p^{lc} is heuristically computed as

$$p^{\text{lc}} = \frac{2}{\pi} \arctan\left(b \, \frac{\sigma^{\text{nl}}}{\sigma^{\text{cl}}} \frac{\sigma^{\text{conv}}}{\sum \underline{w}} \right).$$

The heuristic is chosen such that a motivation ratio $\frac{\sigma^{\text{nl}}}{\sigma^{\text{cl}}}$ in favor of the neighboring lane as well as a high convenience value σ^{conv} motivates a lane change. The arctan function allows the modeling of the saturation ($\mathbb{R}^+ \to [0, 0.5\pi]$), but can also be replaced by a similar function. The summation $\sum \underline{w}$ is the maximum possible value for σ^{conv}, which normalizes the convenience value. This is not necessary for σ^{nl}, σ^{cl} as one is only interested in their ratio. Where there is no vehicle B or C, the corresponding σ values are set to $\sum \overline{w}$ or $\sum \underline{w}$. The parameter b is a tuning parameter that has to be found from traffic observations. Besides the probability of a lane change, the lane change maneuver itself has to be considered, which is presented next.

Longitudinal and Lateral Probability Distributions

The probability for a lane change obtained from the previous scenario with vehicles $A-D$ strongly influences the future probability distribution. In order to refer to the cases when vehicle D drives on the left or right lane, the notation Dl and Dr is used, respectively. The longitudinal probabilities for the left and right lane are computed similarly as in previous scenarios. The extension is that the probability for a lane change decreases the probability of driving in one lane while it increases the probability in the other lane. Additionally, the input probability distribution after the lane change has to be changed to the distribution of the new lane by the input reset operator, which is defined as $\mathtt{inputReset}(\Delta p_i^\alpha(t_k)) := \lambda_i^{Dr,\alpha}(t_k)\sum_\alpha \Delta p_i^\alpha(t_k)$ when changing to the right lane and $\Delta p_i^\alpha(t_k)$ are the probability values that are added to the right lane. The enhanced probability update of (4.25) when changing from left to right is

$$\begin{aligned}\Delta\tilde{p}(t_k) &= p^{\mathrm{lc}}(t_k)\tilde{p}^{Dl}(t_k),\\ \tilde{p}^{Dl}(t_{k+1}) &= \tilde{\Gamma}^{Dl}(t_k)\,\tilde{\Phi}(\tau)\,\tilde{p}^{Dl}(t_k) - \Delta\tilde{p}(t_k),\\ \tilde{p}^{Dr}(t_{k+1}) &= \tilde{\Gamma}^{Dr}(t_k)\,\tilde{\Phi}(\tau)\,\tilde{p}^{Dr}(t_k) + \mathtt{inputReset}(\Delta\tilde{p}(t_k)),\end{aligned}$$

while the updates for time intervals are identical to (4.25).

Besides the longitudinal probability distribution, the lateral distribution has to be changed as well. For this, the lane change is divided into phases with time span $\tau = t_{k+1} - t_k$. After each time step, the probabilities are shifted to the next phase until the lane change has been completed. This is illustrated in Fig. 5.25, where t^{lc} is the time passed since the lane change was initiated and the gray areas indicate the phases of driving in the initial and final lane. The probabilities of the lane change phases are denoted by $\hat{p}_l^{\mathrm{lc}}$ and the index refers to the l-th phase. In order to account for the uncertainty in the lateral position during a lane change, a deviation probability $f_l(\delta)$ is defined for each phase. The final deviation probability, which is spanned over the initial and neighboring lane, is computed as $f(\delta) = \sum_l \hat{p}_l^{\mathrm{lc}} \cdot f_l(\delta)$. The lane change probabilities and stochastic reachable sets for a lane change scenario are computed in the following example.

Example 5.4 (Lane Changing): The considered traffic situation is depicted in Fig. 5.23 with vehicles $A-D$, where vehicle A is the autonomous vehicle and a lane change is only considered for vehicle D. The traffic situation can easily be extended to three lanes as previously discussed. In order to motivate a lane change for vehicle D, the initial speed of vehicle C is chosen lower than for the vehicles A and B in the adjacent lane.

The parameters and settings are identical to the ones found in Tab. 5.2 and 5.3, except that the values of the conditional input probability $q_i(0)$ differ from vehicle to vehicle. The initial state distributions are presented in Tab. 5.7 and the weighting vectors for the lane change probability are $\overline{w} = \begin{bmatrix}0 & 1 & 2 & 3 & 4 & 4\end{bmatrix}$, $\underline{w} = \begin{bmatrix}0 & 0 & 0 & 1 & 1 & 1\end{bmatrix}$ and $b = 0.11$.

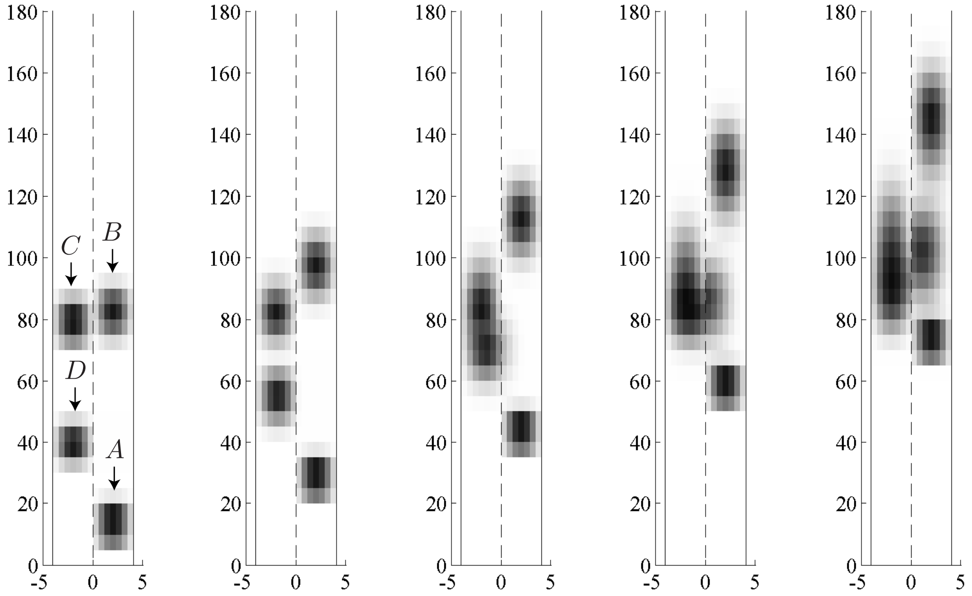
The probability distributions on the road are displayed in Fig. 5.23 for 5 selected time intervals. Dark regions indicate high probability, while bright regions represent areas of

Tab. 5.7.: Initial state: Set with uniform distribution.

vehicle A	vehicle B	vehicle C	vehicle D
$s^A(0) \in [7, 13]$ m $v^A(0) \in [14, 16]$ m/s	$s^B(0) \in [72, 84]$ m $v^B(0) \in [14, 16]$ m/s	$s^C(0) \in [72, 84]$ m $v^C(0) \in [4, 8]$ m/s	$s^D(0) \in [30, 40]$ m $v^D(0) \in [14, 16]$ m/s

low probability. In order to improve the visualization, the colors are separately normalized for each vehicle. The average velocities are displayed in Fig. 5.24. Further, the normalized motivation values σ^{cl}, σ^{nl}, convenience value σ^{conv}, and lane change probability p^{lc} are plotted in Fig. 5.26 and the probability distribution $\hat{p}_l^{\text{lc}}$ of the lane change phases at $t = 4$ s is plotted in Fig. 5.25.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.43 s when canceling probabilities below $p^{\max} = 10/(d \cdot c) = 4.2e - 3$ for a prediction horizon $t_f = 5$ s.



(a) $t \in [0, 0.5]$ s. (b) $t \in [1, 1.5]$ s. (c) $t \in [2, 2.5]$ s. (d) $t \in [3, 3.5]$ s. (e) $t \in [4, 4.5]$ s.

Fig. 5.23.: Position distribution for different time intervals. The coordinate axes refer to positions in [m].

5.5.6. Known Behavior

Finally, the case is considered when the future behavior of other traffic participants is known. The planned trajectory $\hat{x}(t)$ of other vehicles is available when e.g. autonomous vehicles broadcast their planned trajectories to other autonomous vehicles. The planned

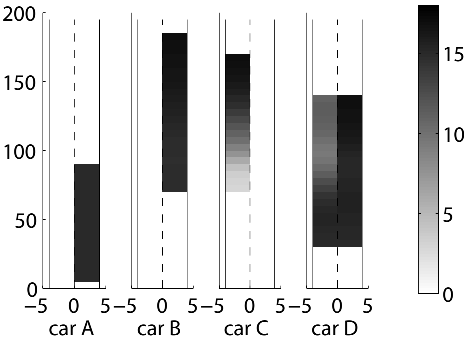


Fig. 5.24.: Average velocity indicated by the color bar in [m/s] for $t \in [0, 5]$ s. The coordinate axes refer to positions in [m].

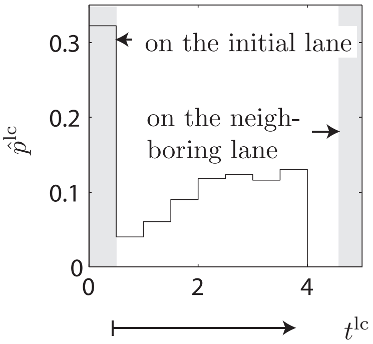


Fig. 5.25.: Probability of lane change phases at $t = 4$ s.

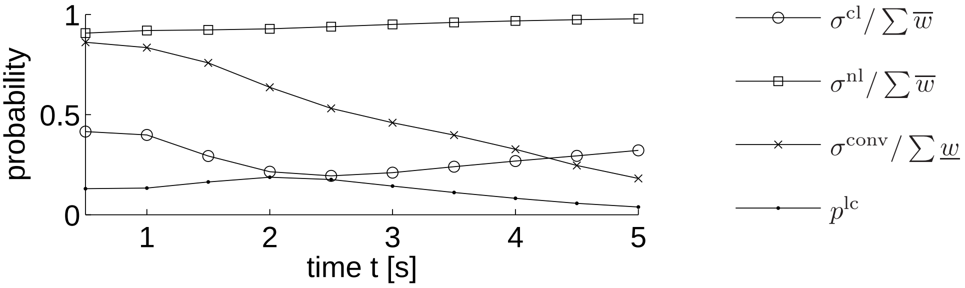
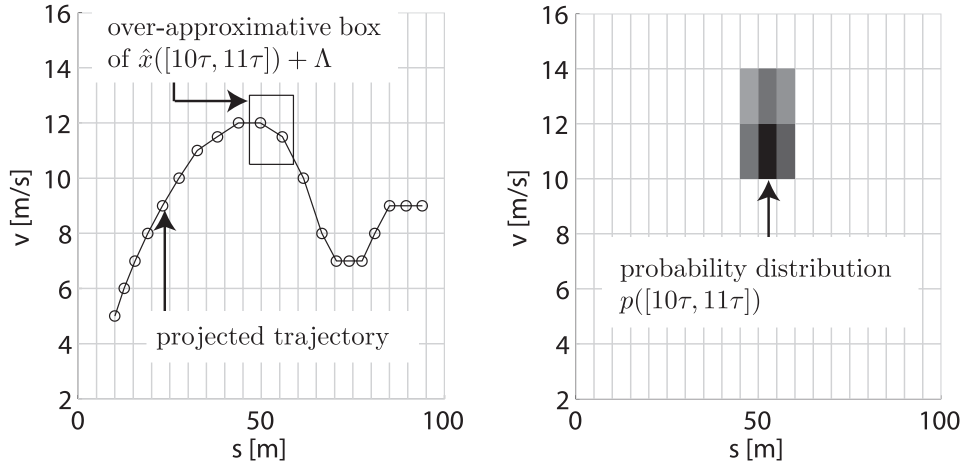


Fig. 5.26.: Motivation values.

trajectory of the ego vehicle is always known since the trajectory planner and the safety verification module are connected (see Fig. 5.1).

In practice, the planned trajectory cannot be perfectly realized, which is modeled by a bounded uncertainty Λ so that $x(t) \in \hat{x}(t) + \Lambda$, where the set Λ is translated by $\hat{x}(t)$. The connection to the probabilistic description of vehicles is established by first projecting the planned trajectory $\hat{x}(t)$ and the bounded uncertainty Λ onto the position and the velocity coordinate; see Fig. 5.27(a). It is assumed that the probability is uniformly distributed within Λ so that the probability of a state space cell is determined by the fraction of the volume in a cell. This is exemplarily presented for the uncertainty in Fig. 5.27(a), whose corresponding probability distribution within the discretized state space is shown in 5.27(b). The same procedure is performed for the input space.

In this section, the behavior of vehicles for road following, vehicle following, intersection crossing, and lane changing has been modeled. It remains to compute the probability of a crash of the ego vehicle with other vehicles. This is based on the probability distributions of all vehicles in the traffic scene, which is discussed next.



(a) Projected trajectory.

(b) Discrete probability distribution.

Fig. 5.27.: Mapping of known behavior to discrete probabilities.

5.6. Crash Probability

In this section, the crash probability of the ego vehicle with all other vehicles in the traffic scene is estimated. The other question of guaranteeing whether a crash may occur or not, can be answered by checking if the reachable positions of the ego vehicle intersect with the ones of all other vehicles, which are computed as presented in Sec. 5.4.1. In the case of a conflict, i.e. the estimated crash probability is zero, but an intersection of reachable sets has been detected, a low crash probability is assumed.

Before the approach for the computation of the crash probability is presented, some assumptions are made which allow the crash probability as it is understood in this work to be defined. The purpose of the safety assessment of autonomous vehicles is to obtain a measure for the threat at different points in time when the planned trajectory is executed as planned. This plan should be strictly followed by the autonomous car, which implies that the errors made when following the planned trajectory are independent of the future behavior of other traffic participants. Note that the compliance to the current plan during the safety assessment is no contradiction to the continuous updates of the trajectory planner; see Fig. 5.1.

Another aspect is that the crash probability is computed independently of previous crash probabilities in this work. This has the advantage that for each point in time, a situation can be judged independently of previous occurrences. This is not the case, when computing the physical probability that a crash will happen. Consider a scenario in which two situations are equally dangerous at two different points in time. However, the probability that the vehicle crashes in the first situation is much greater than in the second situation, because a crash can only occur in the second situation, if the vehicle survived the first situation and crashes in the second situation. For this reason, it is assumed that the autonomous vehicle has not crashed until the investigated point in time or time interval. One can also say that the physically motivated crash probability is the total probability of a crash, while the definition used in this work is the conditional probability of a crash under the condition that no crash has happened yet. Clearly, if the probability of a crash is low for all points in time, both definitions yield similar results.

The definition is detailed for a situation with only one other traffic participant. For a better distinction of variables from the other vehicle with the ones of the ego car, all variables referring to the ego car are indexed by a hat ($\hat{\square}$) from now on.

Definition 5.1 (Conditional Crash Probability): Given are the vectors $\hat{\xi}, \xi \in \mathbb{R}^\eta$ which uniquely define the position and orientation of the ego vehicle and another vehicle, respectively. The probability distributions at time t_k are denoted by $f(\xi, t_k)$ and $\hat{f}(\hat{\xi}, t_k)$. For both distributions, it is assumed that the ego vehicle has not yet crashed. The indicator function $\texttt{ind}(\xi, \hat{\xi})$ is 1 if the bodies of the ego vehicle and the other vehicle intersect and 0 otherwise. Under the previously discussed assumptions that the probability distributions $f(\xi)$ and $\hat{f}(\hat{\xi})$ are independent, and the ego vehicle has not crashed until the investigated point in time t_k, the conditional crash probability $p^{\text{crash}}(t_k)$ is defined as

$$p^{\text{crash}}(t_k) = \int_{\mathbb{R}^\eta} \int_{\mathbb{R}^\eta} f(\xi, t_k) \cdot \hat{f}(\hat{\xi}, t_k) \cdot \texttt{ind}(\xi, \hat{\xi}) \, d\hat{\xi} \, d\xi. \qquad \square$$

In this work, ξ and $\hat{\xi}$ are two-dimensional position vectors of the vehicle centers. The orientation is indirectly specified since the vehicles are always oriented according to the tangential vector of their path.

This definition is extended to several traffic participants by computing the partial crash probabilities with other traffic participants. The final conditional crash probability is obtained by summing over all partial probabilities. Note that the above definition can only be applied to the ego vehicle and not to other traffic participants, since their probability distributions are no longer independent.

5.6.1. Computational Realization

For the implementation of the above definition, the probability distribution $f(\xi, t_k)$ of traffic participants has to be formalized. Since the path and the deviation along the path are segmented, and the probability distribution is uniform within one segment, one obtains a piecewise constant probability distribution in $\mathbb{R}^2$ for $f(\xi, t_k)$. In order to obtain regions of constant probability distribution with simple geometry, waypoints on the paths of other traffic participants are extracted every path segment distance. These waypoints are connected to a simplified path consisting of connected straight lines. The straight lines are also referred to as the simplified path segments s_g, where g is the path index. The deviation is also subdivided into segments d_h, where h is the deviation index. The segmentation of the path and the deviation can be observed in Fig. 5.28 and 5.4.

The region that is spanned when the path coordinate s is within s_g and the deviation coordinate δ is within d_h is denoted by $\mathcal{C}_{gh}$. The regions $\mathcal{C}_{gh}$ are trapezoids and the union of all trapezoids results in a path-aligned occupancy grid. A further region that is of interest, is the region that is occupied by the body of a vehicle when the center of the body is within $\mathcal{C}_{gh}$. This set is denoted by $\mathcal{B}_{gh}$ and its orientation equals the direction of the path segment g, as illustrated in Fig. 5.28.

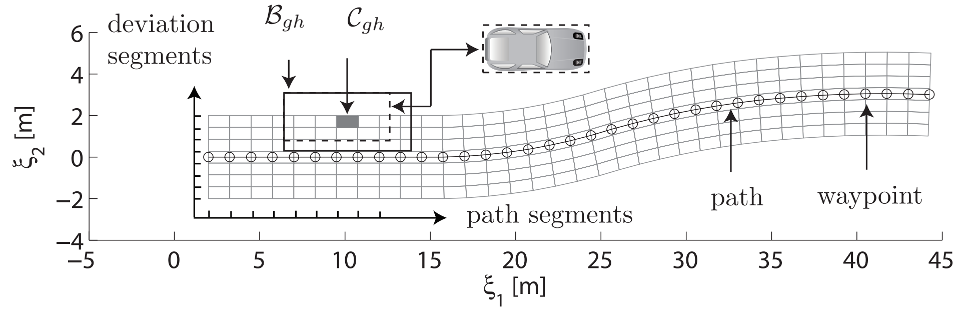


Fig. 5.28.: Exemplary path with occupancy regions.

In order to compute the crash probability, the probability p^{pos}_{gh} that the center c of a traffic participant is located in $\mathcal{C}_{gh}$ has to be computed first. Introducing the probability for a deviation segment $p^{\text{dev}}_h = P(\delta \in \mathscr{d}_h)$ and the one for a path segment $p^{\text{path}}_g = P(s \in \mathscr{s}_g)$, the probability that $(s \in \mathscr{s}_e \wedge \delta \in \mathscr{d}_f) \leftrightarrow c \in \mathcal{C}_{gh}$ is $p^{\text{pos}}_{gh} = p^{\text{path}}_g \cdot p^{\text{dev}}_h$ due to the independency assumption in Sec. 5.3. The deviation probabilities are fixed over time or time varying when considering a lane change maneuver.

In contrast to the deviation probabilities, the path segment probabilities have to be extracted from the joint probabilities of state and input p^{α}_i originating from the Markov chain computations. This is done by first summing over all inputs $\hat{p}_i = \sum_\alpha p^{\alpha}_i$. Each state space cell i represents a position and velocity interval $\mathscr{s}_e$ and $\mathscr{v}_m$, such that $\mathcal{X}_i = \mathscr{s}_e \times \mathscr{v}_m$, where only the probability of the path segments $\mathscr{s}_e$ is of interest: $p^{\text{path}}_e = \sum_m P(s \in \mathscr{s}_e, v \in \mathscr{v}_m)$ and $P(s \in \mathscr{s}_e, v \in \mathscr{v}_m) \leftrightarrow P(x \in \mathcal{X}_i) = \hat{p}_i$.

Besides the probability that the center of a vehicle is in a certain trapezoid $\mathcal{C}_{gh}$, it is necessary to know the probability that two vehicle bodies intersect when the center of one of the bodies is in $\mathcal{C}_{gh}$ and that of the other one is in $\hat{\mathcal{C}}_{ef}$. This probability is denoted by p^{int}_{ghef} and is computed by uniformly gridding the uncertain sets $\mathcal{C}_{gh}$ and $\hat{\mathcal{C}}_{ef}$ as shown in Fig. 5.29. The grid points are possible centers of vehicle bodies and by counting the relative number of cases for which the bodies intersect, the probability p^{int}_{ghef} is obtained. Instead of gridding the set of vehicle centers, one can also conservatively assume that $p^{\text{int}}_{ghef} = 1$ if the sets of possible vehicle bodies $\mathcal{B}_{gh}$ and $\hat{\mathcal{B}}_{ef}$ intersect, and $p^{\text{int}}_{ghef} = 0$ otherwise. This computation has been used in [188] and generates overestimated crash probabilities. From now on, the computation of the crash probability according to the overestimated value of p^{int}_{ghef} is referred to as the *conservative* computation and the one according to the estimated value of p^{int}_{ghef} is referred to as the *relaxed* computation.

The probabilities p^{pos}_{gh} that centers are in certain regions $\mathcal{C}_{gh}$ and the probability p^{int}_{ghef} that two vehicle bodies intersect when their centers lie in $\mathcal{C}_{gh}$ and $\hat{\mathcal{C}}_{ef}$, allow the conditional crash probability according to Def. 5.1 to be computed by

$$p^{\text{crash}} = \sum_{g,h,e,f} p^{\text{int}}_{ghef} \cdot \hat{p}^{\text{pos}}_{gh} \cdot p^{\text{pos}}_{ef} \,. \tag{5.6}$$

The sum is taken over all possible combinations of g, h, e, f. This results in a huge number of possible combinations so that the following computational techniques are used to accelerate the computation.

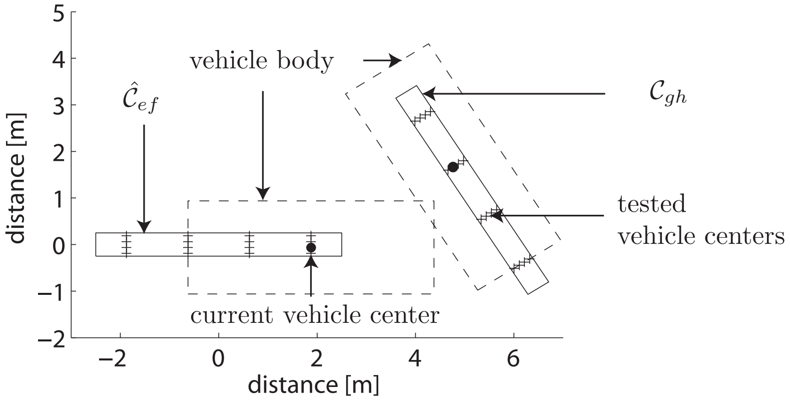


Fig. 5.29.: Estimating the intersection probability of vehicle bodies given the sets of vehicle centers.

5.6.2. Efficient Computation

In order to accelerate the computation of (5.6), pairs of vehicle body regions $(\mathcal{B}_{gh}, \hat{\mathcal{B}}_{ef})$ have to be found that intersect. Then, only a subset of combinations of indices g, h, e, f has to be considered in (5.6). In order to efficiently find intersecting pairs $(\mathcal{B}_{gh}, \hat{\mathcal{B}}_{ef})$, two steps are suggested:

1. First, it is checked if the vehicle bodies $\bigcup_f \mathcal{B}_{ef}$ of a certain path segment e can possibly intersect the vehicle bodies belonging to a path segment g of the ego car. For this, it is checked if the circles enclosing the set of vehicle bodies intersect, where circles are chosen since checking for their intersection is computationally cheap; see Fig. 5.30.
2. Next, the set of vehicle bodies $\mathcal{B}_{gh}$, $\hat{\mathcal{B}}_{ef}$ belonging to pairs of path segments passing the first test are again checked for intersection by the same procedure using enclosing circles.

Besides reducing the number of index combinations, the crash probability computation in (5.6) can be further accelerated by precomputing the probabilities p^{int}_{ghef}. Internally, the precomputed intersection probabilities p^{int} are stored by relative orientation and translation of uncertain centers $\mathcal{C}$. There are different look-up tables for p^{int}_{ghef} depending on checking intersection of the autonomous vehicle with a bicycle/bike, a car, or a truck.

5.6.3. Discussion

The computation of the crash probability based on the probability distribution of traffic participants within consecutive time intervals has the advantage, that no point of time is missed. For example, it prevents the tunneling effect which occurs when two vehicles cross each other at high velocity. In this case, the stochastic reachable sets of points in time might not intersect, while the stochastic reachable sets of time intervals intersect. The disadvantage of the computation with time intervals is that the uncertainty is greater than for points in time. This can lead to wrong crash probabilities as the following example shows.

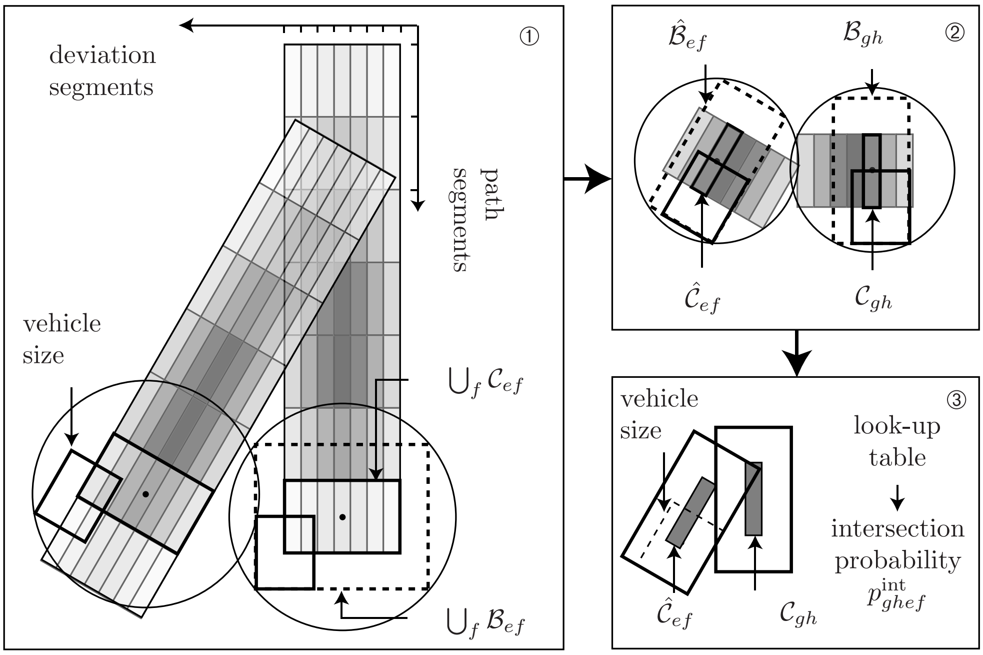


Fig. 5.30.: Crash probability obtained from stochastic reachable sets.

Example 5.5 (Crash Probability Evaluated for a Time Interval): Given are two vehicles driving one after the other with equal and constant velocity. The distance between the vehicles (bumper to bumper) is chosen as 10 m and the velocity is chosen as 30 m/s. For a time interval of $t \in [0, 0.5]$ s, the front bumper of the following vehicle and the rear bumper of the leading vehicle travel the distances as depicted in Fig. 5.31. Clearly, the crash probability is zero since both vehicles travel with equal velocity. However, since the corresponding bumpers of the following and leading vehicle are in the region between 10 m and 15 m with a probability of 1/3 in the time interval $t \in [0, 0.5]$ s, the crash probability is computed as 1/9 when using the independency assumption.

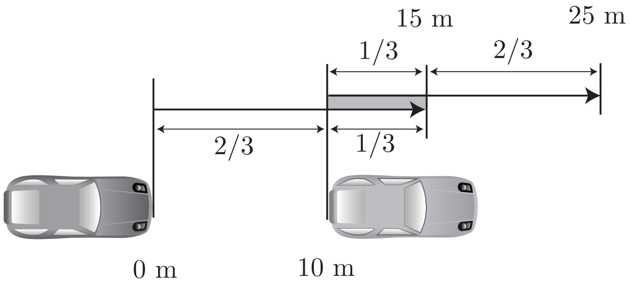


Fig. 5.31.: Crash probability example.

This error can be tackled by an extension, where in contrast to the previous approach, not only the position and orientation of the vehicles is considered, but the velocity is included, too. This is possible when using an enhanced look-up table in which the probabilities for a crash within the time interval $t \in [0, \tau]$ are stored based on the relative position, orientation, and the velocity intervals of both vehicles at $t = 0$. The disadvantage of the required look-up table is that it is very large and that reduction techniques such as the exclusion of pairs of occupancy regions cannot be applied anymore. These two aspects easily make the extended approach about 100 times slower. However, this extension results in more accurate crash probabilities (see [194]). But after comparing the additional effort with the achieved improvement, this extension in not used here. The used approach is demonstrated for an overtaking scenario in the next example.

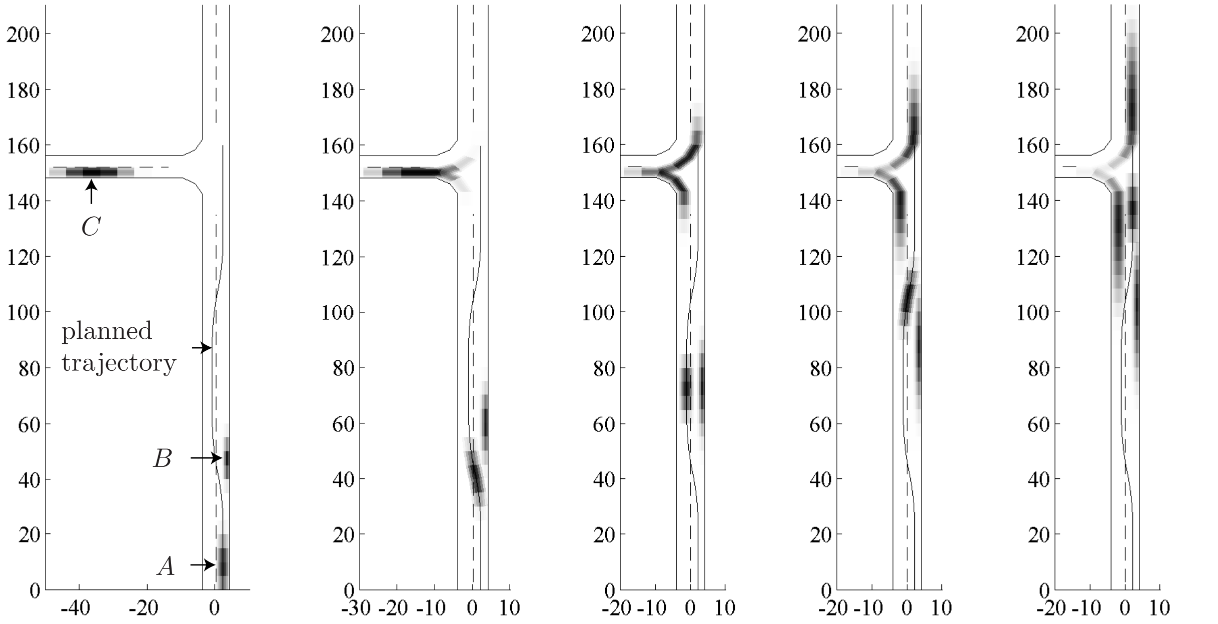
Example 5.6 (Crash Probability for an Overtaking Scenario): The crash probability of different time intervals according to the crash probability computation in (5.6) is computed for an overtaking scenario. In this scenario, the autonomous car A plans to overtake a bicycle B while another car C is approaching. The other car C enters the considered lane from a T-intersection and thus has the option of turning left or right. Since there is no probabilistic information on the turning behavior available, both options are considered with probability 1 such that the crash probabilities with the car are not computed too small. The parameters and settings are identical to the ones found in Tab. 5.2 and 5.3.

The probability distributions on the road are displayed in Fig. 5.32 for 5 selected time intervals. Dark regions indicate high probability, while bright regions represent areas of low probability. In order to improve the visualization, the colors are separately normalized for each vehicle. Since bicycle riders tend to ride close to the side of a lane, the lateral probability distribution is chosen differently to the one of cars. The crash probabilities of the autonomous car at different time intervals with different traffic participants is shown in Fig. 5.33. This plot shows the independent crash probabilities computed according to (5.6). The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.26 s for the probability distributions and 0.49 s for the computation of the crash probabilities, resulting in 0.75 s for a prediction horizon of $t_f = 9$ s. A probability reduction with $p^{\max} = 3/(d \cdot c) = 1.25e - 3$ was used.

More investigations on the accuracy of the crash probability computations are discussed in the next section on Monte Carlo simulation.

5.7. Comparison to Monte Carlo Simulation

In the previous sections of this chapter, a framework for the safety assessment of traffic situations using Markov chains was presented. In order to check these results and get an idea of the efficiency of the Markov chain approach, it is compared to Monte Carlo simulations in this section. The term Monte Carlo simulation or Monte Carlo method refers to methods that are based on random sampling and numerical simulation. Monte Carlo methods are especially popular for complex and highly coupled problems where other probabilistic approaches fail. One of the biggest applications of Monte Carlo simulation is risk analysis, such as the risk analysis of road traffic considered in this thesis. As already mentioned in the introduction of this chapter, Monte Carlo simulation has been applied to the risk analysis of road traffic [15, 31, 32, 52, 59, 60] and air traffic [25, 26, 168]. The results presented below have mainly been obtained from the supervised bachelor thesis of Alexander Mergel [194].



(a) $t \in [0, 0.5]$ s. (b) $t \in [2, 2.5]$ s. (c) $t \in [4, 4.5]$ s. (d) $t \in [6, 6.5]$ s. (e) $t \in [8, 8.5]$ s.

Fig. 5.32.: Position distribution for different time intervals in the overtaking scenario. The coordinate axes refer to positions in [m].

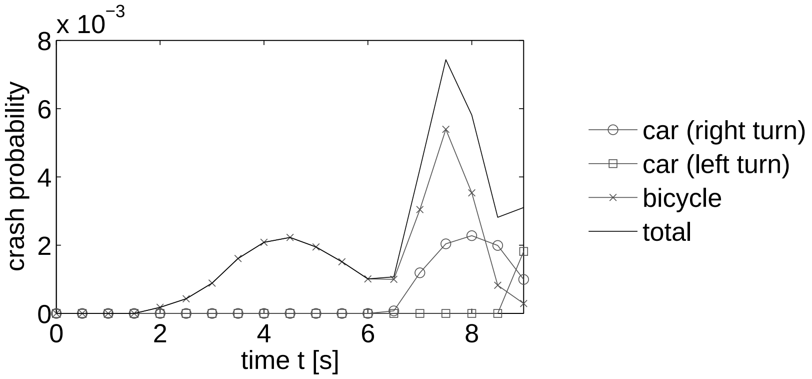


Fig. 5.33.: Crash probability of the overtaking scenario.

5.7.1. Basic Approach

There exists a huge variety of Monte Carlo methods and thus one cannot give a strict guidance on how to apply them in general. However, most methods exhibit the following scheme.

1. Define a domain of possible inputs and initial states.
2. Generate inputs and initial states randomly from the previously specified domains.
3. Perform a deterministic computation starting at the initial states subject to the randomly generated inputs.

4. Aggregate the results of the individual computations into the final result.

For the dynamic model of traffic participants (5.1), the domain of initial states is the set of initial positions and velocities, and the domain of inputs is the normalized range $[-1, 1]$ of acceleration commands. The initial states are randomly distributed according to the initial distribution. The input distribution is more complicated since it evolves over time so that the spectral density plays a major role in the outcome of the Monte Carlo simulation, which is discussed later in detail. The aggregation of the results differs from the two purposes pursued in this work.

One purpose is to compute the probability distribution of traffic participants for future points in time. When using importance sampling, i.e. the samples are created according to their probability distribution so that they have equal weight, the relative number of simulations ending up in certain state space cells determine the probability of reaching these state space regions. This procedure is exactly the same as for the determination of transition probabilities of Markov chains using Monte Carlo simulation; see Sec. 4.3.2. The other purpose is to compute the probability that the autonomous vehicle is crashing in a certain traffic scenario. The ratio of counted collisions N^{crash} to the number of simulations N_s determines the crash probability $p^{\text{crash}} = N^{\text{crash}}/N_s$ when using importance sampling. An alternative way of computing the crash probability is to determine the probability distributions of all traffic participants using Monte Carlo simulation first. In a second step, the crash probability is determined as for the Markov chain approach in Sec. 5.6. However, this procedure is not advantageous, as shown later.

An intrinsic property of Monte Carlo simulation is that the result of the computations is not deterministic, i.e. the result differs from execution to execution. Obviously, this is because the samples for the deterministic simulation are randomly generated. Thus, the probability distributions are possibly far from the exact solution. The good news, however, is that the mean error scales with $\frac{1}{\sqrt{N_s}}$ where N_s is the number of simulations. This result can be derived from Monte Carlo integration, which is briefly introduced in Appendix B. This is completely different when computing with Markov chains, where equal results are obtained for identical initial conditions since no random sampling is applied.

5.7.2. Random Number Generation

One of the most important tasks in Monte Carlo simulation is the generation of random numbers. This is a vast topic itself, dealing e.g. with the problem of how to generate random numbers from computers which are deterministic machines. For this reason, these random numbers are also called pseudo-random numbers and their degree of randomness is checked via certain tests; see e.g. [171]. Here, it is assumed that random numbers can be produced with sufficient quality from a uniform distribution in the interval $[0, 1]$. It remains to map these random numbers such that they are distributed according to a specified distribution.

In the Markov chain approach, the probability distributions are discrete. Each discrete probability represents the probability that a state or an input belongs to a certain cell of the discretized state or input space. In order to obtain a probability distribution in the continuous space, it is assumed that the probability distribution is uniform within a cell. This assumption has also been made for the generation of transition probabilities of Markov chains; see Sec. 4.3. Thus, the initial and input distribution is piecewise constant; see e.g. Fig. 5.34(a).

In order to compare the results of the Markov chain approach with the ones of the Monte Carlo simulation, random numbers according to the piecewise constant distributions have to be generated. The simplest approach for this task is the inverse transform method. This method is based on the inverse of the cumulative distribution, which always exists since only piecewise constant distributions are considered.

The method is illustratively described in Fig. 5.34 for a probability density function of the acceleration command u. First, the cumulative distribution of the input $F(u)$ is computed by integration of the probability density function $f(u)$. Next, the inverse of the cumulative distribution $F^{-1}(r)$ is computed, where r is the new argument. When generating uniformly distributed values of $\mathbf{r}$ within $[0,1]$, the random values of $F^{-1}(\mathbf{r})$ are distributed according to the probability density function $f(u)$. Note that r refers to possible values of the random variable $\mathbf{r}$. The proof is straightforward:

$$P(\mathbf{u} \leq u) = P(F^{-1}(\mathbf{r}) \leq u) = P(\mathbf{r} \leq F(u)) = F(u).$$

The inverse transform method is used when generating the initial probability distribution of the state as well as for the random inputs. For the random inputs, one has to additionally consider correlations between time steps, which is discussed next.

(a) Probability density function. (b) Cumulative density function. (c) Inverse cumulative density function.

Fig. 5.34.: Inverse transform method for the generation of random samples.

5.7.3. Input Generation

In the Markov chain approach, not only the states, but also the inputs are generated by a Markov chain. More precisely, the conditional input distribution $q_i^\alpha = P(\mathbf{y} = \alpha | \mathbf{z} = i)$ is updated according to $q_i^\alpha([t_k, t_{k+1}]) = \sum_\beta \Gamma_i^{\alpha\beta}(t_k) q_i^\beta([t_{k-1}, t_k])$, where $q_i^\beta([t_{k-1}, t_k]) \in \{0, 1\}$ when using Monte Carlo simulation; see (4.22). Input samples for the Monte Carlo simulation are generated from the discrete probabilities $q_i^\alpha([t_k, t_{k+1}])$ as previously described in Sec. 5.7.2.

Another option to create random inputs is proposed by Broadhurst and Eidehall in works on Monte Carlo simulation of road traffic scenes [32, 52, 59, 60]. In these approaches, a random trajectory is created first, and afterwards its likeliness is evaluated. The values of the input trajectories are created by an IID[6] process, but are no longer IID after a weight is assigned by the likeliness function. Before presenting the likeliness function (or also called goal function), some notations are introduced and recapitulated. The deviation from the lane center is denoted by δ, the velocity by v, the normal acceleration by a_N, the steering wheel angle by ϕ, the maximum velocity by $v^{\max}$, and the number of input values by $N_u = t_f/\tau$ (t_f is the time horizon and τ the time increment). The goal function is chosen in [32] to

$$g(u([0,t_f])) = -\sum_{k=1}^{N_u} \left(\lambda_1 \delta(t_k)^2 + \lambda_2 (v(t_k) - v^{\max})^2 + \lambda_3 a_N(t_k)^2 + \lambda_4 \phi(t_k)^2\right), \tag{5.7}$$

where λ_1–λ_4 are tuning parameters which punish the following: Large deviations from the lane center, velocity deviations from the allowed velocity, large accelerations, and steering wheel angles. The punishment of accelerations and steering angles ensures a comfortable ride for the passengers. The probability distribution of the input trajectories $f(u([0,t_f]))$ is assumed to be

$$f(u([0,t_f])) = c_n \exp(k_g \cdot g(u([0,t_f]))) \tag{5.8}$$

according to [32], where the previously introduced goal function is in the exponent. The value k_g is another tuning parameter and c_n is the normalization constant. The related publications [52, 59, 60] use almost the same model, but with different parameter values and the tangential acceleration a_T instead of the steering angle ϕ. It is remarked that the tuning parameters are set according to simulations and not learned from driving experiments in real traffic.

Since the input values are generated independently of previous values, consecutive input values are not strongly correlated despite the weighting. In contrast to this, the input values generated by a Markov chain are correlated by the input transition values in Γ. This is checked by the autocorrelation, i.e. the correlation of the signal against a time-shifted version of itself:

$$S(t,\tau) := E[\mathbf{u}(t)\mathbf{u}(\tau)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u_t u_\tau f(u_t u_\tau)\, du_t\, du_\tau, \tag{5.9}$$

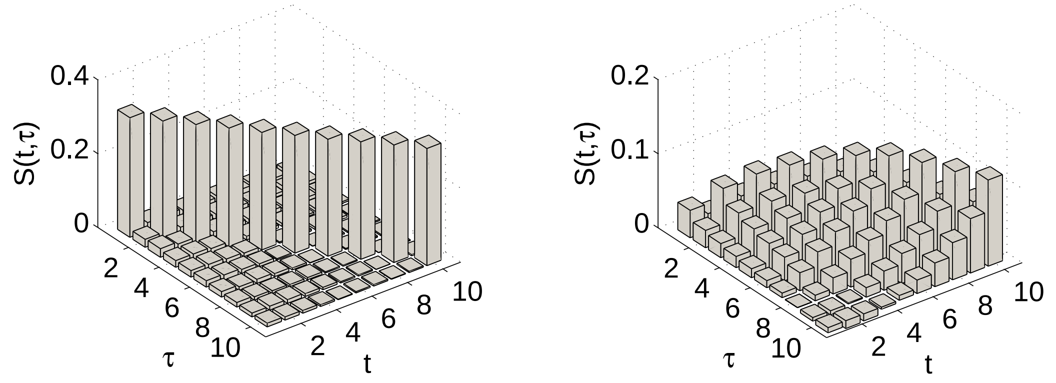
where $E[\,]$ is the expectation, $f(\,)$ is the probability density function, and u_t is a realization of the random variable $\mathbf{u}(t)$. The computation of the above integrals is approximated using Monte Carlo integration as shown in (B.1).

The autocorrelation values for the input trajectories generated from the Markov chain approach and the approach proposed by Broadhurst and Eidehall are plotted in Fig. 5.35. The input trajectories for the autocorrelation computation are uniformly sampled and generated with parameters listed in Tab. 5.8. Note that the parameters λ_1 and λ_4 are set to 0 since it is assumed that the vehicles drive perfectly on their path. There is almost no correlation between the inputs of different time steps in the approach proposed by Broadhurst and Eidehall; see Fig. 5.35(a). This is in contrast to the Markov chain approach, where the input signals are much more correlated, as shown in Fig. 5.35(b).

[6] IID: independent and identically distributed.

Tab. 5.8.: Parameters for input trajectory generation.

General		Broadhurst/Eidehall		Markov chain	
N_s	$1e5$	k_g	100	μ	$[0.01, 0.04, 0.25, 0.25, 0.4, 0.05]$
N_u	10	λ_1	0	$q^\alpha(0)$	$[0,\ 0,\ 0.5,\ 0.5,\ 0,\ 0]$
$v(0)$	15 ± 1 m/s	λ_2	$0.05/N_u/(1+v(0)^2)$	γ	0.2
$v^{\max}$	$100/3.6$ m/s	λ_3	$0.05/N_u/a^{\max 2}$		
τ	0.5 s	λ_4	0		



(a) Approach by Broadhurst, Eidehall.

(b) Markov chain approach.

Fig. 5.35.: Autocorrelation of input trajectories.

Besides the autocorrelation, the spectral density of input signals generated by the Markov chain approach and the approach used in [32, 52, 59, 60] are compared for a deeper analysis. The spectral density describes how the energy of a signal is distributed over its frequency, where the energy of a signal $x(t)$ is defined as $\int_{-\infty}^{\infty} |x(t)|^2 \, dt$ in signal processing. The spectral density is defined as $|X(f)|^2$, where $X(f)$ is the Fourier transform of $x(t)$. For the analysis of several instances of a stochastic signal, one is interested in the expectation of the random spectral density $E[|\mathbf{X}(f)|^2]$.

Proposition 5.4 (Average Spectral Density): The average of the spectral density $E[|\mathbf{U}(f)|^2]$ for input signals $u(t)$, which are piecewise constant for time τ, is computed as

$$\Phi(f) = E[|\mathbf{U}(f)|^2] = \tau^2 \mathtt{si}^2(\pi f \tau) \sum_{k=1}^{N_u} \sum_{l=1}^{N_u} E[\mathbf{u}_{t_k} \mathbf{u}_{t_l}] e^{-j2\pi f \tau (k-l)}. \qquad \square$$

The expectation $E[\mathbf{u}_{t_k} \mathbf{u}_{t_l}]$ is obtained from the autocorrelation (5.9) and $\mathtt{si}()$ is the sinc function which is defined as $\mathtt{si}(x) = \sin(x)/x$. A proof of the proposition can be found in Appendix A.2. The expectations of spectral densities $\Phi(f)$ are visualized in Fig. 5.36 and were computed based on the parameters in Tab. 5.8. It can be seen that the lower frequencies are more dominant in the Markov chain approach, while the frequency distribution of the approach by Broadhurst and Eidehall is close to an IID process with uniform distribution. Thus, the Markov chain approach better imitates the behavior of traffic participants who are generally changing their acceleration with low frequency.

The two approaches described for modeling the acceleration command of a vehicle have not been verified with measured data from real traffic. However, the possibility to correlate input data and the stronger dominance of input signals with low frequency are arguments in favor of input trajectories generated by Markov chains. For this reason, only input trajectories generated from Markov chains are used here. The comparison of probability distributions obtained from the Markov chain and the Monte Carlo simulation approach are presented next.

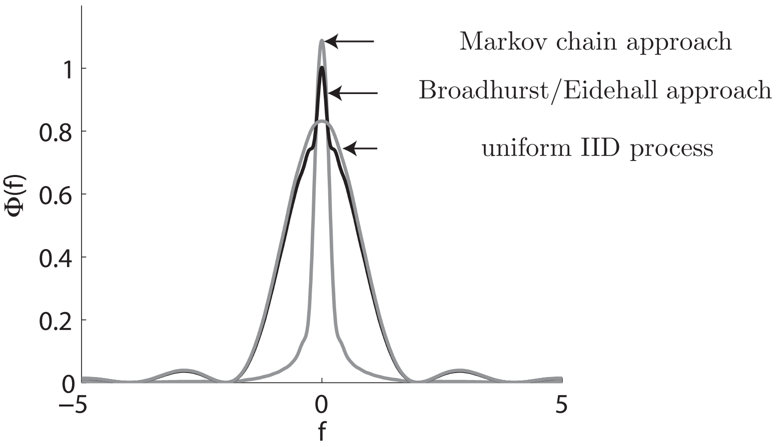


Fig. 5.36.: Average spectral density of input trajectories generated from different approaches.

5.7.4. Comparison of Probability Distributions

In this subsection, the input trajectories generated from Markov chains are used to compare the performance of the Markov chain approach with the Monte Carlo approach. The computed probability distributions are compared by the error or distance

$$d = \sum_i |p_i - p_i^{\mathrm{e}}| \cdot \mathtt{V}(\mathcal{X}_i),$$

where p^{e} is the exact probability distribution. The multiplication with the volume of the cells is required in order to compare results of different discretization. The probability distributions are compared for a braking maneuver and a maneuver where the acceleration is uncertain. In these two scenarios, three different Markov chain discretizations are used as listed in Tab. 5.9. The transition probabilities of the Markov chains are generated using Monte Carlo simulation[7]. Further, the non-zero probability reduction as suggested in Sec. 5.4.3 has been applied to the Markov chain approach with $p^{\max} = 3/(d \cdot c)$, where d and c is the number of discrete states and inputs, respectively. The Monte Carlo approach used in both scenarios is performed with 10^4 simulations.

The braking maneuver is chosen because the probability distribution of $\dot{v} = a^{\max}\,\mathbf{u}$ ($\mathbf{u} \in [\underline{u}, \overline{u}]$) (see (5.1)) has an exact solution, which is described in Example 4.1. In the braking scenario, only the velocities of the Markov chain and Monte Carlo approach

[7] 10^4 samples are generated for each cell.

are compared. The time horizon is chosen as $t_f = 5$ s and the time increment is $\tau = 0.5$ s. The acceleration command is uniform in $[-1/3, 0]$ and the initial speed is uniformly distributed within $v(0) = [17, 19]$ m/s. The exact solution, the Monte Carlo solution, and the solution of the Markov chains are compared: Markov chain A (coarse discretization) is compared in Fig. 5.37 and Markov chain B (fine discretization) in Fig. 5.38. In order to compare different discretizations, the bins of the Monte Carlo approach counting the number of samples are adjusted to the cells of the corresponding Markov chains. The computational times are shown in Tab. 5.10 and were obtained from an AMD Athlon64 3700+ processor (single core) using a Matlab implementation. The Monte Carlo simulation has been obtained using a Runge-Kutta solver for ordinary differential equations and the analytic solution presented in Prop. 5.2. One can see that the analytical solution is about 15 times faster than the Runge-Kutta solver.

Tab. 5.9.: State space discretizations for a position interval of $[0, 400]$ m and a velocity interval of $[0, 60]$ m/s.

discretization	position segments	position resolution	velocity segments	velocity resolution
A	80	5 m	30	2 m/s
B	80	5 m	120	0.5 m/s
C	320	1.25 m	120	0.5 m/s

Tab. 5.10.: Computational times of the braking scenario.

Monte Carlo (simulated)	Monte Carlo (analytical)	Markov chain A	Markov chain B
3.67 s	0.252 s	0.016 s	0.042 s

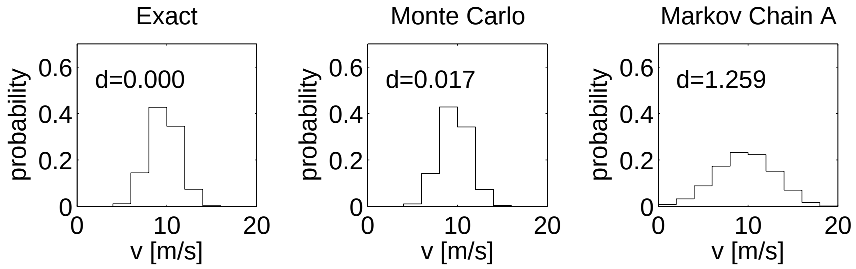


Fig. 5.37.: Braking scenario: Velocity distributions for a coarse discretization ($t = 5$ s).

In the second example, the input is generated from the input transition matrix Γ as described in Sec. 5.5.1 when following a straight road with speed limit. The parameters required to determine Γ and further parameters are listed in Tab. 5.11. The uniform initial position interval is $[2, 8]$ m and the initial velocity interval is $[15, 17]$ m/s. The resulting position and velocity distribution for a coarse discretization can be found in Fig. 5.39 and

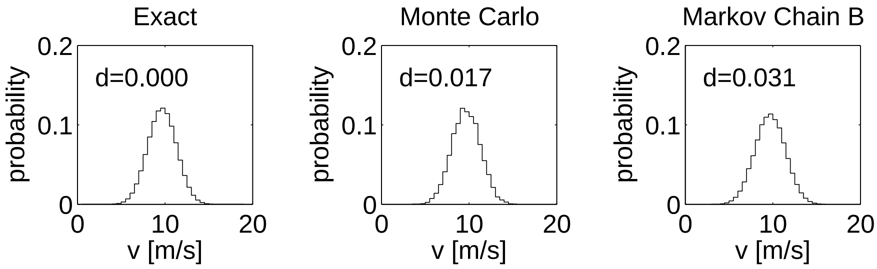


Fig. 5.38.: Braking scenario: Velocity distributions for a fine discretization ($t = 5$ s).

for a fine discretization in Fig. 5.40. Note that the Markov chain model B has a coarse discretization of the position and a fine one of the velocity so that the position result is shown in Fig. 5.39 and the velocity result in Fig. 5.40. Since there is no exact solution for this scenario, an almost exact solution was computed with Monte Carlo simulation using $5 \cdot 10^5$ samples. The computational times can be found in Tab. 5.12, which are again obtained from an AMD Athlon64 3700+ processor (single core) using a Matlab implementation. The Monte Carlo simulation has been obtained using the Runge-Kutta solver and the analytic solution as presented in Prop. 5.2. It can be observed that the Markov chain solution is faster than the analytically obtained Monte Carlo solution and that the discretization of the Markov chain C is fine enough to produce results that are more accurate than the Monte Carlo approach with 10^4 simulations.

Tab. 5.11.: Behavior parameters.

γ	0.2
μ	$\begin{bmatrix}0.01 & 0.04 & 0.25 & 0.25 & 0.4 & 0.05\end{bmatrix}$
$q_i(t=0)$	$\begin{bmatrix}0 & 0 & 0.5 & 0.5 & 0 & 0\end{bmatrix}$ $(\forall i)$
$v^{\max}$	100/3.6 m/s

Tab. 5.12.: Computational times of the road following scenario.

Monte Carlo (simulated)	Monte Carlo (analytical)	Markov chain A	Markov chain B	Markov chain C
3.44 s	0.578 s	0.030 s	0.110 s	0.417 s

Finally, it was analyzed if the quality of the probability distributions depends on the initial condition. As the vehicle model (5.1) is invariant under translations in position, it is only necessary to vary the initial velocity. The influence on the initial velocity on the distances d^{pos}, d^{vel} of the position and velocity is shown in Fig. 5.41. The Monte Carlo simulations are performed with 10^4 samples and the Markov chain approach was computed with the C model. In contrast to the previous computations, the speed limit of 100/3.6 m/s has been removed so that initial velocities above this speed can be investigated. It can be seen that the dependence on the initial velocity and thus on the initial state can be

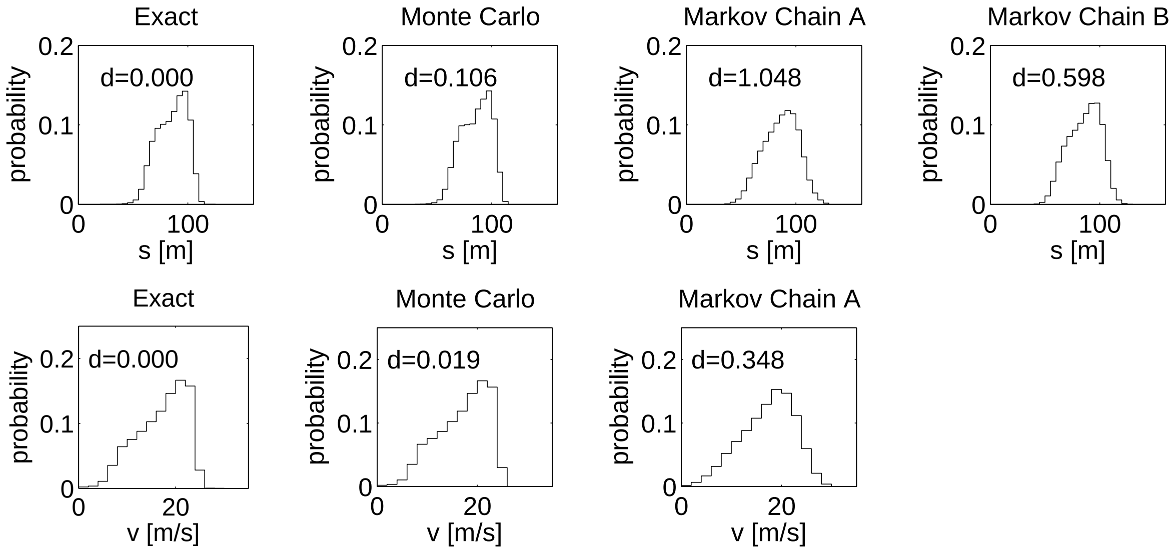


Fig. 5.39.: Road following scenario: Position and velocity distribution for a coarse discretization ($t = 5$ s).

neglected, meaning that the results in Fig. 5.39 and Fig. 5.40 are representative. In the next subsection, the crash probabilities obtained from the Monte Carlo simulation are compared to the ones obtained from the Markov chain approach.

5.7.5. Comparison of Crash Probabilities

One big advantage of Monte Carlo simulation is that the crash probability can be easily computed. When using importance sampling, it is obtained from the number of trajectories leading to a crash N^{crash} divided by the overall number of simulated trajectories: $p^{\text{crash}} = N^{\text{crash}}/N_s$. Note that trajectories causing a crash are not removed from the computation in order to obtain crash probabilities in compliance with Def. 5.1. The traffic participants are modeled by rectangles with a certain length and width. In order to efficiently determine if a crash occurs, the separating axis theorem is applied which allows the detection of intersections of rectangles with low computational costs [76]. The description of the implementation can be found in [194] and an extension considering the velocity of objects is presented in [58].

In the Markov chain approach, one has to compute the probability distribution of the traffic participants as an intermediate step and then compute the probability distribution as described in Sec. 5.6.

The crash probabilities are investigated for a scenario where the autonomous car drives behind another car. The autonomous car starts from the position 0 m with constant velocity 20 m/s and has a uniform position uncertainty of ±3 m. The vehicle driving in front has an uniform position in the interval of $[20, 25]$ m and the initial velocity is within $[15, 17]$ m/s. The other parameters are listed in Tab. 5.11. The crash probability of Markov chains is compared to the exact solution with a coarse and a fine discretization using model A and C (see Tab. 5.9). The (almost) exact solution is obtained from a Monte Carlo simulation with 10^5 samples. Besides two different Markov chain models, the two computational tech-

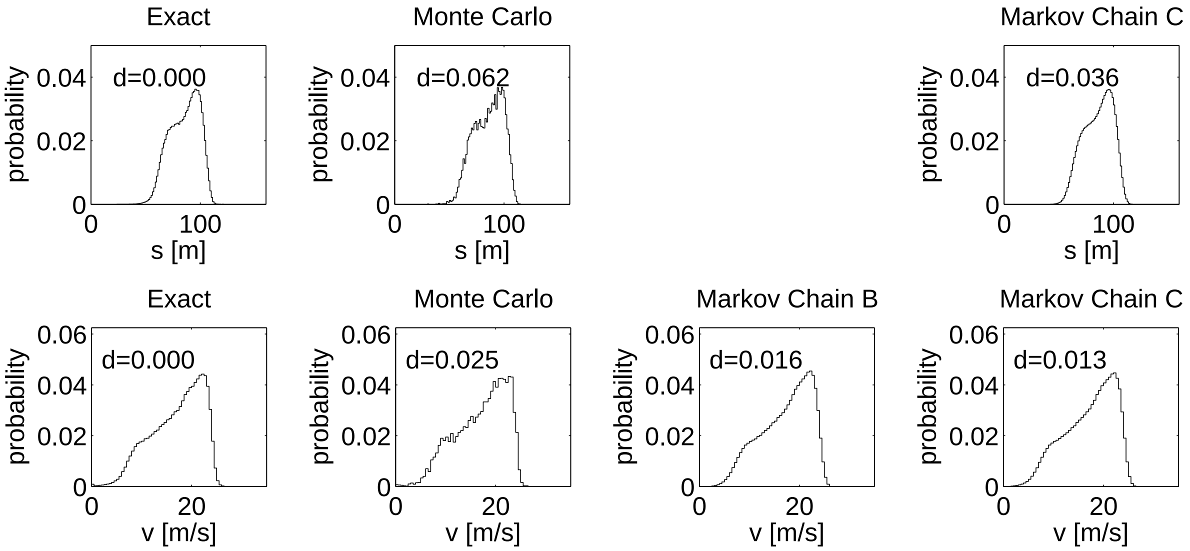
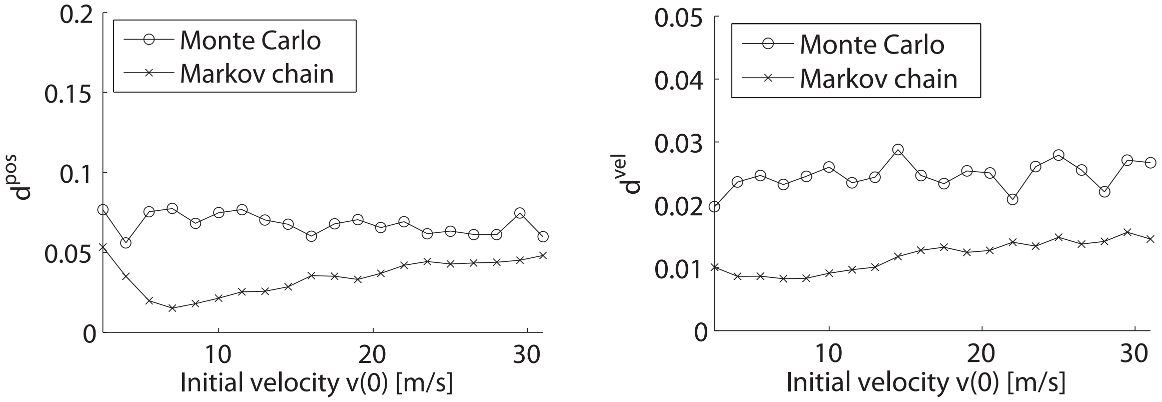


Fig. 5.40.: Road following scenario: Position and velocity distribution for a fine discretization ($t = 5$ s).



(a) Distance d of the position distribution.

(b) Distance d of the velocity distribution.

Fig. 5.41.: Distance d to the exact solution for different initial velocities.

niques for calculating the crash probability are compared: The conservative computation (con) with the relaxed computation (rel); see Sec. 5.6.1. In addition, the crash probability is computed based on the probability distribution at points in time (TP) and of time intervals (TI). This results in 4 different variations: TPcon, TIcon, TPrel, and TIrel. The crash probabilities p^{crash} for different time steps using the 4 different computing methods are shown in Fig. 5.42. It can be observed that the relaxed computation produces much better results than the conservative computation for the coarse model A. In the case of the fine model C, the conservative and relaxed computation produce similar results; however, in terms of the time interval solution the relaxed computation is slightly better. It is again remarked that only the time interval solution ensures that no dangerous situation is missed.

Besides different Markov chain models, Monte Carlo solutions were tested, too. Fig. 5.42(c) shows that the result is very accurate, even when only 10^3 or 10^2 samples are used. For this reason, it can be clearly stated that the Monte Carlo simulation performs better than the Markov chain approach when the crash probability has to be computed. This is reconfirmed by the computational times in Tab. 5.13, where the Monte Carlo approach is more efficient. The computational times for the Markov chain approach are separated into the part for computing the probability distribution and the part that intersects the probability distributions to obtain the crash probability. The computations were performed on an AMD Athlon64 3700+ processor (single core) using a Matlab implementation.

Tab. 5.13.: Computational times of the crash scenario.

Markov chain				
	A (TP)	A (TI)	C (TP)	C (TI)
Prob. dist.	0.175 s	0.175 s	0.525 s	0.525 s
Intersection	0.042 s	0.107 s	0.169 s	0.394 s
Total	0.217 s	0.282 s	0.694 s	0.919 s
Monte Carlo simulation				
	1e2 (sim.)	1e3 (sim.)	1e2 (analy.)	1e3 (analy.)
Total	0.190 s	0.549 s	0.069 s	0.321 s

5.7.6. Examination of Interacting Vehicles

The interaction of two vehicles driving in the same lane has been addressed for the Markov chain approach in Sec. 5.5.3. The formalism to compute the constraint vector η of the following vehicle is described in Prop. 5.3, which is presented again for better readability:

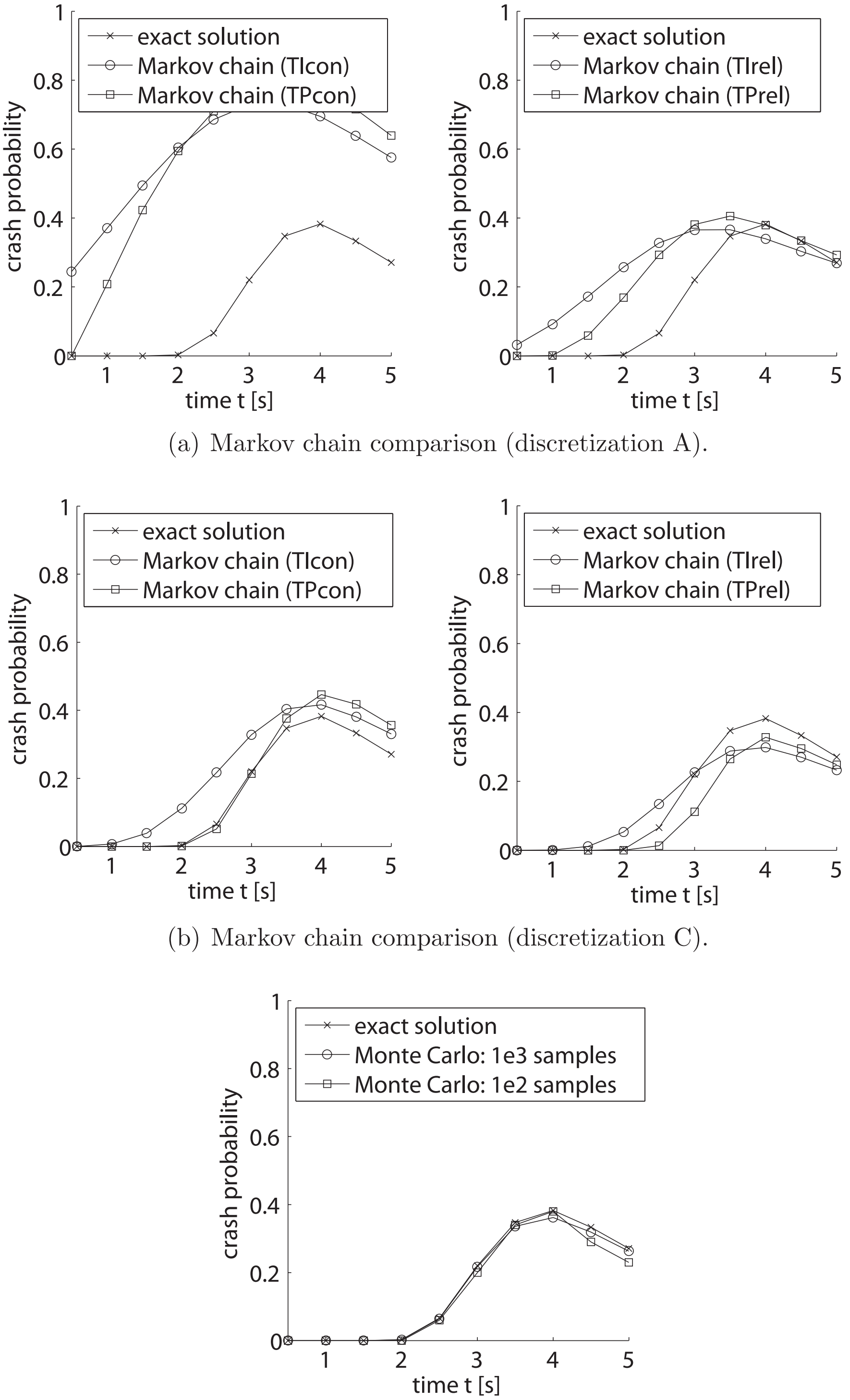
$$\eta_i^\alpha = \sum_{j,\beta} \Theta_{ij}^{\alpha\beta} p_j^{L\beta}. \tag{5.10}$$

In this formula, the constraint value η_i^α for a single state i and input α of the following vehicle is based on all states j and inputs β of the leading vehicle with probability distribution $p_j^{L\beta}$, causing an averaging effect. Another problem is that the above formula only holds if the joint probability $P(\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ of the following vehicle is independent of the one of the leading vehicle $P(\mathbf{z}^L = j, \mathbf{y}^L = \beta)$; see Prop. 5.3.

In the Monte Carlo approach, the independence assumption is not required. After introducing the events $A = (\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ and $B_j^\beta = (\mathbf{z}^L = j, \mathbf{y}^L = \beta)$ from Prop. 5.3, the modified computation of the constraint vector is

$$\eta_i^\alpha = P(C|A) = \frac{P(C,A)}{P(A)} = \frac{\sum_{j,\beta} P(C|A, B_j^\beta) P(A, B_j^\beta)}{P(A)} = \frac{\sum_{j,\beta} \Theta_{ij}^{\alpha\beta} p_{ij}^{F,L\alpha\beta}}{p_i^{F\alpha}}, \tag{5.11}$$

where $p_{ij}^{F,L\alpha\beta} = P(\mathbf{z}^F = i, \mathbf{y}^F = \alpha, \mathbf{z}^L = j, \mathbf{y}^L = \beta)$. In the Monte Carlo approach, this probability is 1 for the cell indices the samples are located in, whereas the remaining values are 0. Thus, the Monte Carlo approach does not suffer from the independence assumption.



(a) Markov chain comparison (discretization A).

(b) Markov chain comparison (discretization C).

(c) Monte Carlo comparison.

Fig. 5.42.: Crash probabilities for different points in time.

It is also not suffering from the averaging effect since (5.11) is evaluated separately for each sample. The computation in (5.11) is not possible for the Markov chain approach since the probability $p_{ij}^{F,L^{\alpha\beta}}$ is not available. It would be available if the state space of the

Markov chain approach would be extended such that there are two dimensions (position and velocity) for one vehicle and the same two dimensions for the second vehicle. However, the number of dimensions would increase by 2, which results in an explosion in the number of discrete states due to the exponential growth of discrete states.

The consequences of the averaging effect and the independence assumption are investigated by emulating the behavior of the Markov chain approach by a Monte Carlo implementation. This is done by computing the constraint vector as described in (5.10), where the probability $p_j^{L^\beta}$ is 1 for the cell indices the sample is located in and 0 otherwise. This result is compared to the exact result obtained from Monte Carlo simulation of (5.11), the result obtained from computing without interaction, and the result obtained when removing crashed samples in the Monte Carlo approach. The removal approach is performed by computing without any interaction and then removing pairs of crashing vehicles.

All results were obtained for a scenario with a leading and a following vehicle on a straight road. The parameters for the simulation are shown in Tab. 5.11, except that the speed limit is chosen to $v^{\max} = 16$ m/s. For the Markov chain computations, the discretization presented in Tab. 5.2 is used. The initial position and velocity of the following vehicle is uniformly distributed within $s^F(0) \in [2, 8]$ and $v^F(0) \in [10, 12]$ m/s. The initial intervals of the leading vehicle are $s^L(0) \in [12, 18]$ and $v^L(0) \in [10, 12]$ m/s. The samples of the different Monte Carlo simulations can be seen in Fig. 5.43, where the diagonal line[8] indicates when the following vehicle has crashed into the leading vehicle so that all samples above the diagonal line are collision-free. The figures show 10^3 samples at time $t = 5$ s. It can be observed that the emulated Markov chains contain a lot of crashed vehicles. This is mainly caused by the averaging effect and the independence assumption. A further emulation which investigates the error due to the independence assumption while fixing the averaging effect can be found in [194]. There are also some accidents in the exact solution because the stored values in $\Theta_{ij}^{\alpha\beta}$ are approximately computed so that not all trajectories starting in the corresponding cells are guaranteed to be collision-free; see Sec. 5.5.3. Clearly, in the sample removal approach, no accidents occur. The disadvantage of the sample removal method is that only accidents within the time horizon are canceled while a certain constellation of vehicles states might inevitably lead to a collision in the future.

The resulting marginal probability distributions of each vehicle are shown in Fig. 5.44 obtained with 10^4 samples after $t = 5$ s. Indeed, the Monte Carlo emulation resembles the probability distribution of the Markov chain approach. The plots also show that the intersection of probability distributions of the position does not necessarily imply that vehicles are crashing since e.g. the sample removal approach is collision-free. Thus, the approach for computing the collision probability out of the probability distribution of traffic participants as shown in Sec. 5.6 cannot be applied to two arbitrary traffic participants. However, it can be used to obtain the collision probability of the autonomous car with another vehicle, because their probability distributions are independent; see Def. 5.1.

The Monte Carlo simulations allowed a deeper insight into the advantages and disadvantages of the Markov chain approach which are discussed below.

[8] The diagonal line is shifted by the vehicle length of 5 m.

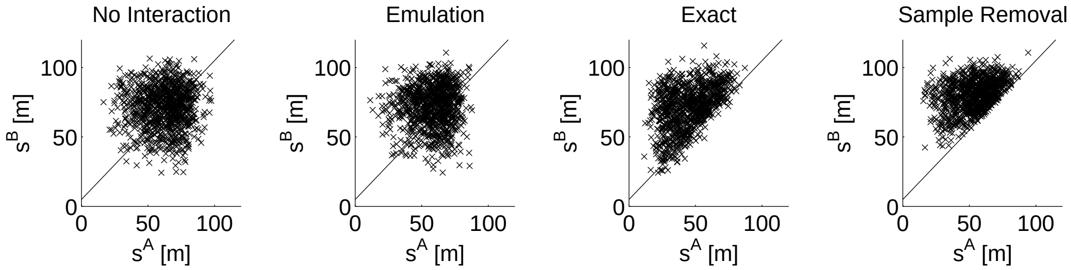


Fig. 5.43.: Joint position distribution of samples after $t = 5$ s.

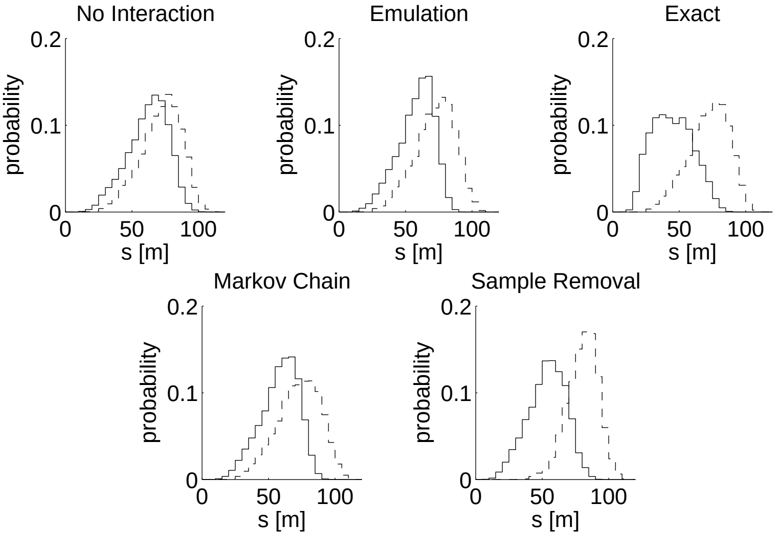


Fig. 5.44.: Marginal position distributions after $t = 5$ s.

5.7.7. Discussion

The Markov chain approach and the Monte Carlo approach have some inherent differences concerning their error sources. The main error in the Markov chain approach is introduced due to the discretization of the state and input space. The error in the transition probabilities can be made arbitrarily small since they are computed beforehand. Thus, the Markov chain approach has only systematic errors from the discretization but no stochastic errors since no random sampling is applied.

In the Monte Carlo approach, there are no systematic errors (no bias) because each simulation is correctly solved with the original dynamical system equations. However, the Monte Carlo simulation suffers under stochastic errors due to the sampling of the initial conditions and input sequences. Due to the stochastic errors, the resulting distributions and crash probabilities differ from execution to execution although the initial conditions and inputs are unchanged. This implies that the obtained results might be far off the exact solution – however, the likeliness of an extremely bad result is small and the mean error converges with $\frac{1}{\sqrt{N_s}}$, where N_s is the number of samples.

For a simple scenario, where a vehicle drives along a road, the resulting probability distributions of the Markov chain approach are slightly more accurate and faster than for the Monte Carlo simulation if an analytical solution exists. When no analytical solution exists, the Markov chain approach is at least about 10 times faster. Because there are many matrix multiplications in the Markov chain approach, it can be significantly accelerated by using dedicated hardware such as DSPs (digital signal processors). However, when computing crash probabilities, the Monte Carlo approach clearly returns better results since it does not suffer from the discretization of the state space.

Another disadvantage of the Markov chain approach is that it is difficult to find an algorithm which accurately computes the probability distributions for interacting vehicles – a good solution is still to be found. For instance, one could investigate if it makes sense to remove probabilities in the Markov chain approach as it is done in the Monte Carlo approach when removing crashed samples. The errors in the current implementation could be explained by an emulation using Monte Carlo simulation, which shows that the Monte Carlo approach is more flexible. This might be of interest in unstructured environments or when there is no single plausible path for a traffic participant. In such cases, one could mix Monte Carlo related approaches with the Markov chain approach.

This is illustrated in Fig. 5.45, showing another vehicle which approaches a standing vehicle. This vehicle might wait behind the standing vehicle or pass by. There are three plausible paths in Fig. 5.45 and the partition of each path originating from the path and deviation segments is visualized. Path 1 represents the option that the vehicle waits and there are two alternatives paths 2 and 3 for passing the standing vehicle. When one is interested in the probability distribution of other vehicles to e.g. plan a safe trajectory, the more accurate and efficient Markov chain approach should be applied along these paths. If one is only interested in the probability of a crash, Monte Carlo simulation should be applied along the paths. However, Monte Carlo simulation should not be directly applied in 2-D scenarios since many samples end up at road borders or other obstacles, making this approach not very efficient. This is demonstrated for example in [194] and addressed in many publications, e.g. [15, 31, 32, 52, 59, 60].

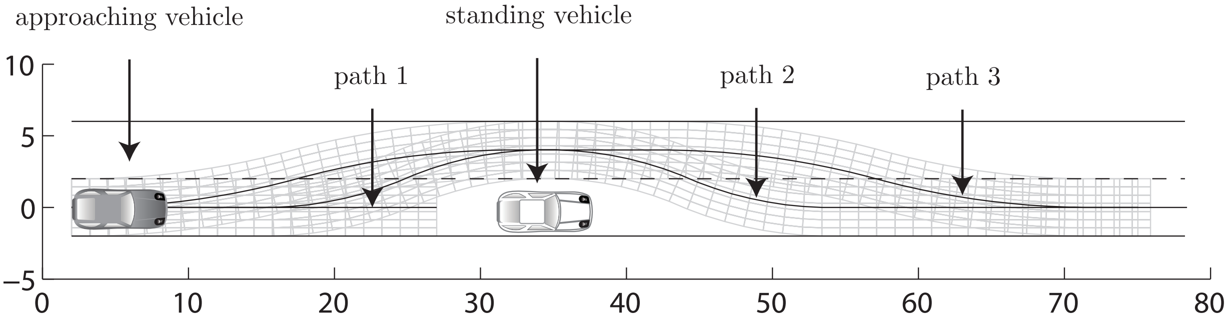


Fig. 5.45.: Evasion of a standing car with alternative paths.

5.8. Driving Experiment

In previous sections, the safety assessment framework for autonomous cars was demonstrated by numerical examples. In this section, the safety assessment is conducted based on real measurements from a driving experiment with the experimental vehicle *MUCCI* (<u>Mu</u>nich's <u>C</u>ognitive <u>C</u>ar <u>I</u>nnovation) [192, 198]. This vehicle has been converted from a standard Audi Q7 model to an autonomous vehicle within the research project *Cognitive*

Automobiles [154].

5.8.1. Experimental Vehicle MUCCI

The experimental vehicle *MUCCI* is equipped with several sensors in order to properly perceive the environment. The sensors used for the driving experiment described in this work are:

- Inertial measurement unit (IMU): This device measures the translational and rotatory acceleration of the vehicle. The rotatory accelerations are used to adjust the active camera platform such that it is aligned with the horizon, despite the roll and pitch of the vehicle.
- Active camera platform: The cameras are used to detect lane markings on the road which allows a representation of the road geometry to be built without any information from a navigation system with road network information. The shape of the lane markings is modeled by clothoids – a representation that has also been investigated for modeling possible paths of other traffic participants [197].
- Light detection and ranging (LIDAR): Other objects in the traffic scene are detected by this device, which measures the reflections of transmitted laser beams. After some preprocessing, the enclosing circle radius, the relative position, and the relative velocities of other traffic participants are obtained [161].

This equipment is also used in other vehicles of the *Cognitive Automobiles* project such as for *MuCAR-3*, developed by the University of the Bundeswehr München, and for a modified VW Passat developed by the Karlsruhe Institute of Technology. The VW Passat participated in the 2007 Darpa Urban Challenge [92]. Besides similar sensor technologies, all vehicles share a common computer hardware and software framework [172]. The software is organized around the real time database *KogMo-RTDB* [74, 75]. This centralized architecture has shown great benefits, since data produced by one software module is often used by several other modules. For example, the pose and velocity of other traffic participants is used by the vision software, the trajectory planner, and the safety assessment module.

Data is written into and read from the real time database by a C or C++ interface. The results of the safety assessment algorithms previously presented in Sec. 5.4-5.7 were implemented in Matlab. In order to use the algorithms within the software framework of the autonomous car, the code has been rewritten using the C++ language. Analogously to the Matlab implementation, the fact that the transition matrices of the Markov chains are sparse and that most values of the probability vector are 0 has been taken advantage of.

5.8.2. Scenario

The algorithms developed for the safety assessment of the autonomous vehicle have been tested with sensor information recorded at the University of the Bundeswehr München in Neubiberg, Germany. An aerial image of the test location, on which the driven road section is highlighted, can be found in Fig. 5.46. The test scenario was driven with two autonomous cars and a human-driven car as depicted in Fig. 5.46.

At the beginning of the scenario, the human-driven car is the leading vehicle, followed by *MuCAR-3*, which in turn is followed by *MUCCI* (see location ① and ②). At position ③, the autonomous vehicle *MuCAR-3* decides to overtake the human-driven car. During the overtaking maneuver, the autonomous car *MUCCI* closes the gap to the human-driven car (location ④). The scenario is finished as soon as *MuCAR-3* has successfully ended its overtaking maneuver at location ⑤.

The safety assessment has been performed for *MUCCI*, but could in principle be ported to *MuCAR-3* due to the common hardware and software framework. However, the porting would still require several modifications because the software modules reading and writing to the common real time database differ from vehicle to vehicle. Screenshots of the video image of *MUCCI* and the graphical user interface (GUI) of the safety assessment software are shown in Fig. 5.47 at different locations marked in Fig. 5.46. The GUI is not directly connected to the safety verification algorithms, but gathers the displayed information from the real time database. Data that is written from the safety verification module to the real time database include the predicted probability distributions of other traffic participants, the computation time for the prediction, and the crash probabilities over time.

The computation time was about 0.1 s when one vehicle was detected and about 0.15 s when two vehicles were detected using the computer hardware described in [75]. The prediction horizon was fixed to 5 s such that the prediction was about 30-50 times faster than real time.

The main purpose of this test drive was to demonstrate the interaction with other software modules connected through the real time database. Another aspect was to show the real time capabilities of the used algorithms. However, due to the not yet fully achieved capability of driving in real traffic, the compliance of the predictions with real traffic behavior could not be tested. This shortcoming and possible future work for handling more complicated scenarios is discussed next.

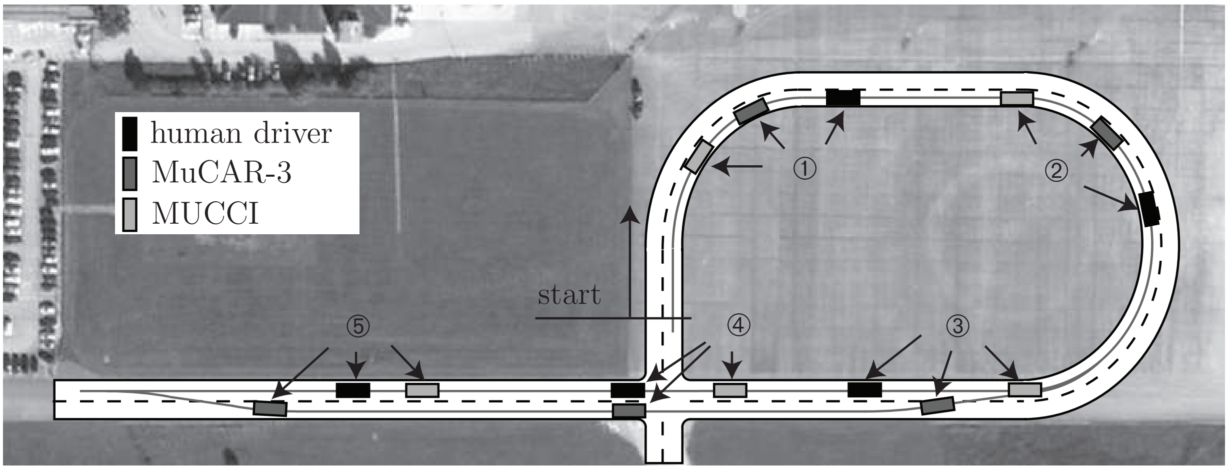
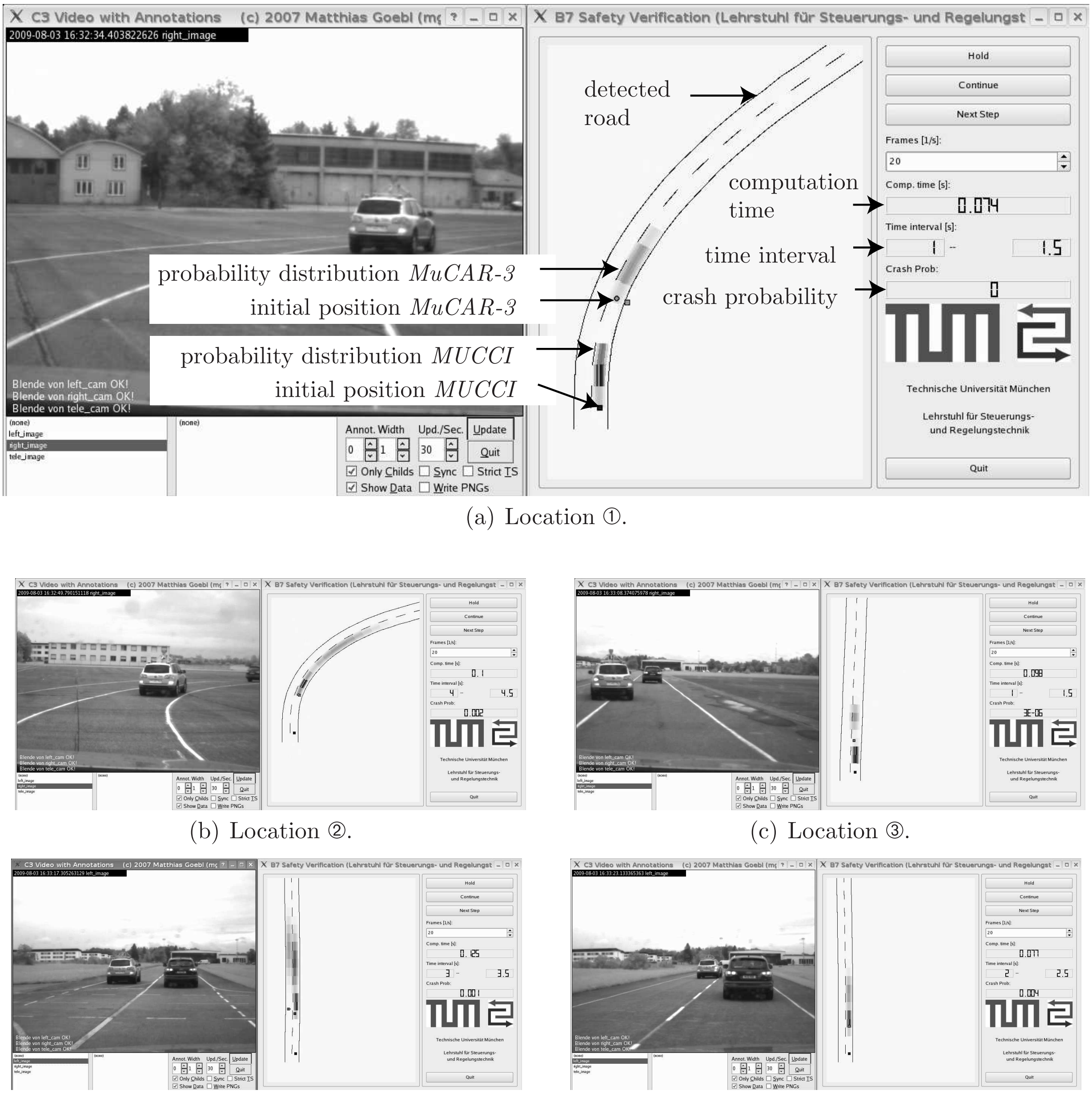


Fig. 5.46.: Test drive scenario.


C3 Video with Annotations (c) 2007 Matthias Goebl (m
B7 Safety Verification (Lehrstuhl für Steuerungs- und Regelungst
detected road
computation time
time interval
crash probability
probability distribution *MuCAR-3*
initial position *MuCAR-3*
probability distribution *MUCCI*
initial position *MUCCI*
Hold
Continue
Next Step
Quit
Update
Technische Universität München
Lehrstuhl für Steuerungs- und Regelungstechnik
Blende von left_cam OK!
Blende von right_cam OK!
Blende von tele_cam OK!

2009-08-03 16:32:34.403822626 right_image

Annot. Width: 0, 1; Upd./Sec.: 30

☑ Only Childs ☐ Sync ☐ Strict TS

☑ Show Data ☐ Write PNGs

Frames [1/s]: 20

Comp. time [s]: 0.074

Time interval [s]: 1 – 1.5

Crash Prob: 0

(a) Location ①.

(b) Location ②.

(c) Location ③.

(d) Location ④.

(e) Location ⑤.

Fig. 5.47.: Screenshots of the test drive scenario.

5.9. Summary

A framework for the safety assessment of autonomous vehicles with respect to the planned driving actions has been developed. Thereby, the whole chain – consisting of modeling (vehicle model, stochastic behavior model), computational realization (Markov chain abstraction, crash probability computation), comparison to alternative approaches (Monte Carlo simulation), and testing (test drive of a prototype vehicle) – has been processed.

In the modeling section, the concept has been introduced that each traffic participant has one or several paths to be followed. Due to the concept of defining possible paths for each vehicle, the motion along a path can be described by the position and the velocity only. The deviation probability distribution along the paths is modeled time invariant, except for lane change maneuvers. A dynamic adaption of the deviation probability distribution is subject to future work. The probability distribution of the path coordinate is obtained from an abstraction of traffic participants to Markov chains. Since the longitudinal dynamics of traffic participants consists of position and velocity only, the abstraction to Markov chains is computationally feasible.

A model considering arbitrary motions on a two-dimensional plane would be too high-dimensional for a practical Markov chain abstraction. However, Monte Carlo simulations also suffer from computational complexity in fully two-dimensional scenarios. When a single path does not sufficiently describe a situation, multiple paths could compensate for the shortcomings of the path concept; see Fig. 5.45. Alternative paths, which are not generated from the center of possible lanes could be obtained by Monte Carlo simulation or optimization techniques with alternative cost functions, realizing different types of drivers (e.g. safe or sporty drivers). The advantage of multiple paths with deviation probability is that many fewer paths have to be considered than in a pure Monte Carlo simulation of a two-dimensional scenario.

A particularity of the Markov chain abstraction for the longitudinal vehicle dynamics is that the abstraction does not have to be complete. The completeness can be achieved by additional simulations of the vehicle dynamics which result in lower and upper bounds of positions and velocities. Since the Markov chain computations can be incomplete, small probabilities for certain state and input values are canceled, followed by a normalization of the remaining probability values. This procedure allows fewer probabilities to be stored and further reduces the computational burden due to the sparse representations of probability values.

The stochastic acceleration values of other traffic participants have been modeled by another Markov chain. The transition probabilities of the acceleration values are based on simulations and heuristics. A learning approach which directly incorporates recorded traffic data would be a reasonable alternative, which becomes realizable when the detection of traffic participants and their tracking is more reliable in the future. For the heuristic approach, different stochastic models for typical patterns such as road following, vehicle following, intersection crossing, and lane changing have been suggested. Other behaviors that do not fall into these categories are handled by the unstructured environment approach used for a mobile robot in human-populated areas [195].

The computed probability distributions of the Markov chains are the basis for the computation of the crash probability. In this work, the probability of a crash is computed under the assumption that the autonomous vehicle has not crashed until the investigated point in time (conditional crash probability). This has the advantage that the crash probability of a particular time interval can be assessed independently from previous occurrences. Computational techniques for the efficient computation of the conditional crash probability have been presented.

The obtained probability distributions and crash probabilities have been compared to a Monte Carlo implementation. This comparison not only reveals implementation errors, but also allows the strengths and weaknesses of the Markov chain abstraction to be assessed. One of the main advantages of the Markov chain abstraction is the better efficiency in computing probability distributions, especially when no analytical solution is available for the vehicle model. The main disadvantages of the Markov chain approach compared to the Monte Carlo simulation are that it computes worse crash probabilities and is less flexible. The worse flexibility is especially evident when the interaction between traffic participants is considered.

It is finally remarked that the probability distribution of other vehicles could not only be used to estimate the threat of a planned trajectory, but might also be used by the trajectory planner for a goal-oriented adaption of existing trajectories to safer trajectories.

6. Conclusion and Future Directions

6.1. Conclusion

New solutions for problems in classical reachability analysis, stochastic reachability analysis, and for the safety assessment of autonomous vehicles have been presented.

Reachability Analysis

In classical and stochastic reachability analysis, the major problem is not finding algorithms that are able to solve reachability problems, but to find algorithms that scale well with the dimension of the continuous state space. For this reason, reachable sets are represented by zonotopes in this work. Zonotopes have outperformed previous set representations for the reachability analysis of linear time invariant (LTI) systems with uncertain input. However, the use of zonotopes for more complex systems, such as linear systems with uncertain parameters, nonlinear systems, and hybrid systems, is rather scarce. The presented approaches for these new frontiers of reachability analysis with zonotopes are all based on the computational techniques for LTI systems. This has the advantages that on a lower level, the superposition principle can be applied and that zonotopes are mapped to zonotopes from one time interval to the other. The scalability of the presented approaches decreases in general in the following order: Linear systems with uncertain parameters, nonlinear systems, hybrid systems. The algorithm for linear systems with uncertain parameters can be applied to systems with 100 or more continuous state variables without any difficulty. The only problem is that due to the wrapping effect, the reachable set might grow to infinity even though the system is stable. In the case of nonlinear systems, the applicability of the proposed algorithm strongly depends on the degree of nonlinearity. For mild systems, up to 100 continuous state variables are also possible. However, when much splitting of the reachable set is involved, only small systems can be handled. Similar considerations apply to hybrid systems. When few switchings of the continuous dynamics have to be considered, quite large problems can be handled. However, in the case of many switchings, the reachable set suffers from inappropriate over-approximations. If a system has few continuous state variables and many guard sets, the use of polytopes is preferable since the intersection of polytopes with polytopial guard sets yields polytopes so that no additional over-approximation has to be accepted.

Stochastic Reachability Analysis

Stochastic reachability algorithms suffer even more from the curse of dimensionality since most approaches rely on the discretization of the continuous state space. In order to address this problem, the system class has been limited to linear systems for one of the proposed approaches. For linear systems with Gaussian white noise, an analytical solution exists for the computation of probability distributions. This solution has been extended to Gaussian white noise whose mean is uncertain within a zonotope. Since no probability distribution is known for this uncertainty, the enclosing hull of all possible probability distributions is computed – for points in time and time intervals. The approach is scalable and can be used for systems with 100 or more continuous state variables. In addition, the algorithm does not suffer from the wrapping effect. It is again remarked that in this thesis, stochastic reachable sets are used as a synonym for probability distributions of the state, which differs from the definition in most other works. The duality of both definitions when the set of unsafe states is absorbing has been addressed. The second presented approach is the Markov chain abstraction, which allows stochastic reachable sets to be computed for general systems, but with only a few continuous state variables, because this approach uses state space discretization. The specialty of the proposed Markov chain abstraction is that the time increment can be set independently of the cell geometry, allowing tune the update rate and the discretization resolution to be independently tuned for an efficient update of the Markov chain. This is especially useful when the Markov chain has to be executed online for a time critical safety analysis, while the abstraction can be computed offline.

Safety Assessment of Autonomous Cars

These online capabilities are of use for the safety assessment of autonomous cars so that Markov chains are applied to this problem. In order to update the safety assessment with high frequency for long prediction horizons, the prediction of traffic participants has to be several times faster than real time. With the assumption that other traffic participants follow paths up to a certain accuracy, only the position and velocity have to be discretized, resulting into Markov chains with an appropriate number of discrete states. Besides the Markov chain which models the physical behavior of a vehicle when an acceleration command is given, a stochastic driver model has to be created which generates appropriate probability distributions for the acceleration command. The presented driver models found in other works on safety assessment have low correlations between the generated inputs and offer an unfavorable spectral density. This is different for the proposed Markov chain approach of input values, and has the advantage of being consistent with the Markov chain model of the physical behavior. A specialty of the safety assessment of autonomous vehicles is that the set of unsafe states is represented by the position occupation of other traffic participants, which is time varying and stochastic. Efficient computational techniques have been presented to compute the intersection probability with such time varying and stochastic sets, resulting in the crash probability of the considered time interval. The presented Markov chain approach has been compared to Monte Carlo simulation and has been implemented in an autonomous prototype vehicle. In terms of computing the probability distributions, the Markov chain approach showed better results, while the Monte Carlo simulation is superior when computing crash probabilities. In the following section, future directions of all considered research areas are discussed.

6.2. Future Directions

Future directions are separately discussed for classical reachability analysis, stochastic reachability analysis, and safety assessment of autonomous cars.

Reachability Analysis

For linear systems, there already exist reachability algorithms which show good performance. However, an algorithm for linear systems which only requires a start button to be pushed is not yet available. For this open goal, it is necessary to automatically adapt important parameters such as the time step size. Another promising aspect is to develop an algorithm for linear systems with uncertain parameters when the parameters are time varying.

There are also many promising directions for nonlinear systems when using the conservative linearization approach as suggested in this work. A problem of this approach is that the splitting of zonotopes is inefficient or leads to unsatisfactory over-approximations. This problem can be alleviated by not splitting zonotopes $\mathcal{Z}$, but computing with families of zonotopes such that the sets A and B enclose a zonotope $A \cup B \supseteq \mathcal{Z}$ and further define the split zonotopes by $\mathcal{Z}_1 = A \cap \mathcal{Z}$ and $\mathcal{Z}_2 = B \cap \mathcal{Z}$. Another area of future work is to efficiently unify zonotopes to zonotopes after many splits so that one can compute with fewer zonotopes in certain state space regions. One of the major sources for the over-approximation is the conservative computation of the set of linearization errors using interval arithmetics. This could be improved by advanced interval arithmetics, which takes advantage of monotonicity in Lagrange remainders as shown in [145], or by Taylor models for Lagrange remainders as suggested in [23].

There is also much potential for the proposed algorithms for hybrid systems, e.g. by improving the conversion of reachable sets from generator representation to halfspace representation and vice versa. One could develop improved order reduction techniques of zonotopes for a more efficient conversion to halfspace conversion. From this development, the reachability algorithms for linear time varying and nonlinear systems would also benefit. For the conversion to generator representation, one could try optimization methods which optimally adapt the shape of enclosing parallelotopes or zonotopes. The technique for the enclosure could also be changed from enclosing H-polytopes instead of V-polytopes, because H-polytopes have a more compact representation. The computational demand for hybrid systems could be decreased by using decomposition methods so that it is sufficient to verify smaller subsystems [63]. It is also promising to parallelize algorithms so that they are more attractive for multi-core computer architectures.

Stochastic Reachability Analysis

Two approaches for stochastic reachability analysis have been investigated. One of them is the Markov chain approach, for which there is not much potential for improvement since it is naturally limited by the discretization of the state space, allowing only problems with a few continuous state variables to be solved.

There is much more potential for the enclosing hull approach, which over-approximates the probability distribution of stochastic systems. It has been shown that this approach scales well for linear systems. However, there is much potential for finding tighter enclosing hulls of linear systems. Besides an improved accuracy, one could think of extending this concept to nonlinear and hybrid systems. The extension to nonlinear systems could be accomplished by conservative linearization, analogously to the concept used for classical reachability analysis of nonlinear systems. A possible representation of enclosing hulls for this problem class could be mixed Gaussian distributions whose centers are uncertain within sets. An extension to hybrid systems is an even bigger challenge due to the intersection with guard sets, resulting in probability distributions which might even be inappropriately over-approximated by a combination of mixed Gaussian distributions with sets.

Safety Assessment of Autonomous Cars

In terms of the safety assessment of autonomous cars, there are a number of ideas that have not yet been realized. First of all, there are concepts from other publications that are not integrated into the current framework. One concept is the selection of relevant traffic participants by first checking if their reachable sets intersect [77], which can be easily realized using Prop. 5.1. The other concept is to integrate visibility considerations which provide a probability that a traffic participant reacts to another traffic participant in front, to the left/right, or behind [60]. In this work, traffic participants only react to vehicles driving in front, for which the inattentiveness probability ϵ has been introduced.

A further aspect is the generation of possible paths of traffic participants. Up to now, the paths have been generated according to the center line of drivable lanes. However, this approach is not suitable if, for example, a lane is blocked. In this case one has to generate possible paths to circumvent the obstacle, as shown in Fig. 5.45. These paths could be obtained by Monte Carlo simulation or optimization techniques with alternative cost functions, realizing different types of drivers (e.g. safe or sporty drivers). The advantage of multiple paths with deviation probability is that many fewer paths have to be considered than in a pure Monte Carlo simulation of a two-dimensional scenario.

The behavior of traffic participants circumventing obstacles could also be learned from traffic observations. In addition, one could learn patterns for parking, lane changing, and so on. From this follows that after the correct identification of a motion pattern, the prediction can use this pattern instead of more general assumptions for driving behavior. Besides the learning of motion patterns, one could also learn the typical (possibly time varying) deviation probability when a certain pattern is followed. Another aspect, which may be better learned than modeled, is the interaction between traffic participants, for example when following another vehicle or changing lane. Even for the simple case of vehicle following, a satisfying interaction model for Markov chain computations is still to be found.

Another issue is the automatic adaption of the time horizon. A good guideline would be a time horizon which covers the duration of the planned maneuver followed by a maneuver which can bring the vehicle to a safe state. The second maneuver is not executed but added in order to ensure the possibility of reaching a safe state, such as standstill in an allowed road section. The probability of reaching a safe state after a maneuver is planned is also addressed in [177].

Additionally, one could incorporate the velocity and the relative angle of traffic participants into the crash probability computation so that the severeness of a crash is also considered for the safety assessment.

Finally, the problem of verifying the safety of an autonomous car in coordinated maneuvers can be further addressed. In a scenario, where other plans are known, the uncertain movement along the planned trajectories is small enough such that non-probabilistic methods can be applied [178].

A. Proofs

A.1. Over-approximation of the Reachable Set due to Inputs

The reachable set due to inputs $\mathcal{P}^{\mathcal{R}}$ can be approximated by assuming constant inputs within smaller time intervals $[t_{i-1}, t_i]$, where $0 = t_0 < t_1 < \ldots < t_{l-1} < t_l = r$.

$$\mathcal{P}^{\mathcal{R}}(r) \approx \int_{t_0}^{t_1} e^{A(t_1 - t)}\, dt\, \mathcal{U} + \int_{t_1}^{t_2} e^{A(t_2 - t)}\, dt\, \mathcal{U} + \ldots + \int_{t_{l-1}}^{t_l} e^{A(t_l - t)}\, dt\, \mathcal{U}. \tag{A.1}$$

In order to obtain not only an approximation, but an over-approximation, the solution for a time interval $[t_{i-1}, t_i]$ is rewritten as a finite Taylor series (see (3.2)):

$$\begin{aligned}
&\int_{t_{i-1}}^{t_i} e^{A(t_i - t)}\, dt\, \mathcal{U} \\
&\subset \Big\{\Big[It + \tfrac{A}{2!}t^2 + \ldots + \tfrac{A^\eta}{(\eta+1)!}t^{\eta+1}\Big]\Big|_{t_{i-1}}^{t_i} + \textstyle\int_{t_{i-1}}^{t_i} \mathcal{E}(t)\, dt\Big\}\mathcal{U} \\
&= \Big\{\Big[I(t_i - t_{i-1}) + \tfrac{A}{2!}(t_i^2 - t_{i-1}^2) + \ldots + \tfrac{A^\eta}{(\eta+1)!}(t_i^{\eta+1} - t_{i-1}^{\eta+1})\Big] + \textstyle\int_{t_{i-1}}^{t_i} \mathcal{E}(t)\, dt\Big\}\mathcal{U} \\
&\subseteq I(t_i - t_{i-1})\mathcal{U} + \tfrac{A}{2!}(t_i^2 - t_{i-1}^2)\mathcal{U} + \ldots + \tfrac{A^\eta}{(\eta+1)!}(t_i^{\eta+1} - t_{i-1}^{\eta+1})\mathcal{U} + \textstyle\int_{t_{i-1}}^{t_i} \mathcal{E}(t)\, dt \cdot \mathcal{U}.
\end{aligned} \tag{A.2}$$

Note that the multiplication of the input set $\mathcal{U}$ with the Taylor terms causes an over-approximation since $C \cdot \mathcal{U} + D \cdot \mathcal{U} \supseteq (C + D) \cdot \mathcal{U}$, where $C, D \in \mathbb{R}^{n \times n}$. This can be easily checked for $C = 1, D = -1$ yielding the set $\{0\}$ for the exact solution and the Minkowski addition of C and $-D$ otherwise.

It remains to replace the integral of $\mathcal{E}(t)$ which is computed in (3.3) as $\mathcal{E}(t) = [\mathbf{-1}, \mathbf{1}] \cdot \phi(t)$, where $[\mathbf{-1}, \mathbf{1}]$ is a matrix whose elements are intervals $[-1, 1]$ and $\phi(t) = \frac{(\|A\|_\infty t)^{\eta+1}}{(\eta+1)!} \frac{1}{1-\epsilon}$. Since $\phi(t) \in \mathbb{R}^+$ is monotone in t, it follows that one can over-approximate the integral by $\int_0^r \phi(\tau) d\tau \subset \phi(r) \cdot r$ so that one obtains the over-approximation

$$\int_{t_{i-1}}^{t_i} \mathcal{E}(\tau) d\tau \subset [\mathbf{-1}, \mathbf{1}][\phi(t) \cdot t]\Big|_{t_{i-1}}^{t_i} = [\mathbf{-1}, \mathbf{1}](\phi(t_i) t_i - \phi(t_{i-1}) t_{i-1}).$$

After introducing $\mathcal{D}^{(\eta)} = \frac{A^\eta}{(\eta+1)!} \cdot \mathcal{U}$, one can rewrite (A.2) as

$$\begin{aligned}\int_{t_{i-1}}^{t_i} e^{A(t_i-t)}\, dt\,\mathcal{U} =& \mathcal{D}^{(0)}(t_i - t_{i-1}) + \mathcal{D}^{(1)}(t_i^2 - t_{i-1}^2) + \ldots + \mathcal{D}^{(\eta)}(t_i^{\eta+1} - t_{i-1}^{\eta+1}) \\ & + [-\mathbf{1}, \mathbf{1}] \cdot \mathcal{U} \cdot (\phi(t_i)t_i - \phi(t_{i-1})t_{i-1}).\end{aligned} \tag{A.3}$$

Inserting (A.3) into (A.1) yields

$$\begin{array}{ll} \mathcal{P}^{\mathcal{R}}(r) \approx & \\ & (\mathcal{D}^{(0)}(t_1 - t_0) \quad +\mathcal{D}^{(1)}(t_1^2 - t_0^2) \quad +\ldots \quad +\mathcal{D}^{(\eta)}(t_1^{\eta+1} - t_0^{\eta+1}) \\ + & (\mathcal{D}^{(0)}(t_2 - t_1) \quad +\mathcal{D}^{(1)}(t_2^2 - t_1^2) \quad +\ldots \quad +\mathcal{D}^{(\eta)}(t_2^{\eta+1} - t_1^{\eta+1}) \\ + & \ldots \\ + & \underbrace{(\mathcal{D}^{(0)}(t_l - t_{l-1})}_{\Sigma} \quad +\underbrace{\mathcal{D}^{(1)}(t_l^2 - t_{l-1}^2)}_{\Sigma} \quad +\ldots \quad +\underbrace{\mathcal{D}^{(\eta)}(t_l^{\eta+1} - t_{l-1}^{\eta+1})}_{\Sigma} \\ + & [-\mathbf{1}, \mathbf{1}] \cdot \mathcal{U} \cdot (\phi(t_1)t_1 - \phi(t_0)t_0) \\ + & [-\mathbf{1}, \mathbf{1}] \cdot \mathcal{U} \cdot (\phi(t_2)t_2 - \phi(t_1)t_1) \\ + & \ldots \\ + & [-\mathbf{1}, \mathbf{1}] \cdot \underbrace{\mathcal{U} \cdot (\phi(t_l)t_l - \phi(t_{l-1})t_{l-1})}_{\Sigma} \end{array} \tag{A.4}$$

The summation symbols indicate that the terms written in one column are summed up. Since the expressions $(t_i^m - t_{i-1}^m)$ are positive scalars, the following statement can be used for the summation: For any two positive scalars $a, b \in \mathbb{R}^+$ and the convex set $\mathcal{V}$, one can state that

$$\{a \cdot s + b \cdot s | s \in \mathcal{V}\} = a \cdot \mathcal{V} + b \cdot \mathcal{V} = (a + b) \cdot \mathcal{V}. \tag{A.5}$$

From this follows that

$$\begin{aligned} \mathcal{D}^{(m-1)}(t_1^m - t_0^m) + \mathcal{D}^{(m-1)}(t_2^m - t_1^m) + \ldots + \mathcal{D}^{(m-1)}(t_l^m - t_{l-1}^m) &= \mathcal{D}^{(m-1)}(\underbrace{t_l^m - t_0^m}_{r^m}), \\ \mathcal{U}(\phi(t_1)t_1 - \phi(t_0)t_0) + \ldots + \mathcal{U}(\phi(t_l)t_l - \phi(t_{l-1})t_{l-1}) &= \mathcal{U}(\underbrace{\phi(t_l)t_l - \phi(t_0)t_0}_{\phi(r)r}), \end{aligned} \tag{A.6}$$

because all the times except t_l^m and t_0^m cancel out. It is remarked that this result is independent of the number of intermediate time intervals l. Inserting (A.6) into (A.4) yields

$$\mathcal{P}^{\mathcal{R}}(r) \approx \mathcal{D}^{(0)}r + \mathcal{D}^{(1)}r^2 + \mathcal{D}^{(2)}r^3 + \ldots + \mathcal{E}(r) \cdot r \cdot \mathcal{U}.$$

As stated before, this result is independent of the number of intermediate time steps l, such that one can choose $l \to \infty$, meaning that the above result is no longer an approximation:

$$\mathcal{P}^{\mathcal{R}}(r) = \mathcal{D}^{(0)}r + \mathcal{D}^{(1)}r^2 + \mathcal{D}^{(2)}r^3 + \ldots + \mathcal{E}(r) \cdot r \cdot \mathcal{U} = \sum_{i=0}^{\eta} \left(\frac{A^i \cdot r^{i+1}}{(i+1)!} \cdot \mathcal{U} \right) + \mathcal{E}(r) \cdot r \cdot \mathcal{U}.$$

A.2. Spectral Density of Stochastic Piecewise Constant Signals

Given is a signal $u(t)$ which is constant within consecutive time intervals $[t_k, t_{k+1}]$, where $t_{k+1} - t_k = \tau$. This can also be written as

$$u(t) = \sum_{k=1}^{N_u} u(t_k)\texttt{rect}\left(\frac{t-(k-0.5)\tau}{\tau}\right), \texttt{rect}(t) = \begin{cases} 1, \text{ for } -0.5 < t < 0.5 \\ 0, \text{ otherwise} \end{cases} .$$

Next, its Fourier transform is calculated in order to obtain the spectral density. Using the properties $\mathcal{F}\{\texttt{rect}(t)\} = j\texttt{si}(\pi f)$, $\mathcal{F}\{x(t/a)\} = |a|X(af)$, and $\mathcal{F}\{x(t+\tau)\} = X(f)e^{-j2\pi f\tau}$, the following relations between the time and frequency domain can be formulated.

$$\begin{aligned} \texttt{rect}(t) &\circ\!\!-\!\!\!-\!\!\bullet\, j\texttt{si}(\pi f) \\ \texttt{rect}\left(\frac{t}{\tau}\right) &\circ\!\!-\!\!\!-\!\!\bullet\, j\tau\texttt{si}(\pi f\tau) \\ \texttt{rect}\left(\frac{t-(k-0.5)\tau}{\tau}\right) &\circ\!\!-\!\!\!-\!\!\bullet\, j\tau\texttt{si}(\pi f\tau)e^{-j2\pi f(k-0.5)\tau}. \end{aligned}$$

For the whole input signal, the Fourier transform is therefore

$$U(f) = j\tau\texttt{si}(\pi f\tau)e^{j\pi f\tau}\sum_{k=1}^{N_u} u(t_k)e^{-j2\pi fk\tau}.$$

The average spectral density function can then be calculated to

$$\Phi_u(f) = E[|\mathbf{U}(f)|^2] = \tau^2\texttt{si}^2(\pi f\tau)\sum_{k=1}^{N_u}\sum_{l=1}^{N_u} E[\mathbf{u}_{t_k}\mathbf{u}^*_{t_l}]e^{-j2\pi f\tau(k-l)}.$$

B. Monte Carlo Integration

The objective of Monte Carlo integration is to approximate the integral of a high dimensional function, which is one of the most common applications of Monte Carlo simulation. According to the mean value theorem for integrals, there exists a value c such that $\int_a^b f(x)\,dx = f(c)(b-a)$. When subdividing the interval $[a,b]$ into N_s equidistant subintervals of length $h = (b-a)/N_s$, the integral can be approximated with $c_i = a + (i - \frac{1}{2})h$ by

$$\int_a^b f(x)\,dx \approx h \sum_{i=1}^{N_s} f(c_i) = \hat{f} \cdot (b-a), \text{ where } \hat{f} = \frac{1}{N_s} \sum_{i=1}^{N_s} f(c_i).$$

When using this technique for integrals in several dimensions, the number of partitions grows exponentially since N_s^n regions are necessary if each dimension has N_s partitions. For this reason, the values c_i at which the function is evaluated are randomized in the Monte Carlo integration. The result of a randomized sampling of c_i can be further improved when introducing a weight function $w(x)$ so that there are more samples at interesting points:

$$\int_a^b f(x)\,dx = \int_a^b \frac{f(x)}{w(x)} w(x)\,dx, \text{ where } \int_a^b w(x)\,dx = 1 \text{ is normalized and } w(x) > 0.$$

A change of variable from x to $y(x) = \int_a^x w(\tilde{x})d\tilde{x}$ such that $\frac{dy}{dx} = w(x)$, $y(x=a) = 0$, $y(x=b) = 1$ yields

$$\int_a^b f(x)\,dx = \int_a^b \frac{f(x)}{w(x)} w(x)\,dx = \int_0^1 \frac{f(x(y))}{w(x(y))}\,dy \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}, \tag{B.1}$$

where $\mathbf{y}_i$ are the randomized values. In the multidimensional case, only the integration region has to be changed to $\int_X f(x)\,dx \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}$. For a uniform sampling, the weighting function is chosen as $w(x) = \frac{1}{\mathtt{V}(X)}$ and $\mathtt{V}()$ is the volume operator.

The variance of the Monte Carlo integral in (B.1) can be computed to

$$\mathtt{Var}\left(\frac{1}{N_s} \sum_{i=1}^{N_s} \frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}\right) = \frac{1}{N_s^2} \mathtt{Var}\left(\sum_{i=1}^{N_s} \frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}\right) = \frac{1}{N_s^2} \sum_{i=1}^{N_s} \mathtt{Var}\left(\frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}\right).$$

using $\mathtt{Var}(a\mathbf{x}) = a^2\mathtt{Var}(\mathbf{x})$ and $\mathtt{Var}(\mathbf{x}+\mathbf{y}) = \mathtt{Var}(\mathbf{x}) + \mathtt{Var}(\mathbf{y})$ when the random variables $\mathbf{x}$ and $\mathbf{y}$ are uncorrelated. Since the random variables $\mathbf{y}_i$ are identically generated, it follows that $\mathtt{Var}\left(\frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}\right) = \mathtt{Var}\left(\frac{f(x(\mathbf{y}))}{w(x(\mathbf{y}))}\right) \forall i$ which further simplifies the variance computation.

$$\frac{1}{N_s^2}\sum_{i=1}^{N_s}\mathtt{Var}\left(\frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}\right) = \frac{1}{N_s^2}\,N_s\,\mathtt{Var}\left(\frac{f(x(\mathbf{y}))}{w(x(\mathbf{y}))}\right) = \frac{1}{N_s}\mathtt{Var}\left(\frac{f(x(\mathbf{y}))}{w(x(\mathbf{y}))}\right). \tag{B.2}$$

Thus, the variance of the result reduces by $\frac{1}{N_s}$. Since the Monte Carlo approach has no systematic error, the result has no so-called bias and the variance equals the mean squared error, meaning that the mean error reduces with $\frac{1}{\sqrt{N_s}}$.

When predicting the future behavior of traffic participants, one is not interested in computing an integral, but one is interested in the probability that a state is in a certain cell of the discretized state space ($p_i = P(x \in \mathcal{X}_i)$). After introducing the indicator function

$$\mathtt{ind}_i(x) = \begin{cases} 1, \text{ if } x \in \mathcal{X}_i \\ 0, \text{ otherwise} \end{cases}$$

the probability p_i can be computed as

$$p_i = P(x \in \mathcal{X}_i) = \int_{\mathcal{X}} \mathtt{ind}_i(x) f(x)\, dx \overset{(B.1)}{\approx} \frac{1}{N_s}\sum_{i=1}^{N_s}\mathtt{ind}_i(x)\frac{f(x(\mathbf{y}_i))}{w(x(\mathbf{y}_i))}, \tag{B.3}$$

where $\mathcal{X} \supseteq \mathcal{X}_i$. When one would like to compute the crash probability, the indicator function has to be exchanged to a function that is 1 if a crash has been obtained and 0 otherwise. There exists a nice result for the probability distribution p_i of traffic participants when using importance sampling.

Example B.1 (Mean Squared Error for Importance Sampling): When using importance sampling ($f(x(\mathbf{y}_i)) = w(x(\mathbf{y}_i))$), the variance in (B.2) simplifies to $\frac{1}{N_s}$. Thus, when computing $p_i = P(x \in \mathcal{X}_i) = \int_{\mathcal{X}} \mathtt{ind}_i(x) f(x)\, dx$ with importance sampling, the variance becomes

$$\mathtt{Var}(p_i) \overset{(B.3),(B.2)}{=} \frac{1}{N_s}\mathtt{Var}(\mathtt{ind}_i(x)) = \frac{1}{N_s}\big(E[\mathtt{ind}_i^2(x)] - E[\mathtt{ind}_i(x)]^2\big).$$

Since the indicator function is either 1 or 0, the square of the indicator function can be removed. Further, the expectation of the indicator function $E[\mathtt{ind}_i(x)]$ equals the probability that the state is in that cell $p_i = P(x \in \mathcal{X}_i)$. Thus, the variance of p_i simplifies to

$$\mathtt{Var}(p_i) = \frac{1}{N_s}p_i\big(1 - p_i\big).$$

With the upper bound $p_i\big(1 - p_i\big) \leq \frac{1}{4}$ because $0 \leq p_i \leq 1$, one finally obtains the simple bound

$$\mathtt{Var}(p_i) \leq \frac{1}{4\,N_s}.$$ □

Bibliography

[1] A. Abate. *Probabilistic Reachability for Stochastic Hybrid Systems: Theory, Computations, and Applications*. PhD thesis, University of California, Berkeley, 2007.

[2] A. Abate, S. Amin, M. Prandini, J. Lygeros, and S. Sastry. Computational approaches to reachability analysis of stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 4416, pages 4–17. Springer, 2007.

[3] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44:2724–2734, 2008.

[4] M. Abdel-Aty and A. Pande. ATMS implementation system for identifying traffic conditions leading to potential crashes. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):78–91, 2006.

[5] T. Alamo, J. M. Bravo, and E. F. Camacho. Guaranteed state estimation by zonotopes. In *Proc. of the 42nd IEEE Conference on Decision and Control*, pages 5831–5836, 2003.

[6] R. Alur, C. Coucoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[7] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[8] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.

[9] F. Anstett, G. Millérioux, and G. Bloch. Polytopic observer design for LPV systems based on minimal convex polytope finding. *Algorithms & Computational Technology*, 3:23–43, 2009.

[10] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control*, LNCS 1790, pages 20–31. Springer, 2000.

[11] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler. Recent progress in continuous and hybrid reachability analysis. In *Proc. of the 2006 IEEE Conference on Computer Aided Control Systems Design*, pages 1582–1587, 2006.

[12] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In *Hybrid Systems: Control and Computation*, pages 20–35, 2003.

[13] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43:451–476, 2007.

[14] E. Asarin, T. Dang, and O. Maler. d/dt: A verification tool for hybrid systems. In *Proc. of the Conference on Decision and Control*, pages 2893–2898, 2001.

[15] K. Aso and T. Kindo. Stochastic decision-making method for autonomous driving system that minimizes collision probability. In *Proc. of the FISITA World Automotive Congress*, 2008.

[16] C. Baier and J.-P. Katoen. *Principles of Model Checking.* MIT Press, 2008.

[17] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38:91–109, 1999.

[18] A. Barth and U. Franke. Where will the oncoming vehicle be the next second? In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 1068–1073, 2008.

[19] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, Y. Wang, and C. Weise. New generation of UPPAAL. In *Proc. of the International Workshop on Software Tools for Technology Transfer*, 1998.

[20] D. Berleant. Automatically verified reasoning with both intervals and probability density functions. *Interval Computations*, 2:48–70, 1993.

[21] D. Berleant and C. Goodman-Strauss. Bounding the results of arithmetic operations on random variables of unknown dependency using intervals. *Reliable Computing*, 4:147–165, 1998.

[22] D. P. Bertsekas and I. B. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *IEEE Transactions on Automatic Control*, 16:117–128, 1971.

[23] M. Berz and G. Hoffstätter. Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing*, 4:83–97, 1998.

[24] A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 2993, pages 142–156. Springer, 2004.

[25] H. A. P. Blom, G. J. Bakker, and J. Krystul. Probabilistic reachability analysis for large scale stochastic hybrid systems. In *Proc. of the 46th IEEE Conference on Decision and Control*, pages 3182–3189, 2007.

[26] H. A. P. Blom, J. Krystul, and G. J. Bakker. *Free Flight Collision Risk Estimation by Sequential Monte Carlo Simulation*, chapter 10, pages 247–279. Taylor & Francis CRC Press, 2006.

[27] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Hybrid Systems: Computation and Control*, LNCS 1790, pages 73–88. Springer, 2000.

[28] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 1486, pages 298–302. Springer, 1998.

[29] M. Brackstone, M. McDonald, and J. Wu. Lane changing on the motorway: Factors affecting its occurrence, and their implications. In *Proc. of the 9th International Conference on Road Transport Information and Control*, pages 160–164, 1998.

[30] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan. RRTs for nonlinear, discrete, and hybrid planning and control. In *Proc. of the 42nd IEEE Conference on Decision and Control*, pages 657–663, 2003.

[31] A. E. Broadhurst, S. Baker, and T. Kanade. A prediction and planning framework for road safety analysis, obstacle avoidance and driver information. In *Proc. of the 11th World Congress on Intelligent Transportation Systems*, October 2004.

[32] A. E. Broadhurst, S. Baker, and T. Kanade. Monte Carlo road safety reasoning. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 319–324, 2005.

[33] M. Bujorianu and J. Lygeros. Toward a general theory of stochastic hybrid systems. In H. Blom and J. Lygeros, editors, *Stochastic Hybrid Systems: Theory and Safety Critical Applications*, number 337 in LNCIS, pages 3–30. Springer, 2006.

[34] M. L. Bujorianu. Extended stochastic hybrid systems and their reachability problem. In *Hybrid Systems: Computation and Control*, pages 234–249, 2004.

[35] M. L. Bujorianu. A statistical inference method for the stochastic reachability analysis. In *Proc. of the 44th IEEE Conference on Decision and Control*, pages 8088–8093, 2005.

[36] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Springer, 2009.

[37] G. Chesi. Establishing stability and instability of matrix hypercubes. *System & Control Letters*, 54:381–388, 2005.

[38] A. Chutinan and B. H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *Hybrid systems: Computation and Control*, LNCS 1569, pages 76–90. Springer, 1999.

[39] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. In *IEEE Transactions on Automatic Control*, volume 48, pages 64–75, 2003.

[40] E. Clarke, A. Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald. *Verification of Hybrid Systems based on Counterexample-Guided Abstraction Refinement*, pages 192–207. LNCS 2619. Springer, 2003.

[41] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 2000.

[42] C. Combastel. A state bounding observer based on zonotopes. In *Proc. of the European Control Conference*, 2003.

[43] C. Combastel. A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes. In *Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, pages 7228–7234, 2005.

[44] G. E. Coxson. Computing exact bounds on elements of an inverse interval matrix is NP-hard. *Reliable Computing*, 5:137–142, 1999.

[45] I. Dagli, M. Brost, and G. Breuel. Action recognition and prediction for driver assistance systems using dynamic belief networks. In *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, LNCS 2592, pages 179–194. Springer, 2003.

[46] I. Dagli and D. Reichardt. Motivation-based approach to behavior prediction. In *Proc. of the Intelligent Vehicles Symposium*, pages 227–233, 2002.

[47] T. Dang. *Vérification et synthèse des systèmes hybrides.* PhD thesis, Institut National Polytechnique de Grenoble, 2000.

[48] T. Dang. Approximate reachability computation for polynomial systems. In *Hybrid Systems: Computation and Control*, pages 138–152, 2006.

[49] T. Dang, C. Le Guernic, and O. Maler. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology*, LNCS 5688, pages 126–141. Springer, 2009.

[50] T. Dang and O. Maler. Reachability analysis via face lifting. In *Hybrid Systems: Computation and Control*, pages 96–109, 1998.

[51] T. Dang and D. Salinas. Image computation for polynomial dynamical systems using the Bernstein expansion. In *Computer Aided Verification*, LNCS 5643, pages 219–232. Springer, 2009.

[52] S. Danielsson, L. Petersson, and A. Eidehall. Monte Carlo based threat assessment: Analysis and improvements. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 233–238, 2007.

[53] M. H. A. Davis. Piecewise-deterministic Markov processes: A general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society*, 46:353–388, 1984.

[54] P. P. Dey, S. Chandra, and S. Gangopadhaya. Lateral distribution of mixed traffic on two-lane roads. *Journal of Transportation Engineering*, 132:597–600, 2006.

[55] A. D'Innocenzo, A. A. Julius, G. J. Pappas, M. D. Di Benedetto, and S. Di Gennaro. Verification of temporal properties on hybrid automata by simulation relations. In *Proc. of the 46th IEEE Conference on Decision and Control*, pages 4039–4044, 2007.

[56] A. Donzé. *Trajectory-Based Verification and Controller Synthesis for Continuous and Hybrid Systems.* PhD thesis, University Joseph Fourier, 2007.

[57] A. Donzé and O. Maler. Systematic simulations using sensitivity analysis. In *Hybrid Systems: Computation and Control*, LNCS 4416, pages 174–189. Springer, 2007.

[58] D. Eberly. Dynamic collision detection using oriented bounding boxes. Technical report, Geometric Tools, Inc., 2002.

[59] A. Eidehall and L. Petersson. Threat assessment for general road scenes using Monte Carlo sampling. In *Proc. of the Intelligent Transportation Systems Conference*, pages 1173–1178, 2006.

[60] A. Eidehall and L. Petersson. Statistical threat assessment for general road scenes using Monte Carlo sampling. *IEEE Transactions on Intelligent Transportation Systems*, 9:137–147, 2008.

[61] J. M. Esposito. Randomized test case generation for hybrid systems: Metric selection. In *Proc. of the 36th Southeastern Symposium of System Theory*, pages 236–240, 2004.

[62] A. Fehnker and F. Ivanvcić. Benchmarks for hybrid systems verification. In *Hybrid Systems: Computation and Control*, LNCS 2993, pages 326–341. Springer, 2004.

[63] G. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations.* PhD thesis, Radboud Universiteit Nijmegen, 2005.

[64] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *International Journal on Software Tools for Technology Transfer*, 10:263–279, 2008.

[65] J. Freyer, B. Deml, M. Maurer, and B. Färber. ACC with enhanced situation awareness to reduce behavior adaptations in lane change situations. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 999–1004, 2007.

[66] K. Fukuda. From the zonotope construction to the Minkowski addition of convex polytopes. *Journal of Symbolic Computation*, 38(4):1261–1272, October 2004.

[67] C. W. Gardiner. *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences.* Springer, 1983.

[68] M. K. Ghosh, A. Arapostathis, and S. I. Marcus. Ergodic control of switching diffusions. *SIAM Journal on Control Optimization*, 35:1952–1988, 1997.

[69] A. Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, LNCS 3414, pages 291–305. Springer, 2005.

[70] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. In *Proc. of the 17th IFAC World Congress*, pages 8966–8971, 2008.

[71] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proc. of Hybrid Systems: Computation and Control*, LNCS 4981, pages 215–228. Springer, 2008.

[72] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*, LNCS 3927, pages 257–271. Springer, 2006.

[73] A. Girard and G. J. Pappas. Verification using simulation. In *Hybrid Systems: Computation and Control*, LNCS 3927, pages 272–286. Springer, 2006.

[74] M. Goebl. *Eine realzeitfähige Architektur zur Integration kognitiver Funktionen.* PhD thesis, TU München, 2009.

[75] M. Goebl and G. Färber. A real-time-capable hard- and software architecture for joint image and knowledge processing in cognitive automobiles. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 734–740, 2007.

[76] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30:171–180, 1996.

[77] D. H. Greene, J. J. Liu, J. E. Reich, Y. Hirokawa, T. Mikami, H. Ito, and A. Shinagawa. A computationally-efficient collision early warning system for vehicles, pedestrian and bicyclists. In *Proc. of the 15th World Congress on Intelligent Transportation Systems*, 2008.

[78] C. Le Guernic. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics.* PhD thesis, Univerité Joseph Fourier, 2009.

[79] L. J. Guibas, A. Nguyen, and L. Zhang. Zonotopes as bounding volumes. In *Proc. of the Symposium on Discrete Algorithms*, pages 803–812, 2005.

[80] Z. Han and B. H. Krogh. Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In *Proc. of the American Control Conference*, pages 1505–1510, 2006.

[81] T. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[82] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.

[83] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In *Hybrid Systems: Computation and Control*, LNCS 1790, pages 130–144. Springer, 2000.

[84] J. Hillenbrand, A. M. Spieker, and K. Kroschel. A multilevel collision mitigation approach - its situation assessment, decision making, and performance tradeoffs. *IEEE Transactions on Intelligent Transportation Systems*, 7:528–540, 2006.

[85] J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 1790, pages 160–173. Springer, 2000.

[86] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1450–1464, 2006.

[87] L. Jaulin, M. Kieffer, and O. Didrit. *Applied Interval Analysis.* Springer, 2006.

[88] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:583–592, 1996.

[89] H. Jula, E. B. Kosmatopoulos, and P. A. Ioannou. Collision avoidance analysis for lane changing and merging. *IEEE Transactions on Vehicular Technology*, 49:2295–2308, 2000.

[90] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas. Robust test generation and coverage for hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 4416, pages 329–342. Springer, 2007.

[91] V. Kaibel and M. E. Pfetsch. *Algebra, Geometry and Software Systems*, chapter Some Algorithmic Problems in Polytope Theory, pages 23–47. Springer, 2003.

[92] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. von Hundelshausen, O. Pink, C. Frese, and C. Stiller. Team AnnieWAY's autonomous system for the DARPA Urban Challenge 2007. *Journal of Field Robotics*, 25:615 – 639, 2008.

[93] A. Kanaris, E. B. Kosmatopoulos, and P. A. Ioannou. Strategies and spacing requirements for lane changing and merging in automated highway systems. *IEEE Transactions on Vehicular Technology*, 50:1568–1581, 2001.

[94] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg. On systematic simulation of open continuous systems. In *Hybrid Systems: Computation and Control*, LNCS 2623, pages 283–297. Springer, 2003.

[95] J.-P. Katoen. Perspectives in probabilistic verification. In *Proc. of the 2nd IEEE International Symposium on Theoretical Aspects of Software Engineering*, pages 3–10, 2008.

[96] M. Kloetzer and C. Belta. Reachability analysis of multi-affine systems. In *Hybrid Systems: Computation and Control*, LNCS 3927, pages 348–362. Springer, 2006.

[97] O. Kosheleva, V. Kreinovich, G. Mayer, and H. T. Nguyen. Computing the cube of an interval matrix is NP-hard. In *Proc. of the ACM symposium on Applied computing*, pages 1449–1453, 2005.

[98] X. Koutsoukos and D. Riley. Computational methods for reachability analysis of stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 3927, pages 377–391. Springer, 2006.

[99] W. Kühn. *Mathematical Visualization*, chapter Zonotope Dynamics in Numerical Quality Control, pages 125–134. Springer, 1998.

[100] W. Kühn. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61:47–67, 1998.

[101] A. Kurzhanski and I. Valyi. *Ellipsoidal Calculus for Estimation and Control.* Birkhäuser Boston, 1996.

[102] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *Hybrid Systems: Computation and Control*, LNCS 1790, pages 202–214. Springer, 2000.

[103] A. B. Kurzhanskiy and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 52:26–38, 2007.

[104] H. J. Kushner and P. G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer, 2000.

[105] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004.

[106] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 1569, pages 137–151. Springer, 1999.

[107] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computation for families of linear vector fields. *Symbolic Computation*, 32:231–253, 2001.

[108] K. Lee and H. Peng. Evaluation of automotive forward collision warning and collision avoidance algorithms. *Vehicle System Dynamics*, 43(10):735–751, 2005.

[109] C.-F. Lin, A. G. Ulsoy, and D. J. LeBlanc. Vehicle dynamics and external disturbance estimation for vehicle path prediction. *IEEE Transactions on Control Systems Technology*, 8:508–518, 2000.

[110] M. L. Liou. A novel method of evaluating transient response. In *Proceedings of the IEEE*, volume 54, pages 20–23, 1966.

[111] J. Lunze and B. Nixdorf. Representation of hybrid systems by means of stochastic automata. *Mathematical and Computer Modeling of Dynamical Systems*, 7:383–422, 2001.

[112] J. Lunze, B. Nixdorf, and J. Schröder. Deterministic discrete-event representations of linear continuous-variable systems. *Automatica*, 35:395–406, 1999.

[113] J. Lunze and J. Schröder. Computation of complete abstractions of quantised systems. In *Proc. of the European Control Conference*, pages 3137–3142, 2001.

[114] I. B. Makhlouf and S. Kowalewski. An evaluation of two recent reachability analysis tools for hybrid systems. In *Proc. of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems*, pages 377–382, 2006.

[115] I. B. Makhlouf, S. Kowalewski, M. Guillermo, C. Grunewald, and D. Abel. Safety assessment of networked vehicle platoon controllers – practical experiences with available tools. In *Proc. of the 3rd IFAC Conference on Analysis and Design of Hybrid Systems*, pages 292–297, 2009.

[116] O. Maler and G. Batt. Approximating continuous systems by timed automata. In *Formal Methods in Systems Biology*, LNCS 5054, pages 77–89. Springer, 2008.

[117] M. B. Mamoun and N. Pekergin. Closed-form stochastic bounds on the stationary distribution of Markov chains. *Probability in the Engineering and Informational Sciences*, 16:403–426, 2002.

[118] M. B. Mamoun and N. Pekergin. Computing closed-form stochastic bounds on transient distributions of Markov chains. In *Proc. of the Symposium on Applications and the Internet Workshops*, pages 260– 263, 2005.

[119] J. Maroto, E. Delso, J. Félez, and J. M. Cabanellas. Real-time traffic simulation with a microscopic model. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):513–527, 2006.

[120] J. C. McCall, D. P. Wipf, M. M. Trivedi, and B. D. Rao. Lane change intent analysis using robust operators and sparse bayesian learning. *IEEE Transactions on Intelligent Transportation Systems*, 8:431–440, 2007.

[121] I. Mitchell, A. M. Bayen, and C. J. Tomlin. Validating a Hamilton–Jacobi approximation to hybrid system reachable sets. In *Hybrid Systems: Computation and Control*, pages 418–432, 2001.

[122] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[123] B. T. Morris and M. M. Trivedi. Learning, modeling, and classification of vehicle track patterns from live video. *IEEE Transactions on Intelligent Transportation Systems*, 9:425–437, 2008.

[124] D. Mortari. The n-dimensional cross product and its application to the matrix eigenanalysis. In *Proc. of the AIAA/AAS Astrodynamics Conference*, 1996.

[125] M. E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, 2:19–20, 1959.

[126] Y. Nishiwaki, C. Miyajima, H. Kitaoka, and K. Takeda. Stochastic modeling of vehicle trajectory during lane-changing. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1377–1380, 2009.

[127] Y. Nishiwaki, C. Miyajima, N. Kitaoka, R. Terashima, T. Wakita, and K. Takeda. Generating lane-change trajectories of individual drivers. In *Proc. of the IEEE International Conference on Vehicular Electronics and Safety*, pages 271–275, 2008.

[128] B. Oksendal. *Stochastic Differential Equations.* Springer, 2003.

[129] E. C. B. Olsen, S. E. Lee, W. W. Wierwille, and M. J. Goodman. Analysis of distribution, frequency, and duration of naturalistic lane changes. In *Proc. of the 46th Annual Meeting of the Human Factors and Ergonomics Society*, pages 1789–1793, 2002.

[130] E. P. Oppenheimer and A. N. Michel. Application of interval analysis techniques to linear systems: Part I - fundamental results. *IEEE Transactions on Circuits and Systems*, Volume 35:1129 – 1138, 1988.

[131] E. P. Oppenheimer and A. N. Michel. Application of interval analysis techniques to linear systems: Part II - the interval matrix exponential function. *IEEE Transactions on Circuits and Systems*, 35:1230 – 1242, 1988.

[132] E. P. Oppenheimer and A. N. Michel. Application of interval analysis techniques to linear systems: Part III - initial value problems. *IEEE Transactions on Circuits and Systems*, 35:1243 – 1256, 1988.

[133] J. O'Rourke. Finding minimal enclosing boxes. *International Journal of Computer and Information Sciences*, 14:183–199, 1985.

[134] M. Peden, R. Scurfield, D. Sleet, D. Mohanand A. A. Hyder, E. Jarawan, and C. Mathers. *World Report on Road Traffic Injury Prevention*. World Health Organization, 2004.

[135] A. Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41:143–189, 2008.

[136] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone. Sensor fusion for predicting vehicles' path for collision avoidance systems. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):549–562, 2007.

[137] S. Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42:117–126, 2006.

[138] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, LNCS 2993, pages 477–492. Springer, 2004.

[139] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52:1415–1428, 2007.

[140] M. Prandini and J. Hu. *Stochastic hybrid systems*, chapter Stochastic reachability: Theoretical foundations and numerical approximation., pages 107–138. Taylor & Francis Group/CRC Press, 2006.

[141] M. Prandini and J. Hu. *Stochastic hybrid systems: theory and safety critical applications*, chapter A stochastic approximation method for reachability computations, pages 107–139. Springer, 2006.

[142] J. Preußig, O. Stursberg, and S. Kowalewski. Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis. In *Hybrid Systems: Computation and Control*, LNCS 1569, pages 209–222. Springer, 1999.

[143] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *Proc. of the 6th International Conference on Computer Aided Verification*, LNCS 818, pages 95–104. Springer, 1994.

[144] T. Raissi, N. Ramdani, and Y. Candau. Set membership state and parameter estimation for systems described by nonlinear differential equations. *Automatica*, 40:1771–1777, 2004.

[145] N. Ramdani, N. Meslem, and Y. Candau. Reachability analysis of uncertain nonlinear systems using guaranteed set integration. In *Proc. of the 17th IFAC World Congress*, pages 8972–8977, 2008.

[146] N. Ramdani, N. Meslem, and Y. Candau. Reachability of uncertain nonlinear systems using a nonlinear hybridization. In *Hybrid Systems: Computation and Control*, LNCS 4981, pages 415–428. Springer, 2008.

[147] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. Wiley, 2008.

[148] F. M. Schlaepfer and F. C. Schweppe. Continuous-time state estimation under disturbances bounded by convex sets. *IEEE Transactions on Automatic Control*, 17:197–205, 1972.

[149] C. Schmidt, F. Oechsle, and W. Branz. Research on trajectory planning in emergency situations with multiple objects. In *Proc. of the IEEE Intelligent Transportation Systems Conference*, pages 988–992, 2006.

[150] J. Schröder. *Modelling, State Observation and Diagnosis of Quantised Systems*. Springer, 2003.

[151] S. Sekizawa, S. Inagaki, T. Suzuki, S. Hayakawa, N. Tsuchida, T. Tsuda, and H. Fujinami. Modeling and recognition of driving behavior based on stochastic switched ARX model. *IEEE Transactions on Intelligent Transportation Systems*, 8:593–606, 2007.

[152] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. In *Proc. of the 40th IEEE Conference on Decision and Control*, pages 2867–2874, 2001.

[153] H. Stark and J. W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, 1994.

[154] C. Stiller, G. Färber, and S. Kammel. Cooperative cognitive automobiles. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 215–220, 2007.

[155] O. Stursberg, S. Engell, and S. Kowalewski. Timed approximations of hybrid processes for controller verification. In *Proc. of the 14th IFAC World Congress*, pages 73–78, 1999.

[156] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh. Specification-guided analysis of hybrid systems using a hierarchy of validation methods. In *Proc. of the 1st IFAC Conference on Analysis and Design of Hybrid Systems*, pages 289–295, 2003.

[157] O. Stursberg, S. Kowalewski, and S. Engell. On the generation of timed discrete approximations for continuous systems. *Mathematical and Computer Models of Dynamical Systems*, 6:51–70, 2000.

[158] O. Stursberg and B. H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 2623, pages 482–497. Springer, 2003.

[159] W. Takano, A. Matsushita, K. Iwao, and Y. Nakamura. Recognition of human driving behaviors based on stochastic symbolization of time series signal. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 167–172, 2008.

[160] R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer, 2004.

[161] M. Thuy and F. Puente León. Non-linear, shape independent object tracking based on 2D lidar data. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 532–537, 2009.

[162] T. Toledo. *Integrated Driving Behavior Modeling*. PhD thesis, Massachusetts Institute of Technology, 2003.

[163] C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi. Computational techniques for the verification and control of hybrid systems. In *Proceedings of the IEEE*, volume 91, pages 986–1001, 2003.

[164] S. Vacek, T. Gindele, J. M. Zöllner, and R. Dillmann. Using case-based reasoning for autonomous vehicle guidance. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4271–4276, 2007.

[165] J. van den Berg. *Path Planning in Dynamic Environments*. PhD thesis, Utrecht University, 2007.

[166] D. Vasquez, T. Fraichard, and C. Laugier. Incremental learning of statistical motion patterns with growing hidden markov models. *IEEE Transactions on Intelligent Transportation Systems*, 10:403–416, 2009.

[167] E. Velenis and P. Tsiotras. Optimal velocity profile generation for given acceleration limits: theoretical analysis. In *Proc. of the American Control Conference*, pages 1478 – 1483, 2005.

[168] A. L. Visintini, W. Glover, J. Lygeros, and J. Maciejowski. Monte Carlo optimization for conflict resolution in air traffic control. *IEEE Transactions on Intelligent Transportation Systems*, 7:470–482, 2006.

[169] F. Vivien and N. Wicker. Minimal enclosing parallelepiped in 3D. *Computational Geometry: Theory and Applications*, 29:177–190, 2004.

[170] C. Weibel. *Minkowski Sums of Polytopes: Combinatorics and Computation*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2007.

[171] S. Weinzierl. Introduction to Monte Carlo methods. Technical report, NIKHEF Theory Group Kruislaan 409, 1098 SJ Amsterdam, The Netherlands, 2000.

[172] M. Werling, M. Goebl, O. Pink, and C. Stiller. A hardware and software framework for cognitive automobiles. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 1080–1085, 2008.

[173] M. Werling and L. Gröll. Low-level controllers realizing high-level decisions in an autonomous vehicle. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 1113–1118, 2008.

[174] Y. U. Yim and S.-Y. Oh. Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction. *IEEE Transactions on Vehicular Technology*, 53:1076–1084, 2004.

[175] R. Yuster and U. Zwick. Fast sparse matrix multiplication. *ACM Transactions on Algorithms*, 1:2–13, 2005.

[176] G. M. Ziegler. *Lectures on Polytopes.* Graduate Texts in Mathematics. Springer, 1995.

Own Publications and Supervised Student Projects

[177] D. Althoff, M. Althoff, D. Wollherr, and M. Buss. Probabilistic collision state checker for crowded environments. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1492–1498, 2010.

[178] M. Althoff, D. Althoff, D. Wollherr, and M. Buss. Safety verification of autonomous vehicles for coordinated evasive maneuvers. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010.

[179] M. Althoff, O. Stursberg, and M. Buss. Online verification of cognitive car decisions. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 728–733, 2007.

[180] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of linear systems with uncertain parameters and inputs. In *Proc. of the 46th IEEE Conference on Decision and Control*, pages 726–732, 2007.

[181] M. Althoff, O. Stursberg, and M. Buss. Safety assessment of autonomous cars using verification techniques. In *Proc. of the American Control Conference*, pages 4154–4159, 2007.

[182] M. Althoff, O. Stursberg, and M. Buss. Erreichbarkeitsanalyse von Verkehrsteilnehmern zur Verbesserung von Fahrerassistenzsystemen. In *Proc. of 3. Tagung Aktive Sicherheit durch Fahrerassistenz*, 2008.

[183] M. Althoff, O. Stursberg, and M. Buss. Online-Analyse von Fahrstrategien kognitiver autonomer Fahrzeuge. In *Proc. of Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel (AAET)*, pages 314–330, 2008.

[184] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proc. of the 47th IEEE Conference on Decision and Control*, pages 4042–4048, 2008.

[185] M. Althoff, O. Stursberg, and M. Buss. Sicherheitsbewertung von Fahrstrategien kognitiver Automobile. *at - Automatisierungstechnik*, 56:653–661, 2008.

[186] M. Althoff, O. Stursberg, and M. Buss. Stochastic reachable sets of interacting traffic participants. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 1086–1092, 2008.

[187] M. Althoff, O. Stursberg, and M. Buss. Verification of uncertain embedded systems by computing reachable sets based on zonotopes. In *Proc. of the 17th IFAC World Congress*, pages 5125–5130, 2008.

[188] M. Althoff, O. Stursberg, and M. Buss. Model-based probabilistic collision detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 10:299 – 310, 2009.

[189] M. Althoff, O. Stursberg, and M. Buss. Safety assessment for stochastic linear systems using enclosing hulls of probability density functions. In *Proc. of the European Control Conference*, pages 625–630, 2009.

[190] M. Althoff, O. Stursberg, and M. Buss. Safety assessment of driving behavior in multi-lane traffic for autonomous vehicles. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 893–900, 2009.

[191] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4:233–249, 2010.

[192] M. Goebl, M. Althoff, M. Buss, G. Färber, F. Hecker, B. Heißing, S. Kraus, R. Nagel, F. Puente León, F. Rattei, M. Russ, M. Schweitzer, M. Thuy, C. Wang, and H.-J. Wünsche. Design and capabilities of the Munich cognitive automobile. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 1101–1107, 2008.

[193] S. Kraus, M. Althoff, B. Heißing, and M. Buss. Cognition and emotion in autonomous cars. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 635–640, 2009.

[194] A. Mergel. Comparison of stochastic reachability analysis and Monte Carlo simulation for the safety assessment of road scenes. Bachelor thesis, 2009. Technische Universität München.

[195] F. Rohrmüller, M. Althoff, D. Wollherr, and M. Buss. Probabilistic mapping of dynamic obstacles using Markov chains for replanning in dynamic environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2504–2510, 2008.

[196] B. Sari. Computation of probabilistic reachable sets for interacting traffic participants. Bachelor thesis, 2008. Technische Universität München.

[197] I. Shevchenko. Wegpunktbasierte Trajektorienplanung kognitiver Fahrzeuge. Master's thesis, TU München, 2007.

[198] M. Thuy, M. Goebl, F. Rattei, M. Althoff, F. Obermeier, S. Hawe, R. Nagel, S. Kraus, C. Wang, F. Hecker, M. Russ, M. Schweitzer, F. Puente León, G. Färber, M. Buss, K. Diepold, J. Eberspächer, B. Heißing, and H.-J. Wünsche. Kognitive Automobile: Neue Konzepte und Ideen des Sonderforschungsbereiches/TR-28. In *Proc. of 3. Tagung Aktive Sicherheit durch Fahrerassistenz*, 2008.