TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Flugsystemdynamik

Robust Trajectory Optimization Applying Chance Constraints and Generalized Polynomial Chaos

Patrick A. Piprek, M.Sc.

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Prof. Dr.-Ing. habil. Boris Lohmann
Prüfer der Dissertation:	1. Prof. Dr.-Ing. Florian Holzapfel
	2. Prof. Dr. Sébastien Gros

Die Dissertation wurde am 18.09.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 19.03.2020 angenommen.

Abstract

In today's engineering applications, the quantification of uncertainties, which are inherent to a dynamic system, is of growing importance to describe and understand the complex behavior of systems in a complete way. Thus, it is also of importance in open-loop direct optimal control (OC) applications ("trajectory optimization"), i.e., when an optimal trajectory of a dynamic system with respect to a cost function is calculated. Here, the presence of uncertainties normally requires a trade-off between the optimality of the cost function and the robustness, i.e., the reduction of the uncertainty influence.

To improve methodologies in OC with uncertainties, this thesis connects the method of generalized polynomial chaos (gPC), which is used for efficient uncertainty quantification, with OC techniques. By this, the solution of robust open-loop direct optimal control problems (ROCPs), i.e., open-loop direct optimal control problems (OCPs) that connect optimality with robustness considering uncertainties, is possible. To this end, the thesis introduces efficient formulations to connect the gPC method with the ROCP, e.g., by means of distributed open-loop direct optimal control (DOC) or chance-constrained open-loop direct optimal control (CC-OC): For DOC, the thesis provides novel formulations of the distribution of the statistical moments from the gPC method that enable an efficient solution of the ROCP. This is especially viable for large OCPs that benefit from being solved unconnected and in smaller parts.

Furthermore, the CC-OC formulation utilizes a developed transcription method for the OCP that is based on the gPC expansion. By this, the optimization is made in the gPC domain and statistical moments can directly be optimized. In addition, chance constraints (CCs) can be solved efficiently by sampling techniques, as the gPC expansion provides an analytic approximation of the dynamic system response. This representation is then also used for the introduction of rare-event CCs in the ROCP formulation based on subset simulation (SubSim). The connection gives e.g., the possibility to use the developed method in future aviation certification processes as well, as a risk analysis of the system when operated at the optimal point can be conducted.

Overall, the developed robust open-loop direct optimal control (ROC) frameworks are examined in different case studies ranging from controller gain design over air race applications to obstacle avoidance to show their applicability in a very general context. The case studies also prove the viability and real-world applicability of the chosen methodologies.

Kurzfassung

In aktuellen Ingenieursfragestellungen bekommt die Beschreibung von Unsicherheiten, welche sich inhärent in dynamischen Systemen befinden, eine immer größere Bedeutung, um das komplexe Systemverhalten in vollständiger Weise zu beschreiben und zu verstehen. Deswegen ergibt sich natürlicherweise auch ihre Bedeutung in der Optimalsteuerung („Trajektorienoptimierung“), also wenn die optimale Trajektorie eines Systems auf Basis einer Kostenfunktion berechnet werden soll. Hierbei wird durch vorhandene Unsicherheiten normalerweise eine Abwägung zwischen Optimalität und Robustheit, also der Verringerung des Unsicherheitseinflusses, nötig.

Um die Methoden der Optimalsteuerung mit Unsicherheiten weiter zu entwickeln, kombiniert die vorliegende Arbeit die Methode des generalisierten Polynomchaos zur effizienten Unsicherheitsbeschreibung mit Optimalsteuerungsmethoden. Dadurch wird die Lösung robuster Optimalsteuerungsprobleme, also der Verbindung von Optimalität und Robustheit, ermöglicht. Dies wird in der vorliegenden Arbeit durch eine effiziente Verbindung des Polynomchaos mit der Optimalsteuerung erreicht, was mit der Entwicklung verteilter und wahrscheinlichkeitsbeschränkter Optimalsteuerungsmethoden endet: Im Bereich der verteilten Optimalsteuerung werden besonders die statistischen Momente so formuliert, dass sich eine effiziente Lösung des Problems ergibt. Das Anwendungsgebiet der verteilten Optimalsteuerung sind dabei hauptsächlich große Optimierungsprobleme.

In der wahrscheinlichkeitsbeschränkten Optimierung wird entsprechend die deterministische Problemformulierung so abgeändert, dass das Polynomchaos direkt mittels der Optimierungsvariablen optimierbar ist. Dadurch können sowohl statistische Momente als auch Wahrscheinlichkeitsbeschränkungen direkt optimiert werden. Hierbei wird u.a. ausgenutzt, dass das Polynomchaos eine analytische Näherung der exakten Systemantwort liefert und somit effizient Stichproben zur Wahrscheinlichkeitsberechnung erstellt werden. Die Wahrscheinlichkeitsbedingungen werden weiterhin insbesondere zur Modellierung seltener Fehlerwahrscheinlichkeiten mittels Subset Simulation genutzt. Dies kann den zukünftigen Einsatz der entwickelten Methoden z.B. in der Luftfahrtzertifizierung oder der Risikoanalyse ermöglichen.

Schlussendlich wird die Anwendbarkeit der entwickelten Methoden zur robusten Optimalsteuerung in verschiedenen Anwendungsbeispielen gezeigt (Reglerauslegung, Luftrennen, Hindernisvermeidung). Im Besonderen soll hierbei die Möglichkeit der Anwendung in realen Optimalsteuerungsproblemen gezeigt werden.

Acknowledgment

This Eng.D. thesis is the result of the work while being employed at the Institute of Flight System Dynamics of the Technical University of Munich in between 2016–2019. Therefore, I would, first of all, like to thank Prof. Dr.-Ing. Florian Holzapfel for the possibility to let me work on this Eng.D. topic at his institute. Additionally, I would like to thank him for his continuous support and help during the work on and preparation of this thesis. Furthermore, I would like to thank Prof. Dr.-Ing. habil. Boris Lohmann for chairing the Eng.D. evaluation. Prof. Dr. Sébastien Gros requires my thanks for being the second examiner, for his valuable feedback, and for the work we did together in developing the methods presented in this thesis.

A special thanks also goes towards the International Graduate School of Science and Engineering, which granted the project "skOPTing"[1] that lay the foundation of this thesis and supported the work as well as the personal development by their program. Here, I would like to thank my colleagues of the skOPTing project, Xiang Fang, Veronica Bessone, and Johannes Petrat. In addition, I wish to thank Prof. Dr. Ansgar Schwirtz for his support of the project.

Furthermore, I am grateful to Chalmers University of Technology as I had the opportunity to do a research stay at their facility that helped a lot in improving my initial work and exploit further capabilities.

Additionally, I would also wish to thank my (former) colleagues in the optimization group, Dr.-Ing. Matthias Bittner, Dr.-Ing. Matthias Rieck, Benedikt Grüter, Johannes Diepolder, Tuğba Akman, and Felix Schweighofer, for their support and the discussions we had that led to the success of this work.

Finally, I wish to thank my friends and family that supported me throughout the work on this thesis. Specifically, I want to thank my parents, who always supported me in my studies from the beginning of my Bachelor until this Eng.D..

Munich, March 2020 Patrick Piprek

[1] Supported by the Deutsche Forschungsgemeinschaft (DFG) through the TUM International Graduate School of Science and Engineering (IGSSE), GSC 81.

Contents

List of Figures

List of Tables

List of Algorithms

List of Definitions and Theorems

Acronyms

AMAN	Arrival Manager
ATM	Air Traffic Management
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CC	chance constraint
CC-OC	chance-constrained open-loop direct optimal control
CC-OCP	chance-constrained open-loop direct optimal control problem
CD	collocation defect
CI	confidence interval
CLT	central limit theorem
CoV	coefficient of variation
DEA	differential evolution algorithm
DOC	distributed open-loop direct optimal control
DOCP	distributed open-loop direct optimal control problem
EASA	European Aviation Safety Agency
ECEF	earth-centered, earth-fixed
EoM	equation of motion
FALCON.m	FSD optimal control tool for MATLAB®
FBC	final boundary condition
FFE	fixed-flat earth
FSD	Institute of Flight System Dynamics
gPC	generalized polynomial chaos
gPC-SC	generalized polynomial chaos-stochastic collocation framework
IBC	initial boundary condition
IFT	implicit function theorem
IP	interior point
Ipopt	Interior Point Optimizer
KKT	Karush-Kuhn-Tucker
LHS	Latin hypercube sampling
LQG	linear–quadratic–Gaussian regulator
LQR	linear–quadratic regulator
MATLAB®	Matrix Laboratory®
MCA	Monte Carlo analysis

MCMC	Markov chain Monte Carlo
MHA	Metropolis-Hastings algorithm
MMHA	modified Metropolis-Hastings algorithm
MPC	model predictive control
MRAC	model reference adaptive control
NED	North-East-Down
NLP	nonlinear program
NMPC	nonlinear model predictive control
OC	open-loop direct optimal control
OCP	open-loop direct optimal control problem
PDF	probability density function
ROC	robust open-loop direct optimal control
ROCP	robust open-loop direct optimal control problem
RV	random variable
RVec	random vector (i.e., a vector of one-dimensional random variables)
SC	stochastic collocation
SESAR	Single European Sky ATM Research
SNOPT	Sparse Nonlinear Optimizer
SQP	sequential quadratic programming
SubSim	subset simulation
TUM	Technical University of Munich
VTOL	vertical take-off and landing
WGS84	World Geodetic System 1984

Symbols and Indices

The nomenclature of this thesis relies upon the following principles:

Vectors are generally written with lowercase LATIN or GREEK letters in bold font, i.e., $\mathbf{a}$ or $\boldsymbol{\alpha}$. If a position vector between two points, generally depicted by uppercase letters, e.g., A and B, is considered, the two points are added to the position vector as superscript indices. The coordinate system C, also normally an uppercase letter, is added to the vector by a subscript index outside a round bracket. Additionally, a vector symbol is used for EUCLIDEAN quantities. This yields the following description:

$$\left(\vec{\mathbf{r}}^{AB}\right)_C = \left(\vec{\mathbf{r}}^{B}\right)_C - \left(\vec{\mathbf{r}}^{A}\right)_C$$

A velocity is consequently denoted by:

$$\left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^E \left(\vec{\mathbf{r}}^{AB}\right)_C = \left(\dot{\vec{\mathbf{r}}}_D^{AB}\right)_C^E = \left(\vec{\mathbf{v}}_D^{AB}\right)_C^E$$

Here, a further index D is added within the bracket, which denotes the kind of the velocity, e.g., kinematic K. Furthermore, a superscript E is added outside the bracket, which describes the coordinate frame with respect to whom the time derivative is taken.

Finally, an acceleration is defined as follows:

$$\left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^E \left(\vec{\mathbf{v}}_D^{AB}\right)_C^E = \left(\vec{\mathbf{a}}_D^{AB}\right)_C^{EF}$$

Once more, the added superscript F outside the bracket depicts the coordinate frame with respect to whom the second time derivative is taken.

Scalars are generally depicted by lower- or uppercase LATIN, e.g., a, or GREEK, e.g., α, letters in normal font and angles are depicted by lowercase GREEK letters, e.g., β.

Finally, matrices are depicted by uppercase LATIN letters in bold font, i.e., $\mathbf{A}$. If a transformation matrix is considered, the coordinate frames are added to the matrix as subscript indices. Here, the first index B represents the output coordinate frame, while the second one A represents the input coordinate frame as follows:

$$\left(\vec{\mathbf{r}}^{AB}\right)_B = \mathbf{M}_{BA} \cdot \left(\vec{\mathbf{r}}^{AB}\right)_A$$

Sub- and superscripts are used to distinguish different representations of the considered vector, scalar, or matrix. Different sub- and superscripts are separated by commas. In case of superscripts, round brackets, i.e., $(\cdot)$, indicate that the script is indeed a superscript and not a power operation.

The following listing summarizes the important symbols and indices used in this thesis:

Symbols

$\mathbf{A}$	linear state matrix
a	lower bound of probability density function
a_s	sigmoid scaling value
$\mathbf{B}$	linear control matrix
$\mathcal{B}$	Beta distribution
b	threshold for subset simulation
b	upper bound of probability density function
b_s	sigmoid offset value
$\mathbf{c}$	inequality constraints
$\bar{c}$	reference length (e.g., mean aerodynamic chord)
c_s	sigmoid shift value
C_D	aerodynamic drag force coefficient
CD	collocation defect
C_L	aerodynamic lift force coefficient
C_l	aerodynamic roll moment coefficient
BL_s	sigmoid boundary level
C_m	aerodynamic pitch moment coefficient
C_n	aerodynamic yaw moment coefficient
cov	covariance operator
C_Q	aerodynamic side force coefficient
c_V	coefficient of variation
D	order of multivariate expansion polynomial
d_{obs}	distance to obstacle
d	order of scalar expansion polynomial
$\mathbf{e}$	control/tracking error
$\mathbb{E}$	expectation operator
e	Mayer term
$\mathcal{F}$	failure event/domain
F	force
$\vec{\mathbf{F}}$	force vector
$\mathbf{F}$	objective (residual) vector for OC

$\mathcal{F}$	random class of σ-algebra
$\mathbf{f}$	state dynamics mapping
g	gravitational constant
$\mathcal{H}$	Hamiltonian
$\mathbf{H}$	Hessian matrix
h	geodetic height
$\left[h^{(m)}\right]^2$	normalization factor of orthogonal polynomial
h_τ	step size of time discretization vector
$\mathcal{I}$	indicator function
I	moment of inertia
$\mathbf{I}$	moment of inertia matrix
J	cost functional
$\mathbf{J}$	Jacobian matrix
$\mathbf{K}$	controller gain matrix
k_M	moment factor
k_T	thrust factor
k	multiplication factor
k_{sg}	sparse grid approximation level
$k_{\max}$	maximum number of iterations
kurt	kurtosis operator
$\mathcal{L}$	Lagrange function
L	Lagrange term
L	roll moment
l	arm length
M	multivariate gPC expansion order
$\vec{\mathbf{M}}$	moment vector
M	pitch moment
$\mathbf{M}$	(transformation) matrix
m_B	mass
$\mathcal{N}$	Gaussian/Normal distribution
N	yaw moment
N	number of uncertainties
$\mathbf{n}$	load factor vector
n_τ	number of discretized time steps
n_{bt}	number of backtracking steps for differential evolution algorithm
n_{pop}	population size of differential evolution algorithm
n_s	number of samples
n_{ss}	number of levels for subset simulation
$\mathcal{P}$	chance constraint feasibility domain
$\mathbf{p}$	vector of optimizable parameters

$\mathbb{P}$	probability operator
p_0	conditional probability of subset simulation
p_A	aerodynamic roll rate
PC	path/point constraints in OC
p_f	failure probability
p_K	kinematic roll rate
$\mathbf{P}_L$	Lyapunov matrix
Q	gPC-SC expansion nodes
$\bar{q}$	dynamic pressure
$\mathbf{q}$	vector of invariant parameters in optimization
q_A	aerodynamic pitch rate
q_K	kinematic pitch rate
$\mathbf{r}$	reference command
r_{obs}	radius of obstacle
r_A	aerodynamic yaw rate
r_K	kinematic yaw rate
S	reference area (projected wing area)
$\mathbf{S}$	sensitivity matrix
s	reference half span
s	smooth approximatior/sigmoid function
s	slack variable of interior point method
skew	skewness operator
T	thrust force
t	Student's t-distribution
t	time
t_0	initial time
t_f	final time
$\mathcal{U}$	Uniform distribution
$\mathbf{u}$	control vector
u	scalar control
u_K	kinematic x velocity of rigid-body
v_K	kinematic y velocity of rigid-body
V_A	absolute aerodynamic velocity
var	variance operator
V_K	absolute kinematic velocity
$W_N^{(D)}$	N-D orthogonal polynomial space
$W^{(d)}$	orthogonal polynomial space
w_K	kinematic z velocity of rigid-body
x_N	x position on flat earth
$\mathbf{x}$	state vector

$\mathbf{y}$	output vector
y_N	y position on flat earth
y	scalar output
$\hat{y}$	gPC expansion coefficients
$\hat{\mathbf{y}}$	multivariate gPC expansion coefficients
$\mathbf{z}$	optimization parameter vector
z_N	z position on flat earth
α	shape parameter of probability density function
α_A	aerodynamic angle of attack
$\alpha^{(j)}$	j-th GAUSSIAN quadrature weight
α_K	kinematic angle of attack
β	rate parameter of probability density function
β	step size of connection problem update
β_A	aerodynamic angle of sideslip
β_K	kinematic angle of sideslip
χ^2	χ^2-distribution
χ_A	aerodynamic course angle
χ_K	kinematic course angle
δ_{mn}	Kronecker symbol
δ_T	thrust lever position
ε	error of gPC approximation
ϵ_{feas}	feasibility tolerance (of nonlinear program)
ϵ_{opt}	optimality tolerance (of nonlinear program)
ϵ	tolerance
η	elevator deflection
$\mathbf{\Gamma}$	adaptation gain matrix
γ	GAMMA distribution
Γ	gamma function
γ_A	aerodynamic climb angle
γ_K	kinematic climb angle
$\mathbf{\Lambda}$	adaptive control matching parameters for controls
λ	eigenvalue
λ_E	geodetic longitude
$\boldsymbol{\lambda}$	Lagrange multipliers of inequality constraints
μ	barrier parameter of interior point method
$\boldsymbol{\mu}$	Lagrange multipliers of equality constraints
μ	mean value
μ_A	aerodynamic bank angle
μ_K	kinematic bank angle
$\boldsymbol{\nu}$	connection variables vector for distributed optimal control

ν	connection variable for distributed optimal control
Ω	random space
ω	random output of event
ω	rotor speed
ω_A	scalar element of aerodynamic rotation vector
$\boldsymbol{\omega}_A$	aerodynamic rotation vector
ω_K	scalar element of kinematic rotation vector
$\boldsymbol{\omega}_K$	kinematic rotation vector
Φ	multivariate orthogonal polynomial
Φ	roll angle
φ_E	geodetic latitude
ϕ	orthogonal polynomial
Ψ	yaw angle
$\boldsymbol{\psi}$	equality constraints
ρ	air density
$\rho_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$	multivariate probability density function
$\rho_{\Theta}(\theta)$	scalar probability density function
ρ	probability density symbol
σ	standard deviation
τ	time discretization vector
$\boldsymbol{\Theta}$	adaptive control matching parameters for states
Θ	pitch angle
θ	random parameter
$\boldsymbol{\theta}$	vector of random parameters
$\boldsymbol{\Theta}$	vector of random variables
Θ	random variable
$\theta^{(j)}$	j-th Gaussian quadrature node
$\boldsymbol{\upsilon}$	error in connection problem
ω	weight for distributed constraint
ξ	aileron deflection
ξ	confidence level
ζ	rudder deflection

Indices

A	aerodynamic frame
B	body fixed frame
CP	connection problem
E	earth-centered, earth-fixed (ECEF) frame
G	gravitational quantity
i	counter variable i

init	initial quantity
j	counter variable j
k	counter variable k
K	kinematic frame
$\bar{K}$	kinematic frame rotated by kinematic bank angle about the kinematic x-axis
l	counter variable l
lb	lower bound
LL	lower level of bi-level problem
N	geodetic / local navigation frame (earth-fixed frame describing local tangent plane of geoid)
O	reference point of N frame
O	North-East-Down (NED) frame
obs	obstacle
off	offset quantity
opt	optimal quantity
P	predicted state
R	origin of body (reference point of B frame)
sc	scaling value in OCP
sq	scaled quantity in NLP
T	total quantity
ub	upper bound
UL	upper level of bi-level problem

Chapter 1

Introduction

This thesis makes developments and proposes specifically tailored formulations of frameworks in the context of open-loop direct optimal control (OC) ("trajectory optimization") with uncertainties (i.e., non-deterministic influences in e.g., the dynamic model), which is further on denoted as robust open-loop direct optimal control (ROC). The developed ROC frameworks are applicable to a wide range of problems in the context of dynamic model optimization as they are relying on the basic principles of OC (e.g., the transcription method). The focus of the work is laid on efficiently introducing uncertainties into the open-loop direct optimal control problem (OCP) ("trajectory optimization problem"), mainly regarding the computational performance, using the method of generalized polynomial chaos (gPC). Based on this efficient uncertainty quantification within the OCP, the frameworks that are used to directly calculate robust, optimal results for the dynamic model are developed. These rely on distributed open-loop direct optimal control (DOC) and chance constraints (CCs) (i.e., probabilistic constraints) The robust trajectories, i.e., the optimal trajectories that are insensitive to the uncertainties influencing the dynamic model, are then resulting from the optimization procedure.

A general motivation on the necessity to calculate robust trajectories and the resulting problem statement is given in Section 1.1. Then, the state of the art is reviewed in Section 1.2, while the emerging mission statement is detailed in Section 1.3. Afterward, Section 1.4 summarizes the resulting contributions of this thesis. The chapter concludes with an outline in Section 1.5.

1.1 Motivation and Problem Statement

Optimization plays an important and immensely growing role in our society and economy mainly because of fading resources and growing competition. Nowadays, optimization is used in almost every major engineering discipline such as civil engineering (e.g., life-cycle analysis of roads and buildings), control engineering (e.g., controller design), or electrical engineering (e.g., filter design). Apart from engineering applications, optimization also plays a major role in e.g., economics and finance (risk-averse profit optimization). These

fields show that the application of optimization techniques is very important to ensure the competitiveness and longevity of companies, but also to maximize the environmental and economic sustainability.

The goals of competitiveness and sustainability are also of special importance within the aviation industry as airlines must operate as efficient as possible considering e.g., the flight time and fuel consumption as well as the grounding hours and turn-around times. Additionally, the market is highly competitive as there are many airlines to choose for the customers. Thus, methods that e.g., optimize the fleet scheduling, flight paths/-trajectories, aircraft design, engine control, and the fuel consumption have always been an important topic in research. Especially reducing the fuel consumption was consistently a major research area that led to the development of e.g., supercritical airfoils and winglets by Richard Whitcomb [20]. Furthermore, new materials, such as carbon, are introduced in the aircraft design to reduce the overall mass, while also trying to increase the payload.

Regarding optimization, the increase in the available onboard computational power allows algorithms that optimize parts of the flight online, e.g., a time or fuel optimal climb segment, to become applicable and makes them an integral part of future aviation [39]. Additionally, not only the optimization of a single aircraft will be important in the future, but also the optimization of multiple aircraft within connected problems: An interesting example in this field is the approach of aircraft to an airport, where a strict assignment of the aircraft into an arrival sequence is necessary. Here, no single aircraft should be put at disadvantage (e.g., delayed arrival or increased fuel consumption), yielding the necessity for optimizing the sequence as a whole.

The questions and chances related to aircraft optimization utilizing the increased onboard computational capability are combined as a core part of Europe's Flightpath 2050 program to achieve a safe, secure, and connected European air transport system, while simultaneously protecting the environment by reducing the emissions [39]. A major task of this program is done in the Single European Sky ATM Research (SESAR) framework that tries to modernize the European Air Traffic Management (ATM) system resulting in a "free" airspace: The idea of this "free" airspace is that all air vehicles (e.g., general aviation, helicopters, or air taxis) operate on their preferred flight paths ("trajectories") while not being restricted by airspace limitations (e.g., flight levels), except for safety considerations [107]. Initial tests in the SESAR framework show the significant emission reduction capability, when the flight is optimized as a whole rather than segment-wise as currently done [108].

Especially, this planning as a whole is one of the strengths of the OC methods used in this thesis: They provide the capability to plan the flight trajectory with respect to an optimization criterion (e.g., reducing the emissions or flight time), while also considering the operational aircraft limitations (e.g., passenger comfort and load factor limits). These optimal trajectories from OC theory can then be used within the developed SESAR ATM

system (e.g., for the already mentioned sequencing). Additionally, OC methods can be applied in the context of air taxi route planning to fully utilize the capacity of the air space. Furthermore, it is also important in the air taxi route planning to consider an appropriate sequencing and thus, the mentioned ATM systems are applicable as well. Thus, OC theory has a broad application range in future aviation. It should be reminded here, once more, that the term OPTIMAL CONTROL refers to OPEN-LOOP DIRECT OPTIMAL CONTROL in the context of this thesis, i.e., no feedback is considered and the solution is obtained by solving a nonlinear program (NLP), i.e., a numerical parameter optimization problem.

To expand the available tools in creating robust, optimal trajectories, this thesis concentrates on calculating these for aircraft (as they would e.g., be used in the context of FLIGHTPATH 2050 in an Arrival Manager (AMAN), which optimizes the arrival sequence at an airport), by using the OC theory with add-ons to consider uncertainties. Here, time as well as fuel optimal trajectories can be calculated to fully exploit and increase the efficiency of the aircraft's flight performance. In this case, it is clear that a trade-off between time and fuel optimality must be made. Additionally, the standard OCP formulation does not consider uncertainties (an uncertainty in the context of this thesis is a system ("dynamic model") influence ("parameter") with known probability density function (PDF)), such as wind, model (e.g., mass), as well as aerodynamic (e.g., lift coefficient) parameters. These can significantly alter the optimal flight paths for both time and fuel optimality. The jet stream the transatlantic flight routes is an example that can be considered as an uncertainty, which can significantly alter the optimal aircraft routing. In the context of the AMAN, an aircraft in an emergency situation, i.e., a high priority in the arrival sequence, can be also considered as an uncertainty. Furthermore, the already mentioned air taxi routing is normally subject to a number of uncertainties: Here, not only the environment (e.g., wind fields) can be considered as such, but also other air taxis, which require an update of the routing to avoid collisions.

Disregarding these uncertainties yields that optimized trajectories, obtained from standard OC theory without uncertainties, are very specific and only valid locally around the design point. This design point is normally the point without uncertainties (i.e., the "mean/nominal" configuration). Here, in reality even small deviations (e.g., disturbances by wind) from the design point can lead to quite large deviations in the real system behavior, when operated with the result obtained for the design case. This deviated system behavior might then no longer be optimal but rather inefficient, although it was originally planned to be an optimal scenario. Additionally, constraints (i.e., operational limits of the system) might be violated in the deviated ("disturbed") case that were fulfilled for the nominal solution. This can then create safety critical situations during the real operation.

Thus, uncertainties have to be considered for the calculation of optimal trajectories and within the formulation of the OCP. This gives an idea on how the real behavior of the system changes with these uncertainties and can then be used for a re-evaluation

of the optimization to get more robust results. Additionally, robust results can also be obtained by directly calculating trajectories that are insensitive to (i.e., only influenced "mildly" by) the uncertainties. Modeling the OCP with uncertainties and finding optima that are least sensitive, and thus, robust with respect to uncertainties, is the main goal of this thesis. For this, the next section reviews already conducted studies in the field of optimization with uncertainties.

1.2 State of the Art

This section gives an overview on the state of the art of solving OCPs with uncertainties. Generally, the basic OCP is solved in the presented studies, and also this thesis, by using direct OC methods, i.e., by discretizing the continuous (i.e., infinite dimensional) OCP into a large-scale, finite-dimensional parameter optimization problem, which is subsequently solved using an NLP solver [16]. Here, the NLP algorithm numerically searches for the optimal point by NEWTON-type optimization schemes [5, p. 294]. The state of the art review begins with a general overview of methods to quantify uncertainties within a dynamic system.

Uncertainty Quantification: Overview Uncertainty quantification is a popular topic within the engineering research community. Its origin is the Monte Carlo analysis (MCA), i.e., the random sampling and deterministic simulation of dynamic systems to get their response surface [89]. As MCA is inefficient, especially in the context of OC because the solution of an OCP is time-consuming and must be done thousands of times for a MCA, there have been multiple improvements and developments to calculate the uncertain system response more efficiently.

At first, stochastic control was an important topic in order to calculate optimal robust gains for control loops [6]. Here, analytic results for the optimal gains of linear systems, e.g., the linear–quadratic–Gaussian regulator (LQG), i.e., a linear system with GAUSSIAN process and measurement noise, were initially calculated [6].

Later on, robust optimization was extended using (stochastic) dynamic programming approaches, which split a large, complicated problem into smaller sub-problems solved in a recursive manner to simplify the solution process [13]. For instance, this recursive nature of dynamic programming is used in combination with a Markov chain to find the maximum likelihood estimates of transition probabilities in study [66]: Here, the authors use their approach to achieve robust results with respect to the limited knowledge available for the transition probabilities of the Markov process (i.e., a process that only depends on the current and no previous states). It should be noted that a basic problem with high-dimensional models in dynamic programming approach is the high computation effort

and storage requirement when solving the problem recursively. Thus, the methods are normally not considered in applications with complex dynamics and large state spaces ("curse of dimensionality"), which are present in the aviation studies of this thesis.

In general, the incorporation of uncertainties within dynamic systems became very popular with the introduction of gPC by Xiu and Karniadakis in 2002 [143]. The gPC method provides a fast, accurate, and efficient way to approximate the response surface, and by this the statistical moments, of an uncertain dynamic system. It is based on Wiener's polynomial chaos [135], published in 1938, where Norbert Wiener introduced a polynomial expression for systems subject to Gaussian uncertainties using Hermite polynomials. The gPC method extends this principle to polynomials from the Wiener-Askey scheme of orthogonal polynomials [143]. Generally, the gPC expansion is similar to the well-known Fourier analysis of an oscillating system [5, p. 1043ff.]. Thus, gPC creates a finite, generally low order, expansion of the dynamic system response to uncertainties. As this thesis concentrates on the application of gPC in OCPs, the literature overview starting from now is mostly dedicated to work on this topic.

As already mentioned, gPC became a major contributor to the scientific community in terms of accurate, efficient, and fast uncertainty influence characterization. By this, the calculation of statistical moments for dynamic systems becomes possible. It was, for instance, used within the following engineering applications: In fluid dynamics, gPC is used to model the uncertainties, e.g., in wall boundary conditions and free stream conditions, within a flow field [144]. Here, the authors use a Galerkin projection [5, p. 1094] to calculate the gPC expansion and show the improved efficiency by their method compared to a MCA. Another study [134] uses gPC to model uncertainties within steady as well as unsteady three-dimensional heat transfer problems. Again, the Galerkin projection is used to solve the problems. The authors show results for up to 38 stochastic dimensions (i.e., uncertainties) as well as 150 million unknowns in their problems. This shows the applicability of gPC for large-scale problems, which are also present in OC.

Furthermore, the propagation of uncertainties within spacecraft attitude kinematics is discussed in [129]: Here, the authors combine gPC with Gaussian mixture models (i.e., a combination of multiple Gaussian PDFs). By this, the authors can describe different continuous PDFs by the standard Gauss-Hermite gPC expansion. This extends the basic gPC theory to PDFs not covered by the Wiener-Askey scheme. Additionally, they show a good convergence of the gPC expansion even for low gPC expansion orders.

Finally, the highly unsteady flow within an internal combustion engine is combined with gPC in [149]. Here, the uncertainties within the combustion chamber are modeled by gPC and the response surface results of the combustion simulation are compared to MCA. As an outcome, gPC is shown to be more efficient than MCA, while also reaching a good match with experimental results.

Overall, the literature review shows that gPC is a very important and also versatile method in engineering applications. Therefore, it is suited to develop the algorithms in this thesis that deal with the calculation of robust and optimal trajectories.

As a remark, it should be noted here that gPC has a fundamentally different philosophy compared to the stochastic dynamic programming approach mentioned in a previous paragraph: The idea of gPC is, essentially, to have an uncertainty that is once "randomly" defined and otherwise constant (i.e., time-invariant; "parametric uncertainty") along the system trajectory (e.g., the initial mass of an aircraft). Therefore, gPC looks at a single perturbation of the dynamic system. Stochastic dynamic programming approaches, however, consider an uncertainty that is varying along the trajectory, i.e., it is time-variant (e.g., measurement noise). Thus, the uncertainty can randomly change along the trajectory. Consequently, the control policies obtained from gPC and stochastic dynamic programming-based OC solutions will differ in application as the basic solution procedure differs. Nonetheless, gPC methodologies could be extended to time-varying uncertainties (simply speaking by time-varying the uncertainty along the trajectory and solving the gPC problem at each time instant individually), which would then consequently also suffers from the curse of dimensionality encountered by dynamic programming approaches. As the single perturbation case covered by gPC is sufficient for the work in this thesis no further details and extensions regarding this topic are discussed.

Uncertainty Quantification: Optimization One of the most fundamental problems in ROC (in this case in the context of closed-loop ROC) with uncertainties is the LQG problem to design the optimal feedback controller gain for a linear dynamic system ("plant") that minimizes a cost functional [6, p. 257ff.]: Here, the well-known linear–quadratic regulator (LQR) baseline problem formulation [122, p. 397ff. & 470ff.] is extended by adding Gaussian white noise to the system dynamics as well as the measurement equation. The general problem formulation reduces to an optimal estimation problem as well as an OCP that are separable and can be solved by two time-dependent Ricatti equations [6, p. 257ff.]. Within this thesis, methods for nonlinear systems as well as more general cost functionals, in mainly an open-loop OC setup, are developed. Still, the LQG problem is an important milestone and the idea of feedback controller design with uncertainties is extended within this thesis as well.

As mentioned, this work concentrates on the application of gPC in trajectory optimization problems to calculate robust optimal trajectories (now again meant in the open-loop OC context). These robust, optimal trajectories should be sufficiently insensitive to the chosen parametric uncertainties. In previous applications, the gPC expansion was often used as an add-on to the standard OCP formulation. This work, on the other hand, tries to find a suitable combination of both standard OC and gPC for an integrated solution. The approach can be regarded as a primal composition of the two major parts of ROC, i.e., the uncertainty quantification as well as the direct OC discretization method.

Previous studies on gPC within the OC field comprise the following topics: In [30], the author solved OCPs using the Gauss pseudo-spectral discretization method. The optimization did run for multiple samples ("nodes/evaluation points") calculated from the uncertainty's PDF by the rules of Gaussian quadrature [5, p. 1184], in sequence and is afterward combined to the gPC expansion (this procedure is also called: "non-intrusive method"). Thus, the OCPs are decoupled and optimized deterministically and an uncertain representation is derived. The developed algorithm is applied to a target search with threat areas and does generally not yield robust trajectories, but only uncertain trajectories. Building upon the previous work in [30], study [58] introduces a result check methodology to compare the generated trajectories from the analytic gPC expansion with OC results at random parameter samples of the uncertainties. Here, the author compares the system response when simulating the system forward in time based on the calculated controls from gPC. The author uses this methodology to check the applicability and viability of the gPC results with varying parameters and found the results to be a good match with a sufficient gPC expansion order.

Furthermore, study [19] introduces the gPC methodology within differential dynamic programming, which is a development of dynamic programming and uses locally-quadratic models in the recursive dynamic programming framework. In this context, the authors introduce variational integrators (i.e., momentum conserving integrators specifically designed to numerically integrate the Hamiltonian system [126, ch. 1]) to increase the numerical stability of the algorithm. Then, the work in [86] uses the gPC expansion to calculate the conflict probability of aircraft in ATM scenarios. The authors incorporate gPC within a Gauss pseudo-spectral transcription method and calculate 4D conflict resolving optimal trajectories. The gPC method is used as an efficient way to estimate the conflict probability for these scenarios and the authors evaluate the performance and effectiveness of their algorithm, which shows good results.

Study [80] formulates a large scale, deterministic OCP using the non-intrusive gPC expansion for a supersonic aircraft climb. The authors transform the general stochastic OCP into a robust, deterministic OCP. The solution of the OCP is, again, based on the Gauss pseudo-spectral method and the authors consequently also use these trajectories to calculate the expansion coefficients of the gPC within their robust open-loop direct optimal control problem (ROCP). Additionally, the authors compare their results with the results based on the algorithm given in [30], which, as mentioned before, merely calculates an uncertain representation of the system by evaluating specific samples of the system. Additionally, they show the improved robustness by their approach.

Furthermore, the authors of study [94] use a similar approach as in [80] to optimize aircraft trajectories for landing in severe weather conditions. Again, a Gauss pseudo-spectral method is applied to discretize and solve the OCP. The authors rewrite the stochastic OCP into a deterministic baseline OCP using the non-intrusive gPC expansion and verify their method by comparison with a MCA, which shows the increased performance.

Additionally, study [139] introduces an incorporation of gPC in robust design optimization. Due to the high dimension of the uncertainty space, the author introduces sparse grid techniques for the non-intrusive gPC formulation. To achieve a robust design, the author suggests the use of a sensitivity-based measure instead of the commonly used variance-based measure. This approach ensures that the uncertainty influence on the optimal design point is reduced. Here, the approach utilizes the fact that the gPC expansion provides an (approximate) analytic representation of the uncertain system response.

Finally, study [109] decomposes the ROCP formulation by using a bi-level OC framework for noise minimal approach trajectories. The upper level problem consists of parameters for a BÉZIER CURVE (i.e., a spline curve) to shape the horizontal flight path in order to decrease the noise level. The lower level contains multiple deterministic OCP subject to wind uncertainty. Post-optimal sensitivities [45] are used to update the upper level parameters efficiently. This bi-level approach is also known as primal decomposition method.

Robust Controller Gain Design by Optimization Techniques As already mentioned, optimal gain design for controllers (here in the context of feedback gains) has always been an important topic in general closed-loop control but also in OC theory. Again, the LQR and LQG controller design are important steps in the development. Furthermore, looking at adaptive control, the choice of the adaptation gains (e.g., in model reference adaptive control (MRAC)) is of paramount importance for the stability and performance of the control loop. This is studied in [124] and it is found that depending on the range of the uncertainties, different gains are suitable regarding performance and stability. Therefore, a robust, optimal design of the gains is also of interest and within the scope of this thesis. Take into account that the following literature overview gives an insight into gain design methods using optimization techniques. Thus, it should not serve as an overview to general robust gain design in control loops, but to methods of interest for this thesis.

For standard (i.e., non-adaptive) control loops, the following studies have been conducted: In study [117], the optimal gain design for linear systems with time delay is researched. For this, the authors consider a parametric optimization on the largest absolute eigenvalue of the linear mapping matrix (named like this because of the time delay) and minimize it by varying the parameters. They show the robustness improvement by their method in case studies and by root locus analysis. On the other hand, study [70] uses a particle swarm optimization, i.e., the evaluation of multiple sets of controller gains and a successive "push" of the gains into the direction of the optimum, to calculate the optimal feedback gains for a direct current motor. The authors use the particle swarm to trade-off the objectives of minimum rise time vs. minimal maximum overshoot vs. minimal settling time and compare their results with standard tuning methods to show their superiority. Finally, [120] uses unscented transformations and a performance-based

controller design methodology. This method has the benefit that the design with respect to uncertainties does not need to be conservative and closed-loop statistical performances can be guaranteed.

Particle swarm optimization is also used in MRAC adaptation gain calculation [1]: In this study, the MRAC reference models are switched based on the current operating point. The adaptation gain of the MRAC structure is chosen using the particle swarm based on a tracking error minimization in a pre-processing step for reference cases. The authors achieved an improvement in transient performance compared to standard MRAC.

Furthermore, study [93] introduces an online optimization method for the adaptation gain based on model predictive control (MPC) for a class of input-affine dynamic systems with single input and single output. The optimal adaptation gain is calculated by minimizing the control energy. The authors additionally take precautions to stay within the feasible domain by means of LYAPUNOV analysis [121, ch. 3]. Finally, the research presented in [3] suggests a further online MRAC scheme that uses two NEWTON-type optimizations: One to calculate an appropriate control signal and another one to update the adaptation gain. Both optimizations yield a better tracking than the reference adaptive control loop, by minimizing the error between plant and reference model.

Robust Model Predictive Control Robust MPC is an important research topic as deterministic MPC algorithms can largely benefit from robustifications (e.g., in the context of quality and stability of the results). Generally, it can be stated that MPC is a development of OC regarding online applicability of the optimization algorithm as well as online control capability, in an optimal sense, on a receding horizon in a closed control loop. The studies within this field are therefore also relevant for the developed open-loop robust trajectory optimization frameworks in this thesis. Additionally, the developed methods in this thesis can be extended to MPC or nonlinear model predictive control (NMPC) in future works. Therefore, some standard methodologies are looked at in the following paragraphs. A general overview on robust MPC (linear system)/NMPC (nonlinear system) and its basics can be found in [14]. Further overview on more recent methods is given in [87].

A very popular choice for robust MPC are so-called min-max algorithms (related to the "tube-based" algorithms), which try to achieve a worst-case optimal control command design in the presence of uncertainties to increase the robustness [33, 83, 128]. Generally, the min-max algorithms (most often done in the open-loop context because of a too high complexity in the closed-loop) try to find the worst-case ("max") plant reaction to the uncertainties (which is most often at the boundary of the uncertainty domain) and then optimize ("min") the control policy such that it minimizes the cost functional for this worst-case plant reaction. Overall, min-max algorithms normally yield conservative control policies. In [128], the authors describe a min-max differential inequality describing the forward propagation of the system within an invariant tube. The proposed approach

is generally applicable to tube-based MPC methods, i.e., MPC methods that try to keep the disturbed trajectory in an invariant tube around the design/nominal solution. Additionally, it does not rely on a discretization of the control policy, which is instead implicitly defined by the solution of the min-max optimization problem. It should be noted here that tube-based MPC schemes often also introduce an ancillary controller that keeps the deviation between the desired and actual state in an invariant tube and thus, keeping the MPC scheme stable in the presence of uncertainties [127].

Furthermore, study [33] deals with the development of a min-max algorithm that is suitable for linear constrained systems with piecewise affine cost functionals. The algorithm results in a robust dynamic programming problem. Study [83] introduces a quasi-min-max algorithm (the authors use the word "quasi" because the first stage cost can be computed without any uncertainty) for linear systems with parameter variations. The approach is specifically meaningful for gain scheduling applications within controllers. Then, study [127] describes the computation of ellipsoid tubes for NMPC that approximate the min-max MPC problem. It is closely related to study [128] and benefits from the fact that the control policy does not need to be discretized. The authors illustrate their method using a spring-mass-damper system.

Additionally, authors also already incorporated gPC within MPC [72, 73]: In study [72], the authors apply a gPC algorithm for probabilistic collision avoidance. Therefore, they implement a CC ("probabilistic constraint") and use the gPC method to approximate the probability of exceeding this CC within the MPC scheme. Furthermore, study [73] gives a more general algorithm for the MPC with an additive Gaussian process. This Gaussian process is again described using a gPC expansion, which yields a relaxation of the probabilistic CC into a convex, deterministic constraint that can be solved using a standard MPC algorithm.

Finally in study [11], the authors use gPC and CCs for the uncertainty propagation within NMPC. They use Bayes' rule to estimate the probability distribution of the uncertain system response at each sampling point and apply only Gaussian PDFs to solve a deterministic MPC OCP with joint CCs.

Distributed Optimization A powerful tool in standard OC is the method of DOC: It defines a methodology to split up a large OCP into smaller OCPs that are coupled by connection variables without making simplifications on the originally desired solution of the non-distributed OCP. These distributed OCPs are then easier to solve because they are smaller (i.e., have a smaller number of constraints and optimization parameters) and can be solved independently of each other (i.e., they are parallelizable). As a drawback, the smaller OCPs must be solved multiple times, instead of only once as a coupled problem. Still, it is generally easier regarding both computational time as well as complexity, to solve multiple smaller, parallelized OCPs instead of a single, large, and coupled OCP (which might not even be possible due to the size). Thus, this study applies the method

of DOC to ROC with gPC. The resulting OCPs can be distributed well due to the linear nature of the gPC expansion. A general overview on different methods for DOC is given in [88], while the following overview introduces initial work on DOC with gPC.

Developed gPC-DOC frameworks for MPC, and generally with intrusive changes to the dynamic model by inserting the gPC expansion, can be found in [32, 37, 67]: At first, study [32] shows a combination of DOC in the context of MPC with gPC. The authors apply their method to linear systems and use an intrusive gPC formulation to rewrite the deterministic equations into a stochastic form, i.e., the authors insert the gPC expansion in the model equations and rewrite them (this is also called: "intrusive method"). Their goal is to solve a linear probabilistic CC. It should be noted that rewriting dynamic equations is opposite to the formulation of this thesis that conserves the deterministic dynamic system formulation and thus does not require and alteration of the original dynamic model formulation.

Furthermore, study [67] introduces a similar idea as study [32] in the context of stochastic DOC for non-convex problems. The authors again use an intrusive reformulation by the gPC method to change the problem formulation from the deterministic to the stochastic domain. Finally, study [37] once more uses the gPC method and directly inserts it into the DOC formulation for optimization. The authors show that box constraints can be fulfilled with a high accuracy using the gPC method.

Chance Constraints in Optimal Control The formulation of an OCP in the form of a chance-constrained open-loop direct optimal control problem (CC-OCP) is a frequently used formulation in ROC. Still, these formulations remain computational expensive and difficult to solve exactly. The following research has been conducted within the field: In study [130], a general overview of different methods for handling CCs ("probabilistic constraints") is given. The author introduces both analytical methods (e.g., ellipsoid relaxation, i.e., approximating a multivariate Gaussian distribution in the OCP by a cone constraint) and sampling-based methods (e.g., mixed integer programming, i.e., optimization with a cost function, constraints, and (some) integer optimization parameters). These methods are subsequently combined in a hybrid approach. Additionally, a feedback controller is used to satisfy the system constraints. The author introduces an approach to assure feasible solutions by making a risk allocation, i.e., the violation probability of each individual CC is an optimization parameter. These are then summed and constrained by the total violation probability, in a two-stage optimization procedure. In addition, the author shows the applicability of the algorithm in a real-time experimental demonstration.

Furthermore, study [147] introduces a strategy to approximate a CC based on split Bernstein polynomials in a Gauss pseudo-spectral OC framework. This approximation is combined with a Markov chain Monte Carlo (MCMC) algorithm that estimates the expectations of tabulated samples to solve the CC-OCP. Additionally, study [95] introduces

another method for chance-constrained open-loop direct optimal control (CC-OC) using a mixed control strategy. This strategy is a convexification of the original, non-convex CC-OCP. The authors show that two deterministic control strategies are sufficient to achieve robust optimality. Study [25] then introduces a CC-OC framework that is based on Kernel density estimation, i.e., approximating the PDF by basis functions. The approximated PDF can directly be used to estimate the corresponding probability of fulfilling (or violating) the CC. The authors show that their approach is also working for only a small number of available samples.

As already mentioned, CCs are often applied in MPC/NMPC applications. Many approaches apply the already introduced tube-based/min-max theory [33, 83, 128]. Additionally, in order to assure real-time applicability the algorithm, study [50] transforms GAUSSIAN uncertainties to explicit algebraic constraints ("ellipsoid relaxation"). The authors apply their method to spacecraft rendezvous calculation and show that the mission cost is increased by the robust scheme but that the success of the mission can be assured. It should be noted that this trade-off is most common in robust OC applications.

Furthermore, frameworks in CC-OC use the randomization technique to calculate the robust control policy [146]. This technique is generally a MCA method and thus, it is very important to find suitable samples for the evaluation in order to reduce the overall number of MPC problems that need to be solved. Overall, this randomization can then directly be used to estimate the probability of the CC.

Finally, methods of MPC with CCs use feasible set approaches that can guarantee to satisfy the probabilistic constraint [116]. The general idea is to find a maximal set of feasible control policies that all satisfy the CC. Here, this set can be calculated by an optimization algorithm, specifically, by running multiple optimization for different parameter combinations.

Subset Simulation in Optimization In order to deal with rare-event CCs (i.e., events with a small occurrence probability), as also desired for the frameworks developed in this thesis, the method of subset simulation (SubSim) can be applied [7, 9, 81]. This methodology starts the rare-event probability estimation process with a general MCA solution of the OCP and afterward gradually exploits different samples within the failure domain(s), using e.g., a Metropolis-Hastings algorithm (MHA) or modified Metropolis-Hastings algorithm (MMHA). This so-called MCMC algorithm converges to a series of conditional probabilities, which are easier to calculate than directly working on the rare-event, but nonetheless yield the failure probability of the desired rare-event.

As SubSim is very often used in reliability engineering, optimization frameworks of interest for this thesis are also developed in this area: In study [71], SubSim is incorporated in a risk-cost optimization setup for pipelines to minimize the failure occurrence. The authors show that the proposed algorithm is computationally more efficient than standard techniques based on genetic algorithms.

Further work is conducted by introducing SubSim into design optimization [77]: Here, SubSim is combined with a genetic algorithm including constraint handling. The basic idea is that optimal points are rare events, which can therefore be calculated by SubSim. The authors show the efficiency and robustness of the proposed approach compared to standard techniques (e.g., basic genetic algorithms).

Additional work also combines SubSim with global, unconstrained optimization problems [76]: Here, design variables are assigned an artificial PDF and can therefore be treated with SubSim. Then, the maximum of the now probabilistic objective function is searched for. Again, benefits in comparison to a standard genetic algorithms are shown.

Finally, study [123] shows the viability of SubSim in the context of multi-objective optimization. Here, the SubSim provides a fast and reliable way to explore the Pareto optimal region and generate the Pareto frontier. The applicability of this approach is shown in multiple case studies and shows good results compared to standard methods from literature based on genetic algorithms.

Continuing after the general literature review and the state of the art overview, the next section deals with the mission statement of this thesis considering the state of the art in robust, optimal trajectory calculation. Here, the developments by this thesis to the current situation are stated.

1.3 Mission Statement

Looking at the state of the art and literature review in Section 1.2, it is clear that uncertainty quantification is an important topic in engineering applications. But, although major effort has already been put in developing new and efficient methods, especially the open-loop OC formulation with uncertainties is still not fully exploited and researched (e.g., in terms of modeling the uncertainties within the OCP and the NLP transcription as well as an efficient evaluation routine of the resulting probabilistic problem). In this context, it is important to note that the goal of the work at hand is to represent the uncertainties by their real PDF and influence on the dynamic system, rather than making simplifying assumptions or approximations (e.g., ellipsoid relaxation).

To this end, there also has not yet been any deep analysis on rare-event failure probabilities in CC-OC. Still, these rare-event failure probabilities are of major importance in the context of future aircraft certification that is based on fulfilling probability criteria [40–42]. These probabilistic criteria must fulfill high safety standards and thus, low failure margins, where the margins can be achieved using suitable probabilistic constraints in OC methods.

Additionally, due to the already mentioned increase in onboard computational power, the online OC of e.g., commercial aircraft, becomes feasible. Here, considering uncertainties will be a prominent topic to achieve robust, and therefore, safe as well as optimal trajectories. In future applications, this can be combined with gateway controller archi-

tectures that are checking the results obtained from ROC for sanity and validity (i.e., is the result feasible and does it fulfill the safety criteria). If the result is valid, the robust, optimal trajectory can be commanded to the aircraft for trajectory following purposes. Otherwise, fallback strategies are used, which might be based on post-optimal sensitivities or simplified analytic representations. These ensure that the aircraft follows a sub-optimal, but feasible trajectory until the ROC again returns valid solutions. Thus, efficient methods in the context of formulating and solving ROCPs, specifically for the online applicability, ensuring the availability of sensitivities (i.e., first order updates of the system for perturbed parameters), must be developed. Here, not only the already mentioned CCs, but also DOC algorithms with uncertainties can prove viable to achieve fast and more reliable convergence, and thus require further research.

Therefore, the main mission of this thesis is to evolve the theory and available frameworks in ROC: Here, it must be considered that an efficient uncertainty quantification is required in OC formulations and that future applications require probabilistic constraints, especially also for rare-events, which should also be covered by the developed methodologies without simplifying assumptions.

1.4 Contributions of this Thesis

Following the mission statement in Section 1.3, the contributions of this thesis can be summarized as follows:

C. 1 Development of a full discretization ("collocation") transcription for a direct OCP using gPC within the FSD optimal control tool for MATLAB® (FALCON.m) framework (Chapter 5): This direct collocation method reformulates the baseline deterministic OCP into a larger, parallelized OCP using a non-intrusive gPC formulation, specifically, the stochastic collocation (SC) method (Subsection 2.3.3). By this, the gPC description, and thus the uncertain description of the trajectory, can directly be incorporated within the OCP and the statistical moments can be optimized. This method is specifically tailored to the requirements of the trapezoidal collocation discretization method and the FALCON.m software package to solve OCP that was developed at the Institute of Flight System Dynamics (FSD). An important aspect of the developed gPC collocation scheme is the derivation of an analytic Jacobian and Hessian that is used to efficiently solve the created NLP and can be used to calculate sensitivities, which can be applied in online applications.

C. 2 Calculation of optimal gains for adaptive or standard control loops, combining the method of gPC, the system response to reference commands, sensitivities, and CCs within a bi-level OCP (Chapter 3). Using this methodology the statistical moments are again part of, in this case, the bi-level OCP and gains for the adaptive or standard controllers can be calculated that are robust and least-sensitive to a variety

of uncertainties acting on the system. Additionally, these gains can fulfill constraints in the form of probabilities that can e.g., be defined by certification regulations in future applications. Here, a major part is also the combination of gPC and sensitivity analysis for a fast calculation of the controller gains. It should be noted that, although applied here in the context of gain design, the proposed method is more generally applicable to any parameter optimization required in a system subject to uncertainties (e.g., design parameters). This is also clarified when introducing the general methodology in Chapter 3.

C. 3 Development of a framework to distribute the gPC expansion calculation from one large OCP into smaller, parallelizable, and more efficiently to solve OCPs (Chapter 4). These small OCPs are initially solved unconnected and are then connected within a coordination/connection problem (this is similar to the bi-level OCP in Contribution 2). Here, the size of the solved ROCP is significantly reduced as the subproblems are smaller and can be distributed to different workers (i.e., parallelized). Additionally, an efficient formulation for the statistical moments is developed that can be used to the solve the coordination problem without the necessity to have sensitivities from the sub-problems. Still, it is important to note that the distributed open-loop direct optimal control problems (DOCPs) recover the original solution of the connected, large-scale OCP without simplifications, which is achieved by choosing appropriate connection variables ensuring this behavior.

C. 4 Development of a framework to incorporate CCs within an OCP using the developed gPC OC transcription in Contribution 1 (or the DOC in Contribution 3; see Chapter 6). This CC-OC framework is based on the efficient sampling that is possible from the gPC expansion, such that the probability of fulfilling the CC within a constraint function (along the trajectory: "path function"; at specific points on the trajectory: "point function") can be calculated if the gPC expansion is available within the OC formulation. It should be noted that the real system response, at least if a sufficiently accurate gPC expansion is available, is sampled in this context. Thus, no conservatism (by simplifications or assumptions) is introduced into the OCP, which generally yields a very good trade-off between robustness and optimality. Additionally, a method to approximate the sharp CC bound for the NEWTON-type NLP algorithm by a differentiable function is developed. Furthermore, a homotopy strategy is implemented to gradually approximate the original, sharp CC as accurate as possible. This combined direct gPC transcription and CC formulation can also prove viable for MPC applications with probabilistic constraints.

C. 5 Combination of SubSim with the CC-OC framework to be able to calculate the probability of rare event failures (e.g., failure probabilities in the range of $10^{-6} - 10^{-8}$ and smaller; see Chapter 6). Here, the MCMC algorithm of SubSim is incorporated within the already developed CC-OC framework. Therefore, a homotopy strategy

can be used to gradually decrease the failure probability constraint until the desired probability value is fulfilled. Additionally, direct updates of the SubSim within the OCP by sensitivities or in an external homotopy step can be considered. The general goal of this contribution is to define the rare-event CC-OC framework such that the deterministic solution of the ROCP is conserved. This is done to ensure good convergence as well as to be able to use off-the-shelf NLP solvers. Overall, this contribution is a step towards connecting trajectory optimization with safety-critical trajectory planning/design.

In summary, the major contributions of this thesis are in the field of connecting gPC with OC to form general frameworks for ROC. These frameworks are specifically viable for probabilistic events, including also rare-events. Thus, optimality and robustness are connected within ROCPs that generally do not introduce additional conservatism when solved (as e.g., by the previously mentioned relaxation techniques or worst-case approximations). An important aspect of the proposed framework is that adaptations of the deterministic dynamic model formulation are not required, which allows for the use of already tested and validated model structures. Additionally, the developed distribution capability can be exploited to efficiently calculate robust results without making simplifications on the solved problem, which is a further development to the state of the art. Finally, in specific the introduction of a method to evaluate rare-event CCs in ROC is an important development to the state of the art made by this thesis.

1.5 Structure of this Thesis

As stated in the previous four sections, the main goal of this thesis is the development of trajectory optimization frameworks that are, first of all, able to incorporate uncertainties and, second of all, able to optimize the trajectory in a way that the uncertainties have a reduced influence on the results. This means that the trajectory should be ROBUST with respect to the incorporated uncertainties. In order to show the implementation and viability of the developed robust trajectory optimization/ROC frameworks, the thesis is organized as follows:

In Chapter 2, the theoretical background on general and advanced OC theory as well as general and efficient uncertainty quantification methods is established. Here, Section 2.1 starts with introducing the general OCP and the direct OC methods used for solving the problem within the FALCON.m framework. Afterward, Section 2.2 introduces classical methods of uncertainty quantification, e.g., MCA, that are mainly used to compare the implemented algorithm viability. Here, also the SubSim as a method to calculate rare-event failure probabilities applying the principles of the MCMC algorithm is introduced. Furthermore, Section 2.3 introduces the gPC method that is used in this thesis to incorporate uncertainties within the OCP. Here, the main points are the introduction of the general expansion formula, the SC method that allows for a deterministic sampling

of the basic OCP to calculate the uncertain system response, as well as the fast and easy statistical moment calculation. The theory chapter then concludes with some advanced optimization techniques such as (post-optimal) sensitivity analysis, DOC, and CC-OC in Section 2.4 that are important for the ROC frameworks developed in this thesis.

After the theoretical introduction, Chapter 3 introduces the first framework developed for ROC in this thesis: This framework is based on a bi-level setup, in which an upper level solves a parameter optimization problem, while the lower levels calculate the gPC expansion of the system based on OC. The main feature is the calculation of the statistical moments not only for the desired OCP outputs, which should be optimized in the upper level, but also their (post-optimal) sensitivities (Section 3.2). This makes it possible to use the (post-optimal) sensitivities in the upper level for the NEWTON-type update and solution of the OCP. Additionally, the upper level can also feature a differential evolution algorithm (DEA), which does not require post-optimal sensitivities, for models that either do not provide these or to find the region of the global optimum in order to apply the NEWTON-type optimization (Section 3.1).

Continuing with the framework development, Chapter 4 introduces the developed DOC framework including gPC (which is a development of the bi-level framework in Chapter 3): The basic idea of the framework is to split up the deterministic sampling of the gPC expansion coefficients using SC into DOCPs. Afterward, these DOCPs are combined in a connection problem that allows to solve a ROCP (Section 4.1). To achieve this, one part of the chapter is concerned with deriving a distributed representation of the statistical moments that are normally of interest in ROC (Section 4.2). Here, a representation is developed that does not require any sensitivities to update the connection problem. Then, the general DOCP statement is given in Section 4.4. Additionally, an example distribution for a cost function depending on mean and variance is given. The chapter concludes with the derivation of the Karush-Kuhn-Tucker (KKT) conditions for the DOC framework that show the mathematical viability of the developed methods.

Then, Chapter 5 introduces the developed and implemented gPC discretization framework using trapezoidal collocation within FALCON.m. Here, a method is introduced that use the expansion coefficients as optimization variables (Section 5.1). Additionally, the OC collocation defect (CD) is calculated by applying a differential equation for the expansion coefficients. Thus, the approach transforms the OCP from the physical into the gPC domain. This makes it possible to directly optimize or constrain statistical moments in the OCP and additional calculate CCs. Finally, an intuition on the viability of the approach is given in Section 5.2 by showing that it is merely a change in the considered state variables, i.e., a nonlinear transformation using the gPC expansion.

Based on the gPC collocation methods in Chapter 5, Chapter 6 introduces the CC-OC framework developed in this study: This framework uses the possibility provided by the gPC collocation framework, as well as the fast sampling that is then possible using the gPC expansion, to get a good estimate on fulfilling a CC in each iteration of the NLP

solution. Therefore, an initial framework is introduced in Section 6.1 that is based on standard MCA sampling of the CC inside e.g., a path function of the OCP. A major point of this initial framework is the development of a differentiable approximation of the CC for NEWTON-type optimization, while also calculating the desired CC with a high accuracy. For this purpose, an approximation based on sigmoid functions is developed and combined with a homotopy strategy to iteratively sharpen the sigmoid function such that the desired CC is approximated accurately. As a development, the incorporation of SubSim within the framework is explained in Section 6.2 to be able to solve CC-OCPs with rare-event failure probabilities. Within this framework it is possible to calculate failure probabilities that cannot be covered by standard MCA because they are too rare.

Then, Chapter 7 gives an overview on the results of the first example that was used for testing the developed ROC frameworks: Here, the bi-level OC framework of Chapter 3 is used to calculate the optimal adaptation gain of an adaptive controller. For this, a short period approximation of a F16 is used (Section 7.1), where parameters in the state and control matrix are assumed to be uncertain. Using this model, Section 7.2 introduces the problem formulation for the design for the adaptations gains in an adaptive control loop in a robust manner. The robust design results are then introduced in Section 7.3, which especially also introduces the results containing a CC in Subsection 7.3.3.

An example for the DOC framework, introduced in Chapter 4, is given in Chapter 8. Here, an aerobatic aircraft within an race track under wind uncertainties is optimized (Section 8.1). Here, to determine the required gPC expansion order, for an accurate description of the statistical moments of the dynamic model, the dynamic equations are analyzed and the model is optimized and compared to a MCA in Section 8.2. Afterward, Section 8.3 checks and verifies the validity of the DOC framework in application by comparing a distributed mean value cost function with a standard generalized polynomial chaos-stochastic collocation framework (gPC-SC) expansion. Finally, a robustifying cost, which yields a PARETO problem, is added to the DOC formulation and the possible robustness improvements are shown.

As a final application case, Chapter 9 introduces results of the CC-OC framework, here, based on the methods developed in Chapter 6. Here, a quadcopter is optimized in an environment with obstacles (Section 9.1). The dynamic quadcopter model has both uncertainties in internal parameters (rotor efficiency) and external parameters (obstacle size). Once more, the required gPC expansion order is determined by analyzing the dynamic model equations as well as comparison to MCA (Section 9.2). Afterward, a frequent event CC by the MCA-based CC-OC formulation of Section 6.1 is introduced in Section 9.3. Here, the CC should ensure an avoidance of the obstacles with a high probability. A comparison between the optimality reduction and the robustness gain is made as well. Finally, Section 9.4 extends the CC for the obstacle avoidance to also cover rare-event CCs (applying the method in Section 6.2). An important aspect of the application is

the use of the Beta PDF estimation of the failure probability, available from SubSim, in the CC-OC framework to improve the confidence in the robust, optimal trajectory. Once more, robustness improvements as well as optimality reduction are compared.

Concluding the thesis is Chapter 10, which gives some remarks on the developed methods and an outlook on future developments of the methods as well as generally in the field of ROC.

Chapter 2

Open-Loop Optimal Control and Uncertainty Quantification

This chapter introduces the theoretical background for the basic parts of the developed robust open-loop direct optimal control (ROC) frameworks in this thesis. Therefore, the chapter is organized as follows: Section 2.1 introduces the general open-loop direct optimal control problem (OCP) formulation and how this OCP is solved using NEWTON-type, nonlinear program (NLP) solvers. Then, Section 2.2 introduces classic, sampling-based uncertainty quantification methods, while Section 2.3 presents the method of generalized polynomial chaos (gPC), which is used in this work as the main method to efficiently model uncertainties in the ROC frameworks. Section 2.4 concludes the theoretical part with more sophisticated open-loop direct optimal control (OC) methodologies, such as chance-constrained open-loop direct optimal control (CC-OC) and distributed open-loop direct optimal control (DOC), which are applied in the ROC formulations of this thesis.

2.1 Continuous Open-Loop Optimal Control

In general, the continuous OCP (which in this thesis refers to the open-loop) defined in (2.1) is solved in the case of trajectory optimization [16, 51, 74]: Here, the overall goal is to find the optimal control history $\mathbf{u}_{\text{opt}}(t) \equiv \mathbf{u}_{\text{opt}} \in \mathbb{R}^{n_u}$ that corresponds to the optimal state history $\mathbf{x}_{\text{opt}}(t) \equiv \mathbf{x}_{\text{opt}} \in \mathbb{R}^{n_x}$, as well as relevant time-invariant optimal parameters $\mathbf{p}_{\text{opt}} \in \mathbb{R}^{n_p}$ (these parameters might e.g., be a model dimension that should be optimized) minimizing a cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_q} \to \mathbb{R}$ as follows (take into account that the time dependence $\cdot(t)$ of e.g., states and controls, is omitted throughout this thesis for the sake of brevity):

$$
\begin{aligned}
\min_{\mathbf{u},\mathbf{p}} \quad & J\left(\mathbf{x},\mathbf{u},\mathbf{p};\mathbf{q}\right) \\
\text{s.t.} \quad & \mathbf{x}_{\text{lb}} \leq \mathbf{x} \leq \mathbf{x}_{\text{ub}}, \\
& \mathbf{u}_{\text{lb}} \leq \mathbf{u} \leq \mathbf{u}_{\text{ub}}, \\
& \mathbf{p}_{\text{lb}} \leq \mathbf{p} \leq \mathbf{p}_{\text{ub}}, \\
& \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{p};\boldsymbol{\theta},\mathbf{q}) = \dot{\mathbf{x}}, \\
& \mathbf{c}(\mathbf{x},\mathbf{u},\mathbf{p};\mathbf{q}) \leq \mathbf{0}, \\
& \boldsymbol{\psi}(\mathbf{x},\mathbf{u},\mathbf{p};\mathbf{q}) = \mathbf{0}
\end{aligned}
\tag{2.1}
$$

Here, $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_q} \to \mathbb{R}^{n_x}$ provides an equation of motion (EoM) for each state, i.e., the state dynamics, while $\mathbf{c} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_q} \to \mathbb{R}^{n_c}$ depicts the inequality path and point constraints, and $\boldsymbol{\psi} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_q} \to \mathbb{R}^{n_\psi}$ are the equality path and point constraints. It should be noted that all inequality path/point constraints that are fulfilled, i.e., for whom it is $\mathbf{c} < 0$, are said to be *inactive*, while inequality constraints with $\mathbf{c} = 0$ are called *active*. As tracking active inequality constraints is important e.g., during the solution process of the OCP and for sensitivity analysis, the active constraints are collected in the set $\mathbb{A}$. Invariant parameters, i.e., parameters within the OCP that are not optimized, are denoted as $\mathbf{q} \in \mathbb{R}^{n_q}$ in (2.1). These invariant parameters are later on e.g., used in the sensitivity analysis (Subsection 2.4.2) to assess their influence on the optimal solution and are therefore also denoted as "sensitive" parameters. It should be noted that non-optimizable parameters are generally separated from optimizable parameters using a semicolon. Further take into account that the optimization parameters ("decision variables"), i.e., $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{p}$, are generally limited by box constraints, i.e., they have a lower "lb" and an upper bound "ub" (equality constraints are consequently described by $\cdot_{\text{lb}} = \cdot_{\text{ub}}$).

An important aspect of the OCP formulation in (2.1) is the incorporation of random/uncertain parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ in the EoMs (see Appendix B for the definition of a random variable (RV) and general probability theory). These random parameters are not optimizable, like e.g., the relevant parameters $\mathbf{p}$, but are uncertain, i.e., defined by a probability density function (PDF) ($\rho_\Theta(\theta)$; thus, (2.1) solves the OCP at a single realization of this random parameter). It should be noted that by the definition of the OCP in (2.1), the structural influence of the uncertainties in the OCP, and specifically the state dynamics, is required to be known. This is generally not a hard constraint to fulfill as the model structure must anyways be known in standard OC applications.

The efficient incorporation of the uncertainties acting on the state dynamics in (2.1) within the robust open-loop direct optimal control problem (ROCP) is a major scope of this thesis and dealt with in the next chapters. Take into account that the constraints in the OCP (except for the chance constraints (CCs) introduced later on) as well as the cost function are not functionally depending on the uncertain parameters but only indirectly, i.e., through the state and control history, in this thesis. This formulation has proven to

be sufficient as it is normally desired to have deterministic constraints (especially bounds), while the cost function is generally specified by the user in a deterministic way (i.e., without uncertain parameters). Still, the principles of this thesis can be extended directly to adapt the problem formulation in (2.1) as desired.

It is important to see that (2.1) does not incorporate expectation operators or similar that one might expect in OC with uncertainties. This is due to the fact that the gPC methodology, introduced in Section 2.3, relies on a deterministic sampling. Therefore, all OCPs are also solved deterministically although they are uncertain in nature. This relation to gPC clears the choice for the OCP formulation given in (2.1) without expectation operators, while required adaptations to (2.1), e.g., for CC-OC, are introduced when necessary (Chapter 6). Thus, as also stated before, (2.1) can be viewed as a single realization of the uncertain OCP.

In (2.1), the scalar valued cost functional J is expressed by a BOLZA COST FUNCTIONAL as follows [16, p. 131]:

$$J\left(\mathbf{x},\mathbf{u},\mathbf{p};\mathbf{q}\right)=e\left(\mathbf{x}\left(t_f\right),\mathbf{u}\left(t_f\right),\mathbf{p};\mathbf{q}\right)+\int_{t=t_0}^{t_f}L\left(\mathbf{x},\mathbf{u},\mathbf{p};\mathbf{q}\right)\,\mathrm{d}t \tag{2.2}$$

Here, the first addend, e, is the so-called MAYER TERM, i.e., a term that is related to the end values of the OCP. The second addend, i.e., the integral of L, corresponds to the LAGRANGE TERM that adds a cost related to the OCP time interval ("running cost"; the initial time t_0 and final time t_f can be part of the optimization parameters $\mathbf{p}$). It should be noted here that generally a minimum of this cost function is calculated in (2.1). A maximization can consequently be achieved by minimizing the negative cost function. Further take into account that other cost function influences, e.g., further addends, can also be added to (2.2) by summation if desired.

The constrained OCP in (2.1) is then solved using a LAGRANGE FUNCTION that connects cost and constraints as follows [16, p. 13ff.]:

$$\mathcal{L}\left(\mathbf{z},\boldsymbol{\lambda},\boldsymbol{\mu};\boldsymbol{\theta},\mathbf{q}\right)=J\left(\mathbf{z};\mathbf{q}\right)+\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{c}\left(\mathbf{z};\mathbf{q}\right)+\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\psi}\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right) \tag{2.3}$$

Here $\boldsymbol{\lambda}\in\mathbb{R}^{n_\lambda}$ and $\boldsymbol{\mu}\in\mathbb{R}^{n_\mu}$ are the LAGRANGE multipliers for the inequality and the equality constraints, respectively. Take into account that the notation in (2.3) is further simplified by introducing the optimization parameter vector (also: "decision variable vector") $\mathbf{z}=\begin{bmatrix}\mathbf{p}^{\mathsf{T}} & \mathbf{x}^{\mathsf{T}} & \mathbf{u}^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}\in\mathbb{R}^{n_z}$ with $n_z=n_p+n_x+n_u$ that represent the time invariant optimization parameters (e.g., the final time), states, and controls. This decision variable vector holds the discretization of the optimal trajectory for the OCP, which is detailed in Subsection 2.1.2. Additionally, the EoMs are essentially represented by an equality path constraint and are therefore, included in $\boldsymbol{\psi}$. Finally, the box constraints on the optimization parameters are generally represented by inequality or equality constraints, based on their definition, and can therefore be treated as part of these for the simplicity of writing as well.

Then, an optimal solution of the LAGRANGE FUNCTION in (2.3) must fulfill the first order necessity conditions. These are known as the Karush-Kuhn-Tucker (KKT) optimality conditions and defined as follows [16, p. 196ff.], [51, p. 87ff.], or [110, p. 19f.]:

Definition 2.1 (Karush-Kuhn-Tucker Optimality Conditions)**.** *Let* $\mathbf{z}_{\text{opt}}$ *be a local optimum of the standard constrained optimization problem in* (2.1) *with the* LAGRANGE FUNCTION *defined as in* (2.3) *and the functions* J, $\mathbf{c}$, *and* $\boldsymbol{\psi}$ *be continuously differentiable. Then, there exist optimal* LAGRANGE *multipliers* $\boldsymbol{\lambda}_{\text{opt}} \in \mathbb{R}^{n_\lambda}$ *and* $\boldsymbol{\mu}_{\text{opt}} \in \mathbb{R}^{n_\mu}$ *that fulfill the following four KKT conditions:*

K. 1 Sign condition fulfilled for all inequality constraint LAGRANGE *multipliers (part of so called "dual feasibility"), i.e.,*

$$\boldsymbol{\lambda}_{\text{opt}} \geq \mathbf{0}$$

K. 2 Optimality condition fulfilled (part of so called "dual feasibility"), i.e.,

$$\nabla_{\mathbf{z}} \mathcal{L}\left(\mathbf{z}_{\text{opt}}, \boldsymbol{\lambda}_{\text{opt}}, \boldsymbol{\mu}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right) = \mathbf{0}$$

K. 3 Feasibility fulfilled, i.e., all constraints are satisfied (so called "primal feasibility"):

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}} \mathcal{L}\left(\mathbf{z}_{\text{opt}}, \boldsymbol{\lambda}_{\text{opt}}, \boldsymbol{\mu}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right) &= \mathbf{c}\left(\mathbf{z}_{\text{opt}}; \mathbf{q}\right) \leq \mathbf{0} \\ \nabla_{\boldsymbol{\mu}} \mathcal{L}\left(\mathbf{z}_{\text{opt}}, \boldsymbol{\lambda}_{\text{opt}}, \boldsymbol{\mu}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right) &= \boldsymbol{\psi}\left(\mathbf{z}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right) = \mathbf{0} \end{aligned}$$

K. 4 Complementary condition (/slackness) fulfilled, i.e.,

$$\boldsymbol{\lambda}_{\text{opt}}^{\mathsf{T}} \mathbf{c}\left(\mathbf{z}_{\text{opt}}; \mathbf{q}\right) = \mathbf{0}$$

For the KKT conditions in Definition 2.1 to hold, certain regularity conditions (also called: "constraint qualifications") should be valid to ensure a feasible descent direction and unique LAGRANGE multipliers [16, p. 28f. & 40ff.], [51, p. 95ff.]. These generally state that constraints must be linearly independent (e.g., linear independence of the gradients of the active inequality constraints). If the regularity conditions hold, the KKT conditions are a necessary optimality criterion. Thus, the KKT conditions give candidates for optimal points. Sufficient conditions can e.g., be found in [17, p. 47f.] or [51, p. 100 & p. 295ff.] and mainly rely on the determination of the definiteness of the Hessian of the LAGRANGE FUNCTION (only using the active inequality constraints). It should be noted that the NLP optimization algorithms, used in this thesis to solve OCPs, are essentially KKT solvers, i.e., they calculate candidates for optimal points that satisfy the KKT conditions. Further take

into account that Definition 2.1, in connection with the application to NLP optimization algorithms and the Hessian of the LAGRANGE FUNCTION, requires that the OCP in (2.1) is sufficiently smooth (i.e., continuous and differentiable) [16, p. 42ff.].

2.1.1 Indirect Solution Methods

This section gives a brief overview of the so-called indirect solution method for the OCP in (2.1). This technique can be mainly described by the *optimize, then discretize* principle and is based on the CALCULUS OF VARIATION and PONTRYAGIN'S MINIMUM PRINCIPLE [17, p. 58 ff.] [51, ch. 3] [74, ch. 5]. Due to the fact that this method is normally not applicable to large-scale OCPs, especially with multiple inequality constraints as present with aircraft trajectory optimization, it is not further considered for the ROC solution of the examples in this work. But, it can e.g., be used to derive the linear–quadratic regulator (LQR) and linear–quadratic–Gaussian regulator (LQG) solutions mentioned before. Thus, the principle is important to understand in order to get the general goal of OC and how this goal is achieved by the direct solution method presented in Subsection 2.1.2.

To show the general procedure of the indirect method, an unconstrained OCP formulation, i.e., the formulation for which there are no equality constraints (other than the EoMs) or inequality constraints in (2.1) and only a LAGRANGE cost term, is looked at. Additionally, the initial time is fixed and no optimizable parameters are considered. Then, the HAMILTONIAN is defined as follows [17, p. 59]:

$$\mathcal{H}\left(\mathbf{x}, \mathbf{u}, \boldsymbol{\mu}; \boldsymbol{\theta}, \mathbf{q}\right) = L\left(\mathbf{x}, \mathbf{u}; \mathbf{q}\right) + \boldsymbol{\mu}^{\mathsf{T}} \mathbf{f}\left(\mathbf{x}, \mathbf{u}; \boldsymbol{\theta}, \mathbf{q}\right) \tag{2.4}$$

Now, this HAMILTONIAN must fulfill the necessary conditions from the PONTRYAGIN MINIMUM PRINCIPLE given as follows [74, p. 233]:

P. 1 Minimum condition: $\mathcal{H}\left(\mathbf{x}_{\text{opt}}, \mathbf{u}_{\text{opt}}, \boldsymbol{\mu}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right) = \min\limits_{\mathbf{u} \in U \subset \mathbb{R}^{n_u}} \mathcal{H}\left(\mathbf{x}_{\text{opt}}, \mathbf{u}, \boldsymbol{\mu}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right)$

P. 2 Adjoint differential equation: $\dot{\boldsymbol{\mu}} = -\nabla_{\mathbf{x}} \mathcal{H}\left(\mathbf{x}_{\text{opt}}, \mathbf{u}_{\text{opt}}, \boldsymbol{\mu}; \boldsymbol{\theta}, \mathbf{q}\right)$

P. 3 Transversality condition (no terminal cost/constraint): $\boldsymbol{\mu}|_{t_0, t_f} = \mathbf{0}$

P. 4 For free final time: $\mathcal{H}\left(\mathbf{x}, \mathbf{u}, \boldsymbol{\mu}; \boldsymbol{\theta}, \mathbf{q}\right)|_{t_f} = 0$

P. 5 For time-autonomous systems: $\frac{\mathrm{d}}{\mathrm{d}t} \mathcal{H}\left(\mathbf{x}_{\text{opt}}, \mathbf{u}_{\text{opt}}, \boldsymbol{\mu}; \boldsymbol{\theta}, \mathbf{q}\right) = 0$

Thus, the indirect method at first tries to formulate the minimum conditions (see Pontryagin P. 1–Pontryagin P. 5) and afterward, solves the two-point boundary value problem, given by the state dynamics and the adjoint equation (Pontryagin P. 2). The solution of the two-point boundary value problem normally requires the discretization of the problem. Additionally, depending on the chosen simulation method, a good initial guess for the backward simulation of the adjoint variables might be required for this (Pontryagin P. 2 and Pontryagin P. 3).

It is interesting to note that the co-states, i.e., the solution of the adjoint differential equation, can be interpreted as the sensitivity of the solution. Thus, they show how the solution, i.e., the cost, could be improved, when the constraints are changed [17, p. 62]. The concept of sensitivities is further detailed in Subsection 2.4.2.

2.1.2 Direct Solution Methods

The direct solution methods for OCPs are the basis of this work because the nonlinear model as well as constraints can generally not be treated by indirect OC methods [16, p. 91]. These methods are best described by the *discretize, then optimize* principle. In fact, with direct OC one solves a large-scale NLP [16] (it should be noted that the terms NLP and OCP can be used synonymously as only discretized OCPs, which are essentially NLPs, are solved in this thesis). This NLP then computes an approximation of the real optimal solution that would be obtained by the indirect method (Subsection 2.1.1).

A method to discretize the OCP in (2.1) to get an NLP is given in Subsection 2.1.2.1 with the collocation technique. Further methods, like shooting [16, p. 93ff.] or pseudo-spectral transcription [15], can also be used, but are not introduced within this thesis as it focuses on the collocation method.

The collocation technique is the discretization method used in the software FSD optimal control tool for MATLAB® (FALCON.m) developed at the Institute of Flight System Dynamics (FSD) and is extended by the gPC methodology in this work. Therefore, Subsection 2.1.2.2 also introduces the FALCON.m framework to get an overview on the current capabilities of the software.

To solve discretized OCPs, i.e., NLPs, there mainly exist two methodologies: The first is the interior point (IP) method introduced in Subsection 2.1.3 and the second is the sequential quadratic programming (SQP) method introduced in Subsection 2.1.4. The main difference of these methods is the treatment of the inequality constraints within the NLP, which is described in the subsections.

2.1.2.1 Collocation Technique and Nonlinear Programming

Collocation techniques are the chosen method for the OC discretization of this work. They rely on a full discretization of both the state and control history, which transforms the continuous OCP in (2.1) into a discretized nonlinear program (NLP). For the following derivations, the discretization of states and controls is on a common grid for simplicity. Extensions to non-common grids are e.g., discussed in [110, p. 75 & p. 133ff.] and also implemented in FALCON.m. Generally, the OC time interval for collocation is mapped to a non-dimensional/normalized time grid with n_τ time steps that is discretized as follows:

$$\tau_i \in [0,1]\,, \quad i = 1, \ldots, n_\tau, \quad \tau_i < \tau_{i+1} \tag{2.5}$$

Consequently, the optimization parameter vector ("decision variable vector"), i.e., the vector the optimizer is adapting to find an optimal point, is discretized at these time steps as well. It is defined as follows:

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{p}^\mathsf{T} & \mathbf{x}_1^\mathsf{T} & \mathbf{u}_1^\mathsf{T} & \mathbf{x}_2^\mathsf{T} & \mathbf{u}_2^\mathsf{T} & \dots & \mathbf{x}_{n_\tau-1}^\mathsf{T} & \mathbf{u}_{n_\tau-1}^\mathsf{T} & \mathbf{x}_{n_\tau}^\mathsf{T} & \mathbf{u}_{n_\tau}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{n_{\tilde{z}}} \tag{2.6}$$

The indices denote the n_τ time steps and the size is $n_{\tilde{z}} = n_p + (n_x + n_u) \cdot n_\tau$. Take into account that by construction of the optimization parameter vector in (2.6), the constraints for optimizable parameters, states, and controls are directly enforced within this vector in the NLP. It should be further noted that the optimizable parameters are time-invariant and therefore, do not need to be discretized on the non-dimensional time grid.

The transformation between the non-dimensional and physical time is then given as follows:

$$\tau = \frac{t - t_0}{t_f - t_0} = \frac{t}{t_f}, \qquad t_0 = 0 \tag{2.7}$$

The time transformation in (2.7) is used to ensure that the integration is always performed with the same step size. Additionally, the initial and final time are then the sole optimization parameters required to specify the time history. Take into account that the initial time is set to zero without loss of generality and just for the sake of brevity. The discretization step itself (e.g., evenly or unevenly spaced) is selected by the user.

Applying (2.7), the state dynamics must be scaled as well to fulfill the integration on the non-dimensional time grid as follows:

$$\mathbf{f}\left(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p}; \boldsymbol{\theta}, \mathbf{q}\right) = \dot{\mathbf{x}}_i = \frac{1}{t_f}\mathbf{x}_i' \tag{2.8}$$

Thus, the state dynamics are essentially scaled with the final time to account for a solution of the discretized NLP on a normalized time grid.

Then, an integration scheme for the transcription is defined, which is called the collocation scheme. This scheme propagates the EoMs over the discretized time grid. The resulting NLP parameter optimization problem can then be passed to a numerical optimizer. A general first-order/single-step collocation scheme is defined as follows:

$$\mathbf{CD}_{i,i+1} = \mathbf{x}_{i+1} - \mathbf{x}_i - \mathbf{g}\left(\mathbf{x}_{i+1}, \mathbf{u}_{i+1}, \mathbf{x}_i, \mathbf{u}_i, \mathbf{p}, h_\tau\right) \stackrel{!}{=} 0 \in \mathbb{R}^{n_x} \tag{2.9}$$

The discretization step size vector on the non-dimensional time grid is depicted by h_τ and $\mathbf{g}\left(\cdot\right)$ is the function that defines the collocation scheme. It should be noted that (2.9) is given in the form of a defect equation, the collocation defect (CD). The CD symbolizes the error of the integration algorithm between two integration time steps. This error must consequently be zero and is enforced as an equality constraint between all discretized time steps in the NLP.

Within this work, the trapezoidal rule is applied as the collocation integration method. Thus, (2.9) becomes (applying (2.8)):

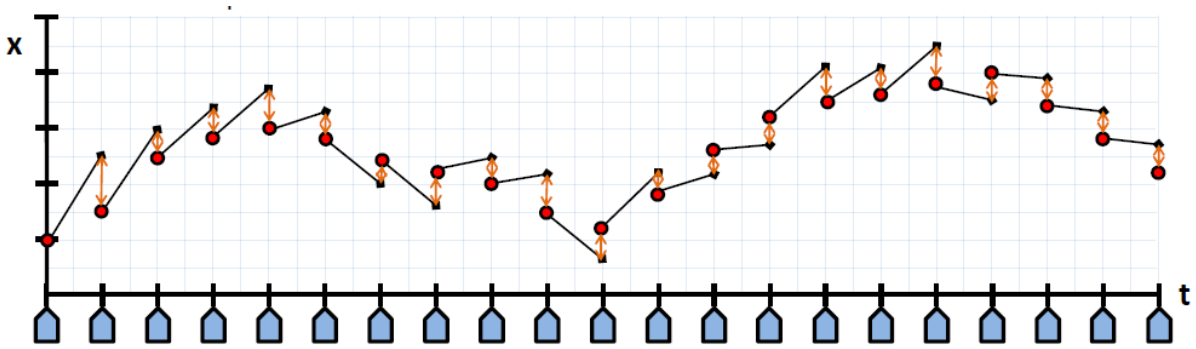

Figure 2.1: *Visualization of general idea for collocation techniques with full state and time discretization (red dots), one step integration (black line), and collocation defects (orange arrows) (after [64]), [110, p. 38].*

$$\mathbf{CD}_{i,i+1} = \mathbf{x}_{i+1} - \mathbf{x}_i - \underbrace{h_\tau}_{=\tau_{i+1}-\tau_i} \overbrace{\frac{t_f}{2}\left(\dot{\mathbf{x}}_{i+1} + \dot{\mathbf{x}}_i\right)}^{\mathbf{g}(\mathbf{x}_{i+1},\mathbf{u}_{i+1},\mathbf{x}_i,\mathbf{u}_i,\mathbf{p},h_\tau)=\frac{\mathbf{x}'_{i+1}+\mathbf{x}'_i}{2}} \stackrel{!}{=} 0 \tag{2.10}$$

Take into account that the trapezoidal rule is an implicit integration scheme part of Lobatto integrators [57]. It therefore exhibits A-stability [31, 57], i.e., is able to integrate stiff differential equations without theoretical limitations on the discretization step size. Thus, it is suited as the collocation scheme for an OCP [16, p. 100].

The introduced discretization procedure is also depicted in Figure 2.1. Here, the x-axis shows the time history t, while the y-axis illustrates the state values x at different times. The red dots are the state values chosen by the optimizer during the NLP evaluation. The black squares symbolize the integrated values using e.g., the trapezoidal rule. The orange arrows depict the value of the CD at the time steps. This CD must be zero for the NLP algorithm to converge to an optimum (KKT Condition 3).

Using the CD in (2.10), an objective/a residual vector for the NLP can be defined. This vector must fulfill the optimality criteria (Definition 2.1), i.e., the minimization of the cost as well as the fulfillment of path and point constraints and is given as follows:

$$\tilde{\mathbf{F}} = \begin{bmatrix} J & \mathbf{y}_1^\mathsf{T} & \mathbf{CD}_{1,2}^\mathsf{T} & \mathbf{y}_2^\mathsf{T} & \mathbf{CD}_{2,3}^\mathsf{T} & \dots & \mathbf{y}_{n_\tau-1}^\mathsf{T} & \mathbf{CD}_{n_\tau-1,n_\tau}^\mathsf{T} & \mathbf{y}_{n_\tau}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{n_{\tilde{F}}} \tag{2.11}$$

The size of the residual vector is given by $n_{\tilde{F}} = 1 + n_\tau \cdot n_y + (n_\tau - 1) \cdot n_x$.

In (2.11), $\mathbf{y}_i \in \mathbb{R}^{n_y}$ depicts path or point constraints, e.g., for outputs like the load factor, that are added at the discretized time steps (not necessarily at each) by the user to constrain the optimal trajectory. These outputs are defined by the following nonlinear function of states, controls, and parameters:

$$\mathbf{y}_i = \mathbf{h}\left(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p}; \mathbf{q}\right) \tag{2.12}$$

Thus, any desired quantity of the dynamic model can be constrained.

Overall, the discretization by (2.6), (2.10), and (2.11) yields the following basic NLP transformation for the original continuous OCP definition in (2.1):

$$\begin{aligned}
&\min_{\tilde{\mathbf{z}}} \quad J\left(\tilde{\mathbf{z}};\mathbf{q}\right)\\
&\text{s.t.} \qquad\qquad\qquad\qquad \tilde{\mathbf{z}}_{\text{lb}} \leq \tilde{\mathbf{z}} \leq \tilde{\mathbf{z}}_{\text{ub}},\\
&\qquad\qquad\qquad\qquad \mathbf{c}(\tilde{\mathbf{z}};\mathbf{q}) = \begin{bmatrix} \mathbf{y}_{1,\text{lb}} - \mathbf{y}_1 \\ \mathbf{y}_1 - \mathbf{y}_{1,\text{ub}} \\ \vdots \\ \mathbf{y}_{n_\tau,\text{lb}} - \mathbf{y}_{n_\tau} \\ \mathbf{y}_{n_\tau} - \mathbf{y}_{n_\tau,\text{ub}} \end{bmatrix} \leq \mathbf{0},\\
&\boldsymbol{\psi}(\tilde{\mathbf{z}};\boldsymbol{\theta},\mathbf{q}) = \begin{bmatrix} \mathbf{x}_2 - \mathbf{x}_1 - \frac{t_f}{2} h_\tau \left(\dot{\mathbf{x}}_2 + \dot{\mathbf{x}}_1\right) \\ \vdots \\ \mathbf{x}_{n_\tau} - \mathbf{x}_{n_\tau-1} - \frac{t_f}{2} h_\tau \left(\dot{\mathbf{x}}_{n_\tau} + \dot{\mathbf{x}}_{n_\tau-1}\right) \end{bmatrix} = \mathbf{0}
\end{aligned} \tag{2.13}$$

Here, $\tilde{\mathbf{z}}_{\text{lb}}$ is the lower bound vector of the optimization parameters, while $\tilde{\mathbf{z}}_{\text{ub}}$ is its upper bound. Further take into account that box constraints, normally used for the inequality constraints, are transformed to the standardized description, i.e., $\mathbf{c} \leq \mathbf{0}$, by introducing two constraints. Here, the first constraint is related to the lower bound $\mathbf{y}_{i,\text{lb}}$ and the second constraint is related to the upper bound $\mathbf{y}_{i,\text{ub}}$. Further note that the EoMs are directly incorporated in the equality constraints by the trapezoidal integration scheme. It should be further considered that (2.13) forms the basic description of the discretized NLP, which can be extended using additional equality and inequality constraints as desired. But, for the sake of consistency, the following derivations are only made using the form in (2.13), which is also most common.

It is important to note that the cost function in (2.2) is discretized as well using the trapezoidal integration scheme (see (2.10)) as follows:

$$J\left(\tilde{\mathbf{z}};\mathbf{q}\right) = e\left(\mathbf{x}_{n_\tau}, \mathbf{u}_{n_\tau}, \mathbf{p};\mathbf{q}\right) + h_\tau \frac{t_f}{2} \sum_{i=1}^{n_\tau-1} \left[L\left(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p};\mathbf{q}\right) + L\left(\mathbf{x}_{i+1}, \mathbf{u}_{i+1}, \mathbf{p};\mathbf{q}\right)\right] \tag{2.14}$$

Then, in order to understand the principle of the NLP solution, an NLP with box constraints on the optimization parameters and otherwise only equality constraints is considered at. Thus, (2.13) becomes:

$$\begin{aligned}
&\min_{\tilde{\mathbf{z}}} \quad J\left(\tilde{\mathbf{z}};\mathbf{q}\right)\\
&\text{s.t.} \quad \boldsymbol{\psi}(\tilde{\mathbf{z}};\boldsymbol{\theta},\mathbf{q}) = \begin{bmatrix} \mathbf{x}_2 - \mathbf{x}_1 - \frac{t_f}{2} h_\tau \left(\dot{\mathbf{x}}_2 + \dot{\mathbf{x}}_1\right) \\ \vdots \\ \mathbf{x}_{n_\tau} - \mathbf{x}_{n_\tau-1} - \frac{t_f}{2} h_\tau \left(\dot{\mathbf{x}}_{n_\tau} + \dot{\mathbf{x}}_{n_\tau-1}\right) \end{bmatrix} = \mathbf{0}
\end{aligned} \tag{2.15}$$

Problem (2.15) can be solved using the following LAGRANGE FUNCTION:

$$\mathcal{L}\left(\tilde{\mathbf{z}}, \boldsymbol{\mu}; \boldsymbol{\theta}, \mathbf{q}\right) = J\left(\tilde{\mathbf{z}}; \mathbf{q}\right) + \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{\psi}\left(\tilde{\mathbf{z}}; \boldsymbol{\theta}, \mathbf{q}\right) \tag{2.16}$$

Applying the KKT conditions in Definition 2.1 on the LAGRANGE FUNCTION in (2.16) yields the following optimality conditions that must be fulfilled by the decision variables and LAGRANGE multipliers:

$$\begin{aligned} \nabla_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_{\text{opt}}, \boldsymbol{\mu}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right) &\stackrel{!}{=} \mathbf{0} \\ \boldsymbol{\psi}\left(\tilde{\mathbf{z}}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right) &\stackrel{!}{=} \mathbf{0} \end{aligned} \tag{2.17}$$

Then, the NEWTON method [5, p. 294] can be applied on the KKT conditions in (2.17) to iteratively calculate an update on the primal (optimization) $\Delta\tilde{\mathbf{z}}$ and the dual (multiplier) variables $\Delta\boldsymbol{\mu}$ at the current iterate k as follows [16, p. 12ff.]:

$$\begin{bmatrix} \nabla^2_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_k, \boldsymbol{\mu}_k; \boldsymbol{\theta}, \mathbf{q}\right) & \nabla^2_{\tilde{\mathbf{z}}, \boldsymbol{\mu}} \mathcal{L}\left(\tilde{\mathbf{z}}_k, \boldsymbol{\mu}_k; \boldsymbol{\theta}, \mathbf{q}\right)^{\mathsf{T}} \\ \nabla_{\tilde{\mathbf{z}}} \boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k; \boldsymbol{\theta}, \mathbf{q}\right) & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \Delta\tilde{\mathbf{z}} \\ \Delta\boldsymbol{\mu} \end{bmatrix} = - \begin{bmatrix} \nabla_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_k, \boldsymbol{\mu}_k; \boldsymbol{\theta}, \mathbf{q}\right) \\ \boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k; \boldsymbol{\theta}, \mathbf{q}\right) \end{bmatrix} \tag{2.18}$$

The symmetric Hessian of the LAGRANGE function is denoted by $\nabla^2_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_k, \boldsymbol{\mu}_k\right)$. The updates can then be used to calculate the new primal and dual variables:

$$\begin{aligned} \tilde{\mathbf{z}}_{k+1} &= \tilde{\mathbf{z}}_k + \Delta\tilde{\mathbf{z}} \\ \boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \Delta\boldsymbol{\mu} \end{aligned} \tag{2.19}$$

It should be reminded here that k is an iteration index in the NLP procedure and not a discretized time index. Take into account that NLP solvers, such as Interior Point Optimizer (IPOPT), additionally implement more sophisticated update strategies using line-search methods [133, p. 30ff.]. This is not considered here for the sake of simplicity.

The procedure in (2.18) and (2.19) is iterated until the KKT conditions in (2.17) are satisfied (Definition 2.1). Then, (2.19) converged to a candidate for an optimal solution with corresponding multipliers. It should be noted that the procedure in (2.18) and (2.19) requires an initial guess for the primal and dual variables. Further take into account that in a numerical optimization procedure, i.e., when applied to the discretized NLP, the KKT conditions cannot be fulfilled exactly but only approximately. Here, a stopping criterion, which the optimal point must fulfill, of the following kind is normally used:

$$\begin{aligned} ||\nabla_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_{\text{opt}}, \boldsymbol{\mu}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right)||_\infty &\leq \epsilon_{\text{opt}} \\ ||\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q}\right)||_\infty &\leq \epsilon_{\text{feas}} \end{aligned} \tag{2.20}$$

Thus, primal and dual feasibility of the KKT conditions (Definition 2.1), specifically their infinity norm $||\cdot||_\infty$, are fulfilled up to a numeric tolerance of ϵ_{opt} and ϵ_{feas} respectively.

It should be noted that (2.18) can also be rewritten using the identities [16, p. 13]:

$$\begin{aligned}\nabla_{\tilde{\mathbf{z}}}\mathcal{L}\left(\tilde{\mathbf{z}}_k,\boldsymbol{\mu}_k;\boldsymbol{\theta},\mathbf{q}\right) &= \nabla_{\tilde{\mathbf{z}}}J\left(\tilde{\mathbf{z}}_k;\mathbf{q}\right)+\nabla_{\tilde{\mathbf{z}}}\underbrace{\boldsymbol{\mu}_k^{\mathsf{T}}\boldsymbol{\psi}_k\left(\tilde{\mathbf{z}}_k;\boldsymbol{\theta},\mathbf{q}\right)}_{\boldsymbol{\psi}_k(\tilde{\mathbf{z}}_k;\boldsymbol{\theta},\mathbf{q})^{\mathsf{T}}\boldsymbol{\mu}_k}\\ \nabla^2_{\tilde{\mathbf{z}},\boldsymbol{\mu}}\mathcal{L}\left(\mathbf{z}_k,\boldsymbol{\mu}_k;\boldsymbol{\theta},\mathbf{q}\right) &= \nabla_{\tilde{\mathbf{z}}}\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k;\boldsymbol{\theta},\mathbf{q}\right)\end{aligned} \tag{2.21}$$

Applying (2.19) and (2.21), the first row in (2.18) becomes [16, p. 15f.]:

$$\nabla^2_{\tilde{\mathbf{z}}}\mathcal{L}\left(\tilde{\mathbf{z}}_k,\boldsymbol{\mu}_k;\boldsymbol{\theta},\mathbf{q}\right)\cdot\Delta\tilde{\mathbf{z}}+\nabla_{\tilde{\mathbf{z}}}\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k;\boldsymbol{\theta},\mathbf{q}\right)^{\mathsf{T}}\cdot\underbrace{\left(\Delta\tilde{\boldsymbol{\mu}}+\boldsymbol{\mu}_k\right)}_{\boldsymbol{\mu}_{k+1}}=-\nabla_{\tilde{\mathbf{z}}}J\left(\tilde{\mathbf{z}}_k;\mathbf{q}\right) \tag{2.22}$$

This can be inserted in (2.18) yielding:

$$\underbrace{\begin{bmatrix}\nabla^2_{\tilde{\mathbf{z}}}\mathcal{L}\left(\tilde{\mathbf{z}}_k,\boldsymbol{\mu}_k;\boldsymbol{\theta},\mathbf{q}\right) & \nabla_{\tilde{\mathbf{z}}}\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k;\boldsymbol{\theta},\mathbf{q}\right)^{\mathsf{T}}\\ \nabla_{\tilde{\mathbf{z}}}\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k;\boldsymbol{\theta},\mathbf{q}\right) & \mathbf{0}\end{bmatrix}}_{\text{KKT matrix}}\cdot\begin{bmatrix}\Delta\tilde{\mathbf{z}}\\ \boldsymbol{\mu}_{k+1}\end{bmatrix}=-\begin{bmatrix}\nabla_{\tilde{\mathbf{z}}}J\left(\tilde{\mathbf{z}}_k;\mathbf{q}\right)\\ \boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k;\boldsymbol{\theta},\mathbf{q}\right)\end{bmatrix} \tag{2.23}$$

The equation can be solved efficiently using linear solvers e.g., provided in the HARWELL SUBROUTINE LIBRARY[1]. These solvers make use of the sparsity pattern [17, ch. 7] and the symmetry to be able to solve the large-scale NLP. Interfaces to these solvers are also provided in the IPOPT NLP algorithm used in FALCON.m (Subsection 2.1.2.2). Equation (2.23) is furthermore beneficial as is shows the symmetry of the KKT matrix, directly updates the equality constraint LAGRANGE multipliers (i.e., an evaluation step is saved), and does not require the Jacobian of the LAGRANGE function, but only the Jacobian of the cost function. Take into account that the KKT matrix is invertible as long as the previously mentioned regularity conditions (Section 2.1), e.g., linear independence of constraints, hold and the Hessian of the LAGRANGE FUNCTION in feasible directions (e.g., only along equality constraints) is positive definite (i.e., the second order sufficient conditions are fulfilled).

It should be noted that the incorporation of inequality constraints poses the major difficulties within the application of the NEWTON method to the general NLP in (2.13). Methods to cope with this issue are introduced in Subsections 2.1.3 and 2.1.4. Generally, the working principle of an NLP optimizer (here, only with equality constraints for the sake of simplicity) can be summarized as in Algorithm 2.1.

Generally, the information flow, iteration scheme, and different parts of the NLP solution process can also be visualized as in Figure 2.2 (note that sensitive as well as uncertain parameters are removed for the sake of simplicity). The NLP solution process starts with the provision of an initial guess for the decision variable vector in (2.6). From this initial guess, and all subsequent iterations k, the states, controls, and time-invariant parameters can be extracted as the structure of the decision variable vector is known by (2.6). Knowing these values, the model as well as the constraints (including their

[1]HSL-IPOPT reference (Retrieved: April 23, 2019)

Algorithm 2.1 General working principle of an nonlinear program optimizer applying the NEWTON method to the Karush-Kuhn-Tucker conditions.

Require:

Initial guess for primal variables $\tilde{\mathbf{z}}_0$.

Initialize guess for LAGRANGE multipliers: $\boldsymbol{\mu}_0$

Termination conditions ϵ_{feas}, ϵ_{opt}, and $k_{\max}$ (i.e., the maximum number of iterations).

1: Initialize iteration counter: $k = 0$
2: Calculate KKT conditions for initial point: $\nabla_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_0, \boldsymbol{\mu}_0\right)$ and $\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_0; \boldsymbol{\theta}, \mathbf{q}\right)$
3: **while** $k < k_{\max} \mid \left(||\nabla_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_k, \boldsymbol{\mu}_k\right)||_\infty > \epsilon_{\text{opt}} \;\; \& \; ||\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k; \boldsymbol{\theta}, \mathbf{q}\right)||_\infty > \epsilon_{\text{feas}}\right)$ **do**
4: Evaluate: $\nabla^2_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_k, \boldsymbol{\mu}_k\right)$ and $\nabla_{\tilde{\mathbf{z}}} \boldsymbol{\psi}\left(\tilde{\mathbf{z}}_k; \boldsymbol{\theta}, \mathbf{q}\right)$
5: Calculate an update of the primal and the dual variables by (2.18) or (2.23).
6: Calculate the new iterates $\tilde{\mathbf{z}}_{k+1}$ and $\boldsymbol{\mu}_{k+1}$ by (2.19).
7: Calculate KKT conditions for new point: $\nabla_{\tilde{\mathbf{z}}} \mathcal{L}\left(\tilde{\mathbf{z}}_{k+1}, \boldsymbol{\mu}_{k+1}\right)$ and $\boldsymbol{\psi}\left(\tilde{\mathbf{z}}_{k+1}; \boldsymbol{\theta}, \mathbf{q}\right)$
8: Increase counter: $k = k + 1$
9: **end while**

10: **return** Optimal decision variable solution $\tilde{\mathbf{z}}_{\text{opt}}$ with corresponding optimal LAGRANGE multipliers $\boldsymbol{\mu}_{\text{opt}}$, and optimal cost J_{opt}.

Jacobian and Hessian) can be evaluated. Then, the information can be used to calculate the objective vector in (2.11) and the corresponding Jacobian and Hessian. This Jacobian and Hessian information is used in the NLP solver to check the convergence (see (2.20)) and update the decision variables (see (2.19) and (2.23)) to get a new iterate $\tilde{\mathbf{z}}_{k+1}$ that is used for the next iteration. This procedure is iterated until (2.20) is fulfilled and a candidate for an optimal point $\tilde{\mathbf{z}}_{\text{opt}}$ has been calculated.

An important aspect of the application of the NEWTON-type method in an NLP is that the update step in (2.18) or (2.21) requires the availability of the Jacobian of the cost function as well as the Hessian of the LAGRANGE function. Additionally, the Jacobian of the constraints is required. Although the Hessian can e.g., be approximated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [48, p. 345ff.], which in turn requires the Jacobian of the LAGRANGE function, the Jacobians must be available analytically. To summarize, the Jacobian of the objective vector with respect to the optimization parameter vector $\tilde{\mathbf{J}} = \frac{\mathrm{d}\tilde{\mathbf{F}}}{\mathrm{d}\tilde{\mathbf{z}}}$ is required at a minimum, while the Hessian of the objective vector with respect to the optimization parameter vector $\tilde{\mathbf{H}} = \frac{\mathrm{d}^2\tilde{\mathbf{F}}}{\mathrm{d}\tilde{\mathbf{z}}^2}$ is beneficial.

The following paragraphs state the derivation of the analytical Jacobian for the CD (see (2.10)) with respect to the optimization parameters (see (2.6)), which is the main part of the objective vector (see (2.11)). At first, the derivative with respect to the final time is given as follows:

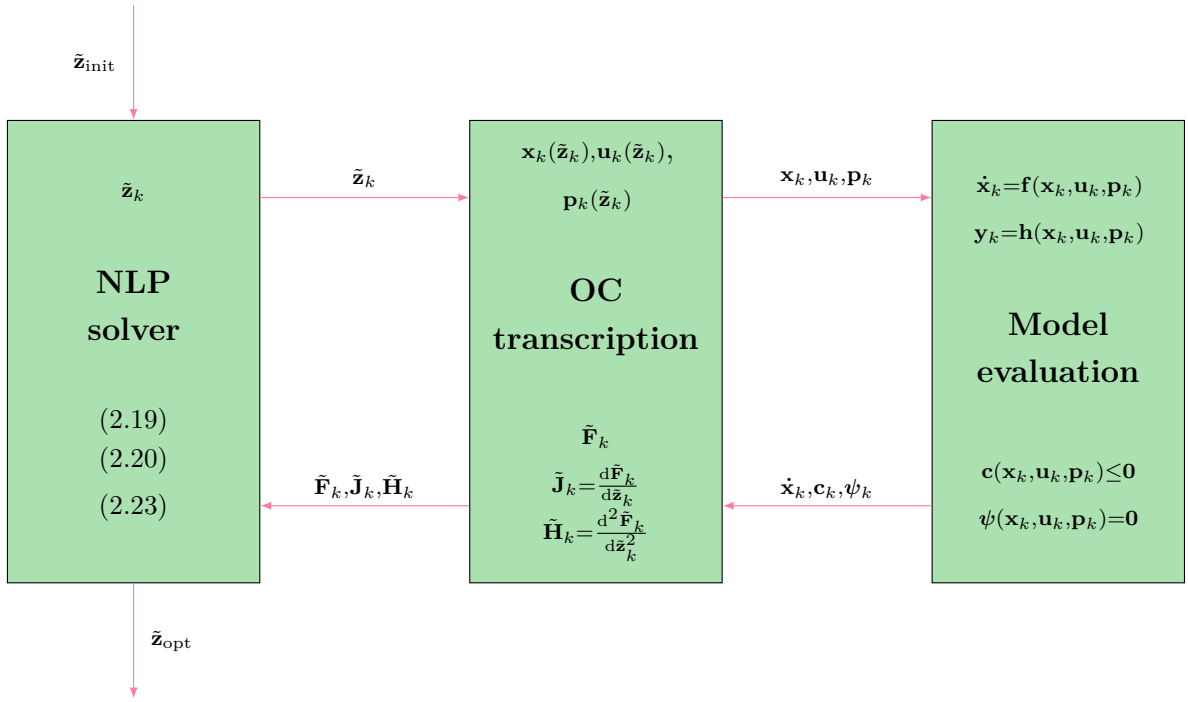

Figure 2.2: *Information flow and general iteration procedure in nonlinear program solution process (after [64], [17, p. 73]).*

$$\frac{\partial \mathbf{CD}_{i,i+1}}{\partial t_f} = -\frac{h_\tau}{2} \cdot \underbrace{(\dot{\mathbf{x}}_{i+1}}_{\mathbf{f}(\mathbf{x}_{i+1},\mathbf{u}_{i+1},\mathbf{p};\boldsymbol{\theta},\mathbf{q})} + \underbrace{\dot{\mathbf{x}}_i)}_{\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{p};\boldsymbol{\theta},\mathbf{q})} \tag{2.24}$$

The derivative with respect to the i-th collocation time point for states and controls is calculated as follows:

$$\frac{\partial \mathbf{CD}_{i,i+1}}{\partial \mathbf{x}_i} = -\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_i} - h_\tau \frac{t_f}{2}\frac{\partial \dot{\mathbf{x}}_i}{\partial \mathbf{x}_i} = -\mathbf{I} - h_\tau \frac{t_f}{2}\mathbf{J}_{\mathbf{x}_i} \tag{2.25a}$$

$$\frac{\partial \mathbf{CD}_{i,i+1}}{\partial \mathbf{u}_i} = -h_\tau \frac{t_f}{2}\frac{\partial \dot{\mathbf{x}}_i}{\partial \mathbf{u}_i} = -h_\tau \frac{t_f}{2}\mathbf{J}_{\mathbf{u}_i} \tag{2.25b}$$

The identity matrix of appropriate size is depicted by $\mathbf{I}$. The symbol $\mathbf{J}$ is used as an abbreviation for the Jacobian of the state dynamics. The index symbolizes the vector (e.g., states $\mathbf{x}$) with respect to whom the Jacobian was derived and evaluated. It is noted here that the FALCON.m toolbox provides all model Jacobians in an analytic manner leading to a fast and accurate solution of the OCP [111].

Similar to (2.25a)–(2.25b), the derivative of the $(i+1)$-th collocation time point for states and controls is derived as follows:

$$\frac{\partial \mathbf{CD}_{i,i+1}}{\partial \mathbf{x}_{i+1}} = \frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_{i+1}} - h_\tau \frac{t_f}{2}\frac{\partial \dot{\mathbf{x}}_{i+1}}{\partial \mathbf{x}_{i+1}} = \mathbf{I} - h_\tau \frac{t_f}{2}\mathbf{J}_{\mathbf{x}_{i+1}} \tag{2.26a}$$

$$\frac{\partial \mathbf{CD}_{i,i+1}}{\partial \mathbf{u}_{i+1}} = -h_\tau \frac{t_f}{2}\frac{\partial \dot{\mathbf{x}}_{i+1}}{\partial \mathbf{u}_{i+1}} = -h_\tau \frac{t_f}{2}\mathbf{J}_{\mathbf{u}_{i+1}} \tag{2.26b}$$

In addition to the CD, the derivatives of the path/point constraints of e.g., the model outputs are required. The calculation is simple, as these do not depend on an integration scheme and are only related to a single time step. The equations are as follows:

$$\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_i} = \mathbf{J}_{\mathbf{y},\mathbf{x}_i} \tag{2.27a}$$

$$\frac{\partial \mathbf{y}_i}{\partial \mathbf{u}_i} = \mathbf{J}_{\mathbf{y},\mathbf{u}_i} \tag{2.27b}$$

The Jacobian $\mathbf{J}_\mathbf{y}$ with respect to the outputs $\mathbf{y}$ is considered and evaluated with respect to the states or controls (stated by the second index).

A feature of the collocation method is creating a sparse gradient. This sparsity pattern is illustrated in Table 2.1. Optimizable time-invariant parameters are neglected here for the simplicity of expression although they can be incorporated in a manner similar to that of the final time (with variable sparsity influence). Here, the banded structure as well as the multiple zeros from discretized times that do not influence another discretized time is observed (e.g., the first discretized time step never depends on the final discretized time step). The cost function influence is illustrated by asterisks since the derivative is dependent on the relevant cost terms (i.e., Mayer or Lagrange).

Take into account that the Hessian can be derived similarly. This is omitted here for the sake of brevity and only conducted afterward for the gPC collocation scheme (Chapter 5). Generally, the introduced baseline trapezoidal collocation scheme of this section is extended in Chapter 5 to directly incorporate the gPC method in order to solve ROCPs. Here, the structure (e.g., sparsity) should be conserved as an efficient solution for this structure is possible in the NLP. The next section introduces FALCON.m, which applies the given trapezoidal collocation scheme.

2.1.2.2 FSD optimAL CONtrol tool for MATLAB: FALCON.m

The FALCON.m core framework [111] was developed during different Ph.D. theses at the FSD [17, 110]. It is a publicly available free-of-charge OC tool for Matrix Laboratory® (MATLAB®) that uses direct OC techniques with full discretization and NEWTON-type NLP solvers such as IPOPT [131] (Subsection 2.1.3) and Sparse Nonlinear Optimizer (SNOPT) [53] (Subsection 2.1.4). Within this work, the FALCON.m framework is extended by the later on introduced uncertainty analysis methods (Sections 2.2 and 2.3) to create a robust add-on to the basic core framework. It should be noted that the developed robust add-on is currently not yet part of the public FALCON.m version and only available for internal use at the FSD.

Generally, FALCON.m uses the trapezoidal collocation method introduced in Subsection 2.1.2.1. The analytic model Jacobian and Hessian are provided by a symbolic differentiation of the model subsystems using the chain rule [110, p. 86ff.]. This allows for a fast and reliable provision of the model Jacobian and Hessian to the NEWTON-type

Table 2.1: *Sparsity pattern of standard trapezoidal collocation gradient* $\tilde{\mathbf{J}} = \frac{\mathrm{d}\tilde{\mathbf{F}}}{\mathrm{d}\tilde{\mathbf{z}}}$.

$\tilde{\mathbf{J}}$	t_f	$\mathbf{x}_1$	$\mathbf{u}_1$	$\mathbf{x}_2$	$\mathbf{u}_2$	$\mathbf{x}_3$	$\mathbf{u}_3$	$\dots$	$\mathbf{x}_{n-1}$	$\mathbf{u}_{n-1}$	$\mathbf{x}_n$	$\mathbf{u}_n$
J	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$\dots$	$*$	$*$	$*$	$*$
$\mathbf{y}_1$	0	(2.27a)	(2.27b)	0	0	0	0	$\dots$	0	0	0	0
$\mathbf{CD}_{1,2}$	(2.24)	(2.25a)	(2.25b)	(2.26a)	(2.26b)	0	0	$\dots$	0	0	0	0
$\mathbf{y}_2$	0	0	0	(2.27a)	(2.27b)	0	0	$\dots$	0	0	0	0
$\mathbf{CD}_{2,3}$	(2.24)	0	0	(2.25a)	(2.25b)	(2.26a)	(2.26b)	$\dots$	0	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{y}_{n-1}$	0	0	0	0	0	0	0	$\dots$	(2.27a)	(2.27b)	0	0
$\mathbf{CD}_{n-1,n}$	(2.24)	0	0	0	0	0	0	$\dots$	(2.25a)	(2.25b)	(2.26a)	(2.26b)
$\mathbf{y}_n$	0	0	0	0	0	0	0	$\dots$	0	0	(2.27a)	(2.27b)

NLP optimizer [110, p. 126ff.]. The model evaluation routine is then coded to a MEX[2] to provide a fast evaluation. In case the model Hessian is not provided, the NLP solver (e.g., IPOPT) uses its readily implemented limited-memory BFGS update to approximate the Hessian [131, p. 39].

It should be noted here that FALCON.m also provides numerical scaling and offset of the optimization and residual parameters [111] to achieve a better conditioned NLP. This conditioning means that all parameters should have approximately the same magnitude and thus, influence. Generally, the optimization parameters in (2.6) and the residual vector in (2.11) are transformed for the NLP optimizer as follows:

$$\begin{aligned} \tilde{\mathbf{z}}_{\mathrm{sq}} &= \mathrm{diag}\left[\tilde{\mathbf{z}}_{\mathrm{sc}}\right] \cdot \left(\tilde{\mathbf{z}} - \tilde{\mathbf{z}}_{\mathrm{off}}\right) \\ \tilde{\mathbf{F}}_{\mathrm{sq}} &= \mathrm{diag}\left[\tilde{\mathbf{F}}_{\mathrm{sc}}\right] \cdot \left(\tilde{\mathbf{F}} - \tilde{\mathbf{F}}_{\mathrm{off}}\right) \end{aligned} \tag{2.28}$$

Here, $\mathrm{diag}\left[\cdot\right]$ is the operator to create a diagonal matrix from a vector. The constant scaling and offset vector for the optimization parameters are denoted by $\tilde{\mathbf{z}}_{\mathrm{sq}}$ and $\tilde{\mathbf{z}}_{\mathrm{off}}$ respectively, while the corresponding vectors for the residual vector are given by $\tilde{\mathbf{F}}_{\mathrm{sq}}$ and $\tilde{\mathbf{F}}_{\mathrm{off}}$. The scaled optimization parameter and residual vector are denoted by $\tilde{\mathbf{z}}_{\mathrm{sq}}$ and $\tilde{\mathbf{F}}_{\mathrm{sq}}$ in (2.28) respectively. For convenience, the notation for the physical, i.e., non-scaled, values is used throughout the thesis, while keeping in mind that the NLP solver only "sees" the scaled problem in (2.28). Here, (2.28) also directly implies that the Jacobian and Hessian must be scaled, which is easily done applying the definition in (2.28) and the knowledge that the scaling and offset vectors are constant (i.e, their derivative is zero).

[2]MATLAB mex documentation (Retrieved: April 23, 2019)

The basic problem structure of FALCON.m, which is implemented in object-oriented MATLAB®, is illustrated in Figure 2.3. Here, it is also already depicted how the uncertainty analysis methods are incorporated into the FALCON.m OCP structure: On the one hand, the uncertain parameters are added to the model parameters to allow for a model evaluation at specific uncertain values. On the other hand, it is clear that special discretization techniques are implemented that are then directly used to solve the uncertain OCP. Furthermore, it can be seen that the OCP also directly carries all information on the statistical properties of the optimal results.

The following listing summarizes the different elements of Figure 2.3 as well as their purpose in FALCON.m and for the solution of the NLP (a more detailed introduction can also be found in [17, p. 102ff.] or [110, p. 73ff.]):

`Problem`: Main instance of the OCP in FALCON.m. The `Problem` interacts with the NLP solver and contains the information for its solution.

- `Phases`: Handle the general structure of the OCP, i.e., states, controls, boundary conditions, or models. Multiple phases can be connected to optimize e.g., a varying dynamic model or multiple waypoints (i.e., different boundary conditions)
  - `Model`: Handles all model relevant connections and interactions, e.g., definition and evaluation.
    - ◇ `ModelHandle`: Function handle to the implemented model dynamics (created e.g., by the FALCON.m simulation model builder). Calculates the state derivatives and outputs with their corresponding Jacobian and Hessian.
    - ◇ `StateDotGrid`: Time history of the state derivatives with the Jacobian and Hessian.
    - ◇ `ModelOutputGrid`: Time history of the model outputs with the Jacobian and Hessian.
    - ◇ `ModelParameters`: Parameters required to evaluate the model, including the optimization, sensitive, and uncertain parameters.
  - `StateGrid`: Time history of the states with bounds for the OCP. If no initial guess is provided, FALCON.m creates one based on the bounds.
  - `ControlGrids`: Time history of the controls with bounds for the OCP. There can be multiple grids with different time discretizations. If no initial guess is provided, FALCON.m creates one based on the bounds.
  - `DefectGrid`: Time history of the CD (see (2.10)) for the OCP.
  - `PathConstraints`: Contains all path constraints, i.e., constraints that are enforced at each point of the discretized time grid, their boundaries, numeric values, and evaluation functions.

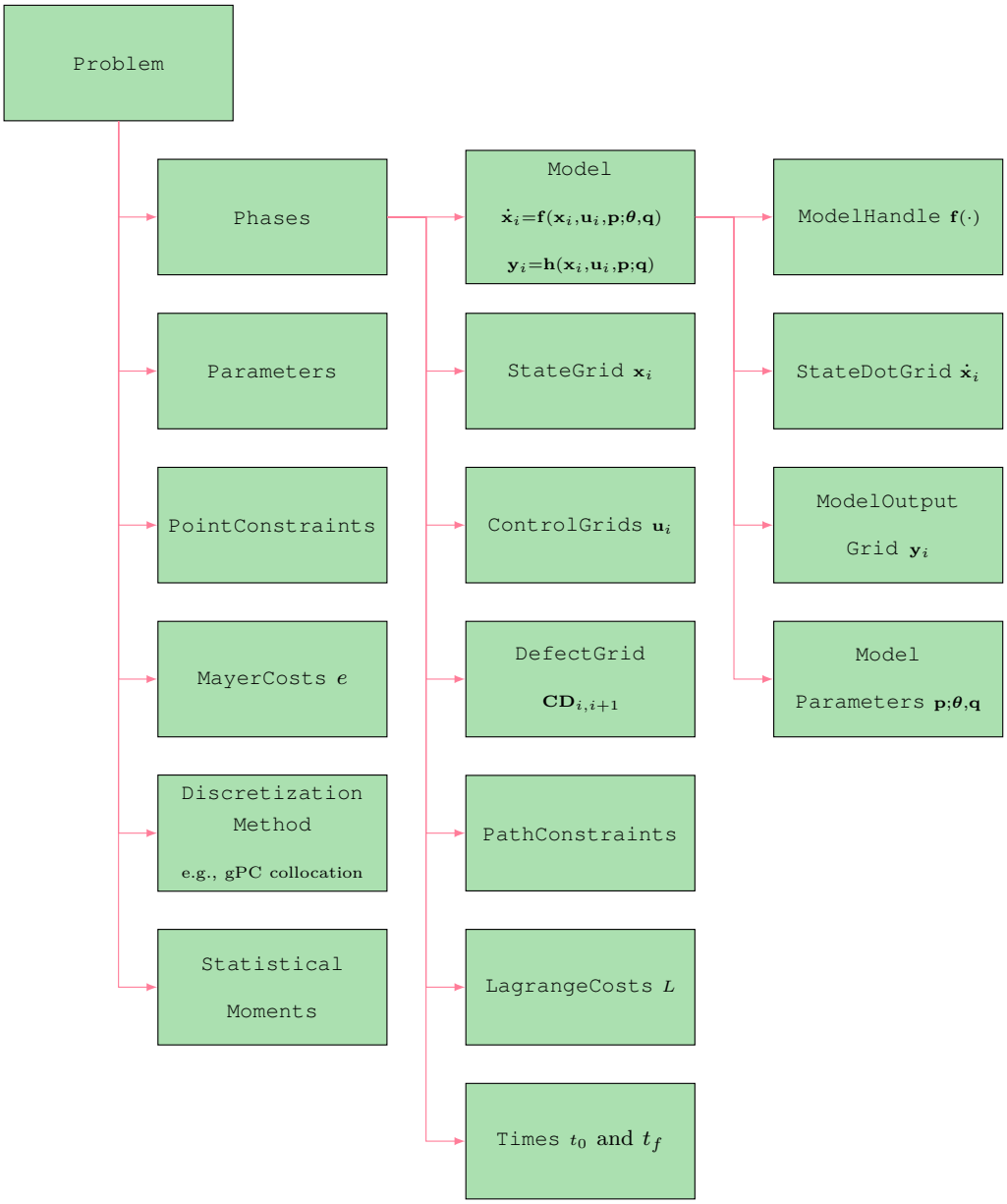

Figure 2.3: *Object-oriented, class-based structure of FALCON.m framework with a top-level optimal control problem, phases, cost and constraint functions, dynamic models, as well as the incorporation of random parameters (adapted from [17, p. 103]).*

  - `LagrangeCosts`: Contains all Lagrange cost terms, i.e., all running costs, with the numeric values and evaluation functions.
  - `Times`: Initial and final time of the phase.

- `Parameters`: Contains all optimizable, non-optimizable, and uncertain parameters, their boundaries, and their numeric values. An initial value is always provided by the user.
- `PointConstraints`: Contains all point constraints, i.e., constraints that are only enforced at specific time points, but can connect multiple phases, their boundaries, numeric values, and evaluation functions.
- `MayerCosts`: Contains all Mayer cost terms, e.g., cost influence at a specific time point of the optimization interval, with the numeric values and evaluation functions.
- `DiscretizationMethod`: Specifies the method how the discretized NLP is created from the continuous OCP (e.g., trapezoidal collocation in Subsection 2.1.2.1)
- `StatisticalMoments`: Array that contains the time history of the statistical moments, e.g., mean and variance, for states, controls, and outputs.

After this general introduction, the NEWTON-type NLP optimizers, used by FALCON.m, are introduced in the following.

2.1.3 Interior-Point Optimization

Generally, interior point (IP) optimization, which is most often applied in the context of barrier optimization, relies on the principle of introducing inequality constraints into the cost function by a logarithmic barrier function. The barrier goes to infinity close to the boundary of the inequality constraint and therefore, a point close to the boundary becomes unlikely to be optimal as it then has a large cost value. Using a more sophisticated methodology, which makes it easier to deal with general inequality constraints, the introduction of the inequality constraints in the cost function can be done by slack variables s. This results in the following OCP formulation [17, p. 49], [133]:

$$\begin{aligned} \min_{\tilde{\mathbf{z}},\,\mathbf{s}} \quad & J\left(\tilde{\mathbf{z}}\right) - \mu \sum_{i=1}^{n_\mathbf{c}} \ln\left(s_i\right) \\ \text{s.t.} \quad & \tilde{\mathbf{z}}_{\text{lb}} \leq \tilde{\mathbf{z}} \leq \tilde{\mathbf{z}}_{\text{ub}}, \\ & \mathbf{c}(\tilde{\mathbf{z}}) + \mathbf{s} = \mathbf{0}, \qquad s_i > 0, \quad i = 1, \ldots, n_c, \\ & \boldsymbol{\psi}(\tilde{\mathbf{z}}) = \mathbf{0} \end{aligned} \tag{2.29}$$

The solution of the IP-OCP in (2.29) can now be obtained by e.g., the NEWTON method. Therefore, the barrier parameter μ is steadily decreased until the influences of the logarithmic term in the cost function vanishes to recover the originally desired cost function

and its optimal point. For more details one can consult [131–133] or [16, p. 73ff.]. IP methods are e.g., used within the NLP solver IPOPT [131]. It should be noted here that the NEWTON method, and thus, the use of a NEWTON-type optimization scheme, does not ensure to find a global optimum.

The major benefit is the applicability to large scale OCPs present in trajectory optimization, as a determination of active and inactive constraints is automatically done by the barrier term. Thus, the set of active constraints $\mathbb{A}$ does not need to be calculated. Additionally, primal-dual barrier methods like IPOPT use the barrier approximation in the complementary slackness condition (KKT Condition 4) rather than the cost. Thus, these algorithms solve the same OCP [133].

Take into account that the extension of the NEWTON step in (2.23) for the IP method is not shown here for the sake of brevity. The derivation can be found in e.g., [133, p. 29f.] and is similar to the already introduced formula in (2.23).

2.1.4 Sequential Quadratic Programming

The sequential quadratic programming (SQP) methods are based on a local quadratic optimization of the OCP as described by the NEWTON method. Therefore, the constraints are at first linearized as follows [53, p. 7]:

$$\begin{aligned} \mathbf{c}\left(\tilde{\mathbf{z}}_*\right) &\approx \mathbf{c}\left(\tilde{\mathbf{z}}\right) + \nabla_{\tilde{\mathbf{z}}}\mathbf{c}\left(\tilde{\mathbf{z}}\right)^\mathsf{T} \underbrace{\left(\tilde{\mathbf{z}}_* - \tilde{\mathbf{z}}\right)}_{\mathbf{d}} \\ \boldsymbol{\psi}\left(\tilde{\mathbf{z}}_*\right) &\approx \boldsymbol{\psi}\left(\tilde{\mathbf{z}}\right) + \nabla_{\tilde{\mathbf{z}}}\boldsymbol{\psi}\left(\tilde{\mathbf{z}}\right)^\mathsf{T} \underbrace{\left(\tilde{\mathbf{z}}_* - \tilde{\mathbf{z}}\right)}_{\mathbf{d}} \end{aligned} \tag{2.30}$$

Here, $\mathbf{d}$ is called SQP step in the following and describes the difference from the current parameter vector $\tilde{\mathbf{z}}$ to an updated vector $\tilde{\mathbf{z}}_*$. The SQP step is optimized in the SQP method.

In addition to linearizing the constraints, a quadratic cost function based on the second derivative of LAGRANGE function in (2.3) and the original cost function in (2.2) is defined [53, p. 7]:

$$\check{J} = \frac{1}{2}\mathbf{d}^\mathsf{T}\nabla^2_{\tilde{\mathbf{z}}}\mathcal{L}\left(\tilde{\mathbf{z}}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right)\mathbf{d} + \nabla_{\tilde{\mathbf{z}}}J\left(\tilde{\mathbf{z}}\right)^\mathsf{T}\mathbf{d} \tag{2.31}$$

It should be noted that this definition of the cost function ensures that the KKT update in (2.23) is recovered by the SQP method and thus, the SQP method directly applies the NEWTON method without simplification. This can be seen, when looking at the SQP LAGRANGE function (with only equality constraints for the sake of simplicity):

$$\check{\mathcal{L}}\left(\tilde{\mathbf{z}}, \boldsymbol{\mu}\right) = \frac{1}{2}\mathbf{d}^\mathsf{T}\nabla^2_{\tilde{\mathbf{z}}}\mathcal{L}\left(\tilde{\mathbf{z}}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right)\mathbf{d} + \nabla_{\tilde{\mathbf{z}}}J\left(\tilde{\mathbf{z}}\right)^\mathsf{T}\mathbf{d} + \boldsymbol{\mu}^\mathsf{T}\boldsymbol{\psi}\left(\tilde{\mathbf{z}}; \boldsymbol{\theta}, \mathbf{q}\right) \tag{2.32}$$

This LAGRANGE function must fulfill the following KKT conditions:

$$\begin{aligned} \nabla_{\mathbf{d}} \check{\mathcal{L}} \left(\tilde{\mathbf{z}}_{\text{opt}}, \boldsymbol{\mu}_{\text{opt}} \right) = \nabla^2_{\tilde{\mathbf{z}}} \mathcal{L} \left(\tilde{\mathbf{z}}, \boldsymbol{\lambda}, \boldsymbol{\mu} \right) \mathbf{d} + \nabla_{\tilde{\mathbf{z}}} J \left(\tilde{\mathbf{z}} \right)^{\mathsf{T}} + \boldsymbol{\mu}^{\mathsf{T}} \nabla_{\mathbf{d}} \boldsymbol{\psi} \left(\tilde{\mathbf{z}}; \boldsymbol{\theta}, \mathbf{q} \right) \overset{!}{=} \mathbf{0} \\ \nabla_{\mathbf{d}} \boldsymbol{\psi} \left(\tilde{\mathbf{z}}_{\text{opt}}; \boldsymbol{\theta}, \mathbf{q} \right) \overset{!}{=} \mathbf{0} \end{aligned} \tag{2.33}$$

Then, the connection to the standard NEWTON method applied to the KKT conditions (first row of (2.23)) can be seen when using the connection between the optimization parameters and the SQP step as introduced in (2.30). After subtracting the derivative of the cost function from both sides and exchanging the transposition of LAGRANGE multipliers and equality constraints, the update for the SQP step created in (2.33) is identical with the update in (2.23). Thus, quadratic programming is a reasonable choice to solve an NLP [51, p. 219f.].

Then, combining (2.30) and (2.31) gives the following quadratic programming minimization problem with $\mathbf{d}$ as the SQP step size vector [17, p. 48]:

$$\begin{aligned} \min_{\mathbf{d}} \quad & \frac{1}{2} \mathbf{d}^{\mathsf{T}} \nabla^2_{\tilde{\mathbf{z}}} \mathcal{L} \left(\tilde{\mathbf{z}}, \boldsymbol{\lambda}, \boldsymbol{\mu} \right) \mathbf{d} + \nabla_{\tilde{\mathbf{z}}} J \left(\tilde{\mathbf{z}} \right)^{\mathsf{T}} \mathbf{d} \\ \text{s.t.} \quad & \tilde{\mathbf{z}}_{\text{lb}} \leq \tilde{\mathbf{z}} + \mathbf{d} \leq \tilde{\mathbf{z}}_{\text{ub}}, \\ & \mathbf{c}_i \left(\tilde{\mathbf{z}} \right) + \nabla_{\tilde{\mathbf{z}}} \mathbf{c}_i \left(\tilde{\mathbf{z}} \right)^{\mathsf{T}} \mathbf{d} \leq 0, \quad i = 1, \ldots, n_c, \\ & \boldsymbol{\psi}_j \left(\tilde{\mathbf{z}} \right) + \nabla_{\tilde{\mathbf{z}}} \boldsymbol{\psi}_j \left(\tilde{\mathbf{z}} \right)^{\mathsf{T}} \mathbf{d} = 0, \quad j = 1, \ldots, n_\psi \end{aligned} \tag{2.34}$$

This quadratic programming problem is then solved in each NLP iteration. Further descriptions of the method can be found in e.g., [53], [16, p. 60ff.], or [51, p. 227ff.]. This method is e.g., used in the NLP solver package SNOPT [53]. Again, only locally optimal solutions are found, because of the application of the NEWTON method.

A benefit of the SQP method is the generally fast convergence and numerically exact solution of the original problem. A disadvantage is often the size of the OCPs that can be solved, as SQP methods often apply active set strategies, i.e., direct tracking of active and inactive constraints, and thus, the tracking of the set $\mathbb{A}$ of the active and inactive inequality constraints can become an issue. Still, SQP methods can also be designed using an IP strategy (see Subsection 2.1.3): For such SQP IP designs, the major difference to basic IP methods is the number of required function evaluations in an NLP step, which is mostly problem dependent and thus, no general statement on the better-suited method can be made.

Take into account that the extension of the NEWTON step in (2.23) for SQP methods can be found in [53, p. 8ff.].

2.2 Sampling-based Methods for Uncertainty Analysis

This section gives an overview on classic/sampling-based methods for uncertainty analysis such as Monte Carlo analysis (MCA) (Subsection 2.2.1) and Latin hypercube sampling (LHS) (Subsection 2.2.2) with their confidence interval (CI) determination (Subsection 2.2.3). These methods are used to verify the results of the more advanced gPC method introduced in Section 2.3, which is used in the OCP for uncertainty analysis. Therefore, only the general procedure of the methods is explained in the following. The interested reader is referred to [54, 82, 89–91, 96, 125] for a more detailed insight. Generally, the classic methods rely on the principle of sampling from the RV PDF and are therefore also called SAMPLING METHODS.

Concluding the classic sampling methods, Subsection 2.2.4 introduces a method to conduct an uncertainty analysis specifically tailored for rare-events with the subset simulation (SubSim). This method is ultimately an evolution of MCA applying Markov chain Monte Carlo (MCMC) techniques and required in this thesis for the rare-event CC-OC framework (Chapter 6).

2.2.1 Monte Carlo Analysis

Monte Carlo analysis (MCA) is a very simple, but powerful tool for uncertainty analysis of a system [54, 89–91, 125], which in principle relies on the central limit theorem (CLT) (Theorem B.13). Thus, MCA is based on a deterministic sampling of the model multiple thousand or even ten thousands of times. The sampling relies on a random number generation from the PDF of the uncertain parameters.

For the case of trajectory optimization, this means that the OCP in (2.1) must be solved at each of these sampling points. This procedure is, in general, very time consuming, although it can be parallelized. It is therefore not applicable for our purpose of calculating robust trajectories in an efficient manner. The long lasting convergence history is based on the fact that the CLT only converges with the square root of the number of samples n_s to the real solution. In order to partially overcome the issues of MCA, Subsection 2.2.2 introduces LHS that is ultimately a MCA with guarantees that the generated samples are representative for the real PDF.

After sampling the system response, statistical moments (Appendix B.2.2) can be easily calculated from MCA. For instance, the mean/expected value of general system outputs $\mathbf{y}$ is given as follows:

$$\mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \mu\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \mu_{\mathbf{y}} \approx \frac{1}{n_s}\sum_{i=1}^{n_s}\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}^{(i)}\right) \tag{2.35}$$

Here, n_s is the number of samples used in the MCA and $\boldsymbol{\theta}^{(i)}$ is a vector of deterministic samples drawn from the single PDF, i.e., created from a sample $\theta_j^{(i)}$ where $\theta_j \sim \rho_{\Theta_j}(\theta_j)$. Furthermore, $\rho_{\Theta_j}(\theta_j)$ is the PDF of the j-th random parameter. The system response, i.e., OCP result, at the deterministic sample $\boldsymbol{\theta}^{(i)}$ is denoted by $\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}^{(i)}\right)$. It should be noted that MCA yields an unbiased estimator, i.e., the exact solution is recovered [81, p. 14].

The variance, i.e., the square of the standard deviation σ, has an equally simple representation using the mean value in (2.35) as follows:

$$\begin{aligned}\sigma^2\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \sigma_{\mathbf{y}}^2 = \operatorname{var}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] &= \mathbb{E}\left[\left(\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) - \mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\right)^2\right] \\ &\approx \frac{1}{n_s - 1}\sum_{i=1}^{n_s}\left\{\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}^{(i)}\right) - \mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\right\}^2\end{aligned} \tag{2.36}$$

Thus, a statistical analysis of the OCP is fairly easy, but time-consuming, using a MCA because the solution of the OCP at the random samples normally takes a significant amount of time.

Another issue with the MCA is that the statistical analysis assumes that samples from the complete uncertainty domain have been taken. In OCPs, this can generally never be assured due to the constraints (see (2.1)) that might result in infeasible solutions (e.g., when considering CCs). These yield that some parameter combinations might create infeasible results which must be discarded for the statistical analysis as they can be unphysical. This is visualized in Figure 2.4: Here, a standard UNIFORM, i.e., PDF $\theta \sim \mathcal{U}(a = -1, b = 1)$, is used as the uncertainty and the output is given by an exponential relation. As a constraint, the output is not allowed to be larger than exp(0.95), which is also visualized by the dotted magenta lines. The feasible function values are given by the blue line, while the infeasible are denoted by the red line. These infeasible values are discarded for the statistical analysis, as especially in a NEWTON-type NLP scheme based on collocation, an infeasible result suggests an unphysical results, which should not be used for analysis purposes. This discarding, especially when it is located in a specific domain of the PDF, introduces a bias in the statistical moment estimation is introduced. This is visualized in Figure 2.4 by the error bars. The magenta error bar visualizes the standard deviation around the real mean value (i.e., the value calculated by sampling over the whole uncertainty domain not considering the constraint), while the blue error bar shows the standard deviation around the constrained mean value. It is clear that both the mean as well as the standard deviation are biased in the constrained case. Depending on the nonlinearity and the number of uncertainties this bias might be difficult to describe. Thus, an MCA must be used with caution in OC and the results must be critically reviewed, especially when infeasible solutions are encountered.

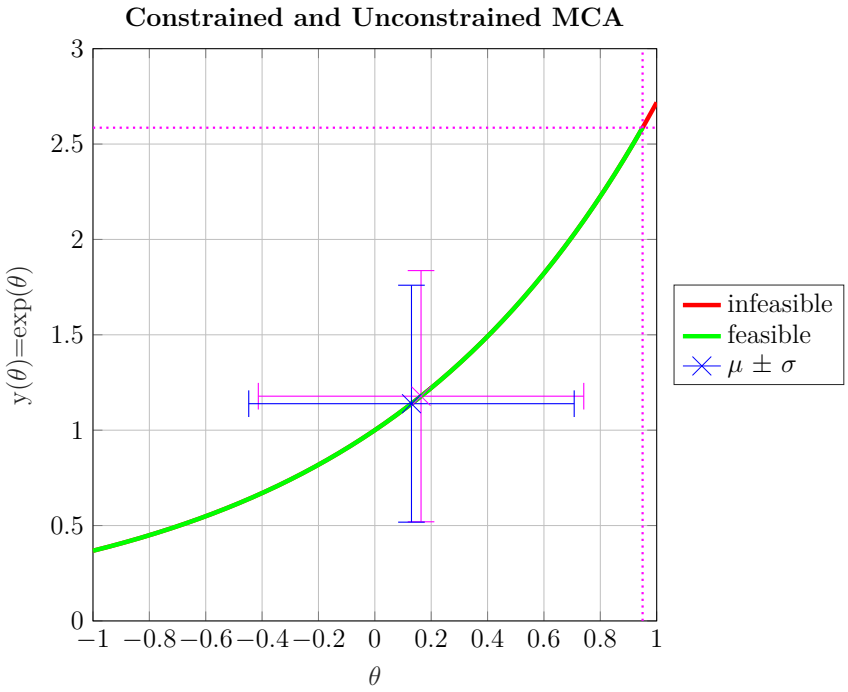

Figure 2.4: *Comparison of constrained and unconstrained Monte Carlo analysis with calculated mean and standard deviation.*

2.2.2 Latin Hypercube Sampling

Ultimately, Latin hypercube sampling (LHS) is a stratified way of conducting a MCA [82, 96]. It still relies on the deterministic sampling and the CLT [96], but stratifies the procedure of deciding at which points to sample, i.e., how the vector of random samples is drawn. While MCA relies on random (or at least on pseudo-random) sampling, LHS relies on sampling within hypercubes. These hypercubes basically split up the uncertainty domain and secure that the amount of samples taken from a specific domain of the uncertain input PDF corresponds to their probability of appearance. For a UNIFORM PDF, these domains are defined by the following intervals [82]:

$$\left[0, \frac{1}{n_s}\right], \left[\frac{1}{n_s}, \frac{2}{n_s}\right], \ldots, \left[\frac{n_s - 1}{n_s}, 1\right] \tag{2.37}$$

By this procedure, it is secured that exactly one sample from each domain is taken and that the original UNIFORM PDF is recovered by the LHS approach. It is important to note that the sampling procedure in (2.37) can be extended to more general domains [82] and multiple dimensions, which is skipped here for the sake of brevity. Additionally, it should be noted that the LHS procedure again yields an unbiased estimator [82, p. 2059], which is important because it is an estimator that has no bias ("offset") between the real and the estimated statistical moments.

In Figure 2.5, the samples created from a UNIFORM PDF by MCA (left plot) and LHS (right plot) are depicted by the blue crosses (25 samples). Here, the LHS domain grid, defined in (2.37), is visualized in the right plot using the dashed green lines. It

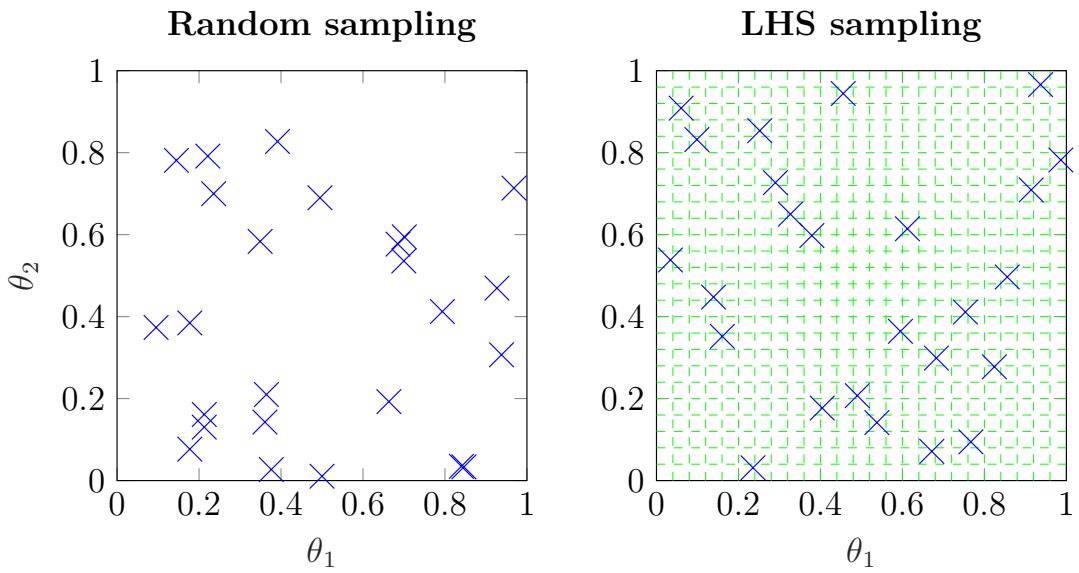

Figure 2.5: *Comparison of Latin-hypercube sampling and Monte Carlo analysis created samples (25 samples; blue crosses) for* UNIFORM *distribution with hypercube grid (green lines) for Latin-hypercube sampling.*

is clear that the LHS sampling produces better distributed samples, i.e., has a better spread over the uncertainty domain. It is also clear that, by splitting the domain, there is generally a minimum separation between the different samples (i.e, they are not clustered like in the random sampling at e.g., the right corner) and there is exactly one sample in each one-dimensional grid interval. Thus, the LHS method "explores" the PDF in a more reasonable way and consequently creates a better statistical moment approximation with fewer samples.

Although the sampling procedure in (2.37) makes the method faster converging, still hundreds or thousands of samples may be required to achieve a good convergence. Then, the mean and standard deviation can, again, be calculated by (2.35) and (2.36). Additionally, the LHS methods suffers from the same problem within constrained optimization as the MCA method, i.e., a bias when infeasible results are encountered (Figure 2.4).

2.2.3 Confidence Intervals

To assess the accuracy of both MCA (Subsection 2.2.1) and LHS (Subsection 2.2.2), confidence intervals (CIs) are a very common choice [81, 125]. Generally, a CI is the interval around the calculated statistical moment that contains the true value with a certain confidence level $1 - \xi$.

For the mean value, the CI is defined as follows [125, p. 93f.]:

$$\mathbb{P}\left[\mu_{\mathbf{y}} - |t_{\xi/2}| \cdot \frac{\sigma_{\mathbf{y}}}{\sqrt{n_s}} \leq \mu_{\mathbf{y}}^* \leq \mu_{\mathbf{y}} + |t_{\xi/2}| \cdot \frac{\sigma_{\mathbf{y}}}{\sqrt{n_s}}\right] \approx 1 - \xi \tag{2.38}$$

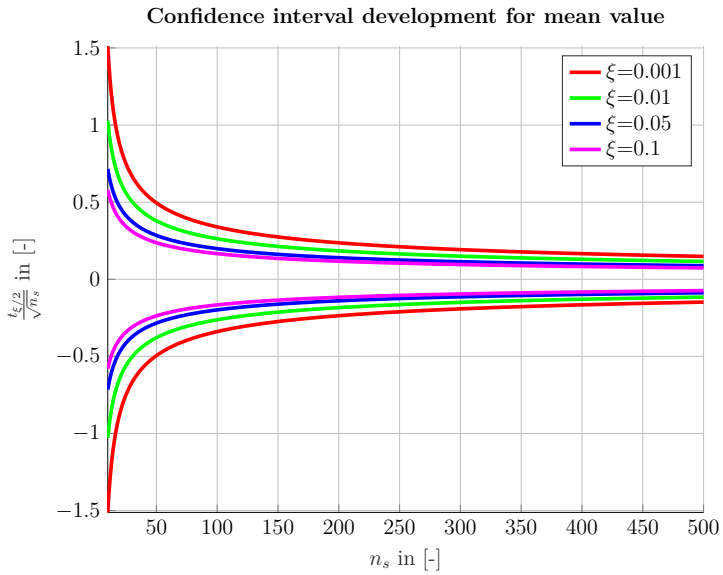

Figure 2.6: *Confidence interval development (i.e., spread factor) for lower and upper bound of mean value over number of samples and for different confidence levels.*

Equation (2.38) is exact for GAUSSIAN distributed RVs and generally a good approximation for a non-GAUSSIAN distributed RV due to the CLT (Section B.7). Take into account that μ^* is the unknown true mean value, while μ and σ are the sampled values for mean and standard deviation respectively (e.g., from (2.35) and (2.36)). The STUDENT'S T-DISTRIBUTION is denoted by t [2, p. 948f.].

The general interpretation of (2.38) is the following: The CI describes the probability ("confidence level") that the exact mean value is in the interval described by the sampled mean value and standard deviation. The spread of this interval is depending on the STUDENT'S T-DISTRIBUTION and the desired confidence level. It should be noted that, as the STUDENT'S T-DISTRIBUTION is depending on the number of samples n_s, the interval in (2.38) tightens with an increasing number of samples. It is further important to note that (2.38) also covers the case of an unknown standard deviation, i.e., only approximated by (2.36), which is normally present [125, p. 93].

To get a better overview on the CI definition in (2.38), Figure 2.6 shows the development of the CI bounds with changing number of samples and for different confidence levels. Here, the y-axis denotes the multiplication factor of the standard deviation in (2.38) including the sign. It is clear from Figure 2.6 that the CI is symmetric around zero and tightens with the number samples. This tightening is initially fast, but starting from approximately 200 samples reduces to the familiar convergence with the square root of the samples, which is known from the CLT. Generally, it can also be seen in Figure 2.6 that the CI increases with a higher accuracy requirement on the confidence level.

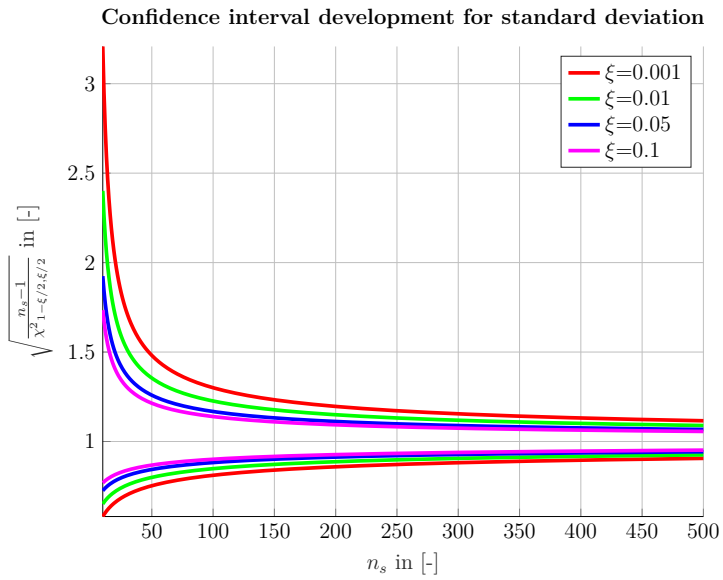

Figure 2.7: *Confidence interval development (i.e., spread factor) for lower and upper bound of standard deviation over number of samples and for different confidence levels.*

Equation (2.38) can be similarly extended to the standard deviation. A CI definition, with the same interpretation as for the mean value, i.e., that the real standard deviation is in the created CI with the probability of the confidence level, is then given as follows [119, p. 166ff.]:

$$\mathbb{P}\left[\sqrt{\frac{n_s-1}{{\chi^2}_{1-\xi/2}}}\sigma_{\mathbf{y}} \leq \sigma^*_{\mathbf{y}} \leq \sqrt{\frac{n_s-1}{{\chi^2}_{\xi/2}}}\sigma_{\mathbf{y}}\right] \approx 1-\xi \tag{2.39}$$

In (2.39), σ^* is the exact, unknown standard deviation, while χ^2 represents the χ^2-distribution [2, p. 940ff.]. Take into account that the χ^2-distribution depends on the number of samples, like the Student's t-distribution, and thus, a tighter CI is again achieved by increasing the number of samples. Once again, (2.39) is exact for Gaussian distributed RVs and sufficiently accurate, due to the CLT, for non-Gaussian distributed RVs [119, p. 167f.].

Similar to Figure 2.6, Figure 2.7 depicts the development of the CI for the standard deviation over the number of samples and for different confidence levels. A major difference to the CI for the mean value is that the CI for the standard deviation is non-symmetric and converges to 1 instead of 0 (as it is also natural from (2.39)). Still, similarities like the initial fast converges and afterward, the slow convergence with approximately the square root of the samples, are also seen for the standard deviation CI.

Both, (2.38) and (2.39) can be used in the following to get an idea of the approximation quality and accuracy of e.g., the gPC methodology introduced in Section 2.3, when comparing the results to MCA or LHS.

2.2.4 Subset Simulation

This final subsection of the classic uncertainty quantification methods introduces the subset simulation (SubSim) method that is often used in reliability engineering to calculate the probabilities of rare-events, i.e., events that only have a small probability of occurring (e.g., 10^{-6} or smaller) [7–9, 81]. These rare-events are generally encountered in risk and failure analysis, while this thesis considers rare-events in the context of the latter. In general, rare-events can no longer be treated by a standard MCA (Subsection 2.2.1) or LHS (Subsection 2.2.2) as a very large amount of samples is required even to just capture the failure event once (let alone sufficiently often for a statistical analysis) [9, p. 38ff.]. Thus, instead of directly trying to sample the rare-event, SubSim works on a conditional probability level, which is introduced in the following.

2.2.4.1 Basic Idea

The basic idea of SubSim is the definition and expansion of the failure probability as a series of conditional failure probabilities as follows [5, p. 1277ff.], [7]:

$$\mathbb{P}\left[\mathcal{F}\right] = \mathbb{P}\left[\mathcal{F}_m\right] = \mathbb{P}\left[\bigcap_{k=0}^{n_{ss}} \mathcal{F}_i\right] = \mathbb{P}\left[\mathcal{F}_0\right] \cdot \prod_{k=1}^{n_{ss}} \mathbb{P}\left[\mathcal{F}_k | \mathcal{F}_{k-1}\right] \tag{2.40}$$

In (2.40), $\mathcal{F} = \mathcal{F}_{n_{ss}}$ is the desired rare-event failure domain (with n_{ss} being the number of levels required in the SubSim to reach that event) and $\mathcal{F}_0 \supset \mathcal{F}_1 \supset \ldots \supset \mathcal{F}_{n_{ss}} = \mathcal{F}$ is a sequence of failure event domains with decreasing failure probability (i.e., the failure becomes less likely), i.e., there are $n_{ss} - 1$ intermediate failure events required to capture the rare-event. It is important to see that these failure events contain each other and converge to the desired rare-event. The conditional probability of failure events is denoted by $\mathbb{P}\left[\mathcal{F}_k | \mathcal{F}_{k-1}\right]$ and is the probability that the failure event $\mathcal{F}_k$ occurs, assuming that the failure event $\mathcal{F}_{k-1}$ has already occurred.

Thus, instead of evaluating the rare-event $\mathbb{P}\left[\mathcal{F}\right] = \mathbb{P}\left[\mathcal{F}_{n_{ss}}\right]$ directly by e.g., increasing the number of samples to obtain more samples in the failure domain, one must only evaluate a fairly frequent event $\mathbb{P}\left[\mathcal{F}_0\right]$ by e.g., MCA or LHS. Afterward, one can continue to evaluate the conditional probabilities $\mathbb{P}\left[\mathcal{F}_k | \mathcal{F}_{k-1}\right]$ regions that are more likely to occur and can be efficiently simulated by, again, MCA or LHS.

Consequently, the SubSim algorithm tries to generate samples from the RV PDF in a way that the failure domain is accurately explored. To this end, it chooses samples iteratively such that these best explore the failure domain by following a series of intermediate failure events. These intermediate failure events are described by the conditional probability. Following these intermediate failure events, the rare-event failure probability is given by (2.40). Take into account that this procedure based on intermediate failure events is generally favorable for risk or failure analysis although being computationally demanding: This is due to the fact that the result of the SubSim is then not only the rare-event

failure probability, but also the failure probabilities for each intermediate event. Thus, it is possible to get a good overview on the system characteristics and how the failure probability evolves [81, p. 29].

Summarizing, the important aspect of the SubSim is the calculation of the conditional failure samples that explore the conditional failure domains. This is achieved by the MCMC algorithm introduced in the following.

2.2.4.2 Markov chain Monte Carlo

Generally, the evaluation of the conditional probabilities is the major task within SubSim and achieved using the Markov chain Monte Carlo (MCMC) approach. The MCMC approach is normally applied when samples cannot be created, or can only be created with major effort, from (unknown) PDFs. A more detailed introduction to MCMC can e.g., be found in [81, p. 31ff.].

Generally, there exist multiple algorithms to create Markov chains, e.g., Metropolis-Hastings algorithm (MHA) [9, p. 122ff.], modified Metropolis-Hastings algorithm (MMHA) [9, p. 152ff.], or INFINITY SAMPLING [81, p. 37ff.]. In SubSim, the MMHA is often applied [148]: The algorithm creates new samples for the RVs in the random vector (i.e., a vector of one-dimensional random variables) (RVec) based on a local sampling around the current RV and accepts the new RVec if the result lies in the failure domain. Thus, a fast convergence of the MCMC algorithm by using the MMHA is assured. Additionally, the MMHA works component-wise on the RVec and thus, does not suffer from the curse of dimensionality, i.e., small acceptance rates of new samples with a large number of uncertain parameters [9, p. 151f.]. A general requirement of this component-wise sampling is the independence of the RVs (Appendix B.5), which can e.g., be assured by a KARHUNEN-LOÈVE transformation [68]. As the MMHA is a frequent choice in SubSim applications [7–9, 81], it is also applied in this thesis and thus, the next paragraphs introduce the algorithm.

The component-wise MMHA for the RVec, is given in Algorithm 2.2 [7, 9, 81, 97]: Here, at first a (symmetric) proposal PDF $\rho^*_{\Theta,i}\left(\tilde{\theta}|\theta_i\right)$ for each element of the RVec is defined in Step 0. In this context, the notation means that the new sample $\tilde{\theta}$ is created based on the current sample θ_i, i.e., the proximity of the current sample is explored (Step 2 in Algorithm 2.2). Based on this new sample an acceptance ratio r (Step 3 in Algorithm 2.2) is calculated as follows [9, p. 214ff.] or [81, p. 47f.]:

$$r = \frac{\rho_{\Theta,t}\left(\tilde{\theta}\right)}{\rho_{\Theta,t}\left(\theta_i\right)} \cdot \frac{\rho^*_{\Theta,i}\left(\tilde{\theta}|\theta_i\right)}{\rho^*_{\Theta,i}\left(\theta_i|\tilde{\theta}\right)} \overset{\text{sym.}}{=} \frac{\rho_{\Theta,t}\left(\tilde{\theta}\right)}{\rho_{\Theta,t}\left(\theta_i\right)} \tag{2.41}$$

Here, $\rho_{\Theta,t}(\cdot)$ is the desired/target PDF, which is the stationary/equilibrium PDF of the Markov chain [81, p. 32ff. & 37]. Take into account that the proposal PDF cancels when it is chosen symmetric ("sym.").

It can be stated for (2.41) that one should choose a target PDF similar to the real one, if information about it is available. If no information is available, the target PDF can be chosen as the standard GAUSSIAN, i.e., $\rho_{\Theta,t} \sim \mathcal{N}(0,1)$. This is an assumption without loss of generality for uncorrelated samples [8, p. 67] and can be achieved by proper probability integral transformations for non-GAUSSIAN RVs [60, p. 4] (e.g., a general PDF can be transformed to the standard GAUSSIAN by using the cumulative and inverse cumulative distribution function; see (B.2)). If one must create correlated samples, the work in [60] can be considered. This is not required in this thesis.

Based on the acceptance ratio in (2.41), the proposed sample vector is created in Step 7 of Algorithm 2.2. Following, this proposed sample vector is accepted as the new sample vector in Step 14, if it yields a system response in the failure domain.

Normally, the MCMC algorithm based on the MMHA introduced in Algorithm 2.2 is fast converging, as especially the sampling of new candidates is done locally around the current sample value (Step 2 in Algorithm 2.2). Thus, the acceptance rate (Step 7 in Algorithm 2.2) is normally very high and progress is made fast. It should be noted that the acceptance rate can even be improved by using e.g., INFINITY SAMPLING [81, p. 50ff.]. Using this approach, all candidate samples $\tilde{\theta}$ are accepted in Step 7 of Algorithm 2.2, but a specific proposal PDF must be applied [81, p. 50ff.].

Take into account that the sampling, acceptance, and exploration is crucial for the convergence of the algorithm: If the spread of the chosen proposal PDF (defined in Step 0 of Algorithm 2.2) is too small, i.e., only samples very close to the current sample are created, the acceptance rate is high, but the spatial dependence of the samples is high as well [9, p. 123]. This can ultimately lead to bias when using the samples for statistical estimations [9, p. 123]. On the other hand, a too large spread is also not desired, as then the acceptance ratio becomes too small and the next sample is more likely to be the current sample, which is again increasing their spatial dependence [9, p. 123]. Thus, a balance between exploration, i.e., spreading, and acceptance must be found [9, p. 123]. Generally, study [148] suggest to have an acceptance ratio of 30% to 50% in order to achieve a good variance reduction of the MCMC algorithm. This can e.g., be achieved by adapting the spread of the proposal PDF adaptively [81, p. 50].

Thus, the issue of the MMHA in Algorithm 2.2 is the choice of an appropriate proposal PDF: Generally, a UNIFORM PDF or a GAUSSIAN PDF can be chosen, to have a simple evaluation, the symmetric property, and a good overview of the PDF spread [9, p. 124] (Algorithm 2.2). In general, study [9, p. 123] suggests to use a spread similar to that of the target PDF, which is given, for uncorrelated samples, by the standard GAUSSIAN [8] (achieved by appropriate transformations if necessary [60, p. 4]). This standard spread can be easily reproduced by using a GAUSSIAN or UNIFORM proposal PDF.

Algorithm 2.2 Modified Metropolis-Hastings algorithm used in subset simulation for each component of the random vector to create new sample for random parameter vector $\boldsymbol{\theta}$ (after [9, p.152], [81, p. 48ff.] and [7, 60, 102]).

Require:

Current sample vector for the random parameters: $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \dots & \theta_i & \dots & \theta_{n_\theta} \end{bmatrix}^\mathsf{T}$.

Symmetric proposal PDF, $\rho^*_{\Theta,i}\left(\tilde{\theta}|\theta_i\right) = \rho^*_{\Theta,i}\left(\theta_i|\tilde{\theta}\right)$, for each element θ_i of the RVec $\boldsymbol{\theta}$.

1: **for** $i = 1, \dots, n_\theta$ **do**
2: Generate candidate sample $\tilde{\theta} \sim \rho^*_{\Theta,i}\left(\tilde{\theta}|\theta_i\right)$
3: Calculate the ratio between the proposed and the candidate sample evaluated at the target PDF, i.e., $r = \frac{\rho_{\Theta,t}\left(\tilde{\theta}\right)}{\rho_{\Theta,t}(\theta_i)}$
4: Set the acceptance ratio of the candidate sample $\tilde{\theta}$ as follows: $a = \min\left[1, r\right]$
5: Draw a random sample from the standard UNIFORM PDF $\mathcal{U}$ (lower bound: $a = 0$; upper bound: $b = 1$) as follows: $s \sim \mathcal{U}\left(a = 0, b = 1\right)$.
6: Create the vector of proposal samples $\boldsymbol{\theta}_p$ based on the following condition:
7: **if** $s \geq a$ **then**
8: Set proposed sample: $\theta_{p,i} = \tilde{\theta}$
9: **else**
10: Set proposed sample: $\theta_{p,i} = \theta_i$
11: **end if**
12: **end for**
13: Update the new sample vector by the rule:
14: **if** $\mathbf{y}\left(\mathbf{z}; \boldsymbol{\theta}_p\right) \in \mathcal{F}_i$ **then**
15: Set new sample vector: $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_p$
16: **else**
17: Set new sample vector: $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}$
18: **end if**

19: **return** New sample vector for the random parameters: $\boldsymbol{\theta}_{\text{new}}$.

Finally, it can be stated that the introduced MCMC algorithm converges to the stationary PDF under the assumption of ergodicity, i.e., that the whole parameter space, and thus, all failure regions, are explored [9, p. 124f.]. This is normally achieved by choosing suitable exploration parameters in the MMHA.

The described MCMC algorithm is now part of the SubSim strategy introduced in the following. Here, the major remaining point is the definition of the conditional failure domains as the MCMC algorithm takes care of the sample creation.

2.2.4.3 Basic Procedure of Subset Simulation

Applying the MMHA in Algorithm 2.2, the general SubSim method can be stated as given in Algorithm 2.3 [7, 9, 81]: Here, the SubSim starts with a basic MCA or LHS (Step 3) and afterward, subsequently explores the failure region by the MMHA (Step 8), yielding the chain of conditional probabilities. Hereby, the algorithm follows a pre-defined number of Markov chains n_c and the number of samples n_{sc} within those Markov chains in each SubSim simulation level (Step 8 in Algorithm 2.3). It should be noted that the split in different chains is an important feature of the SubSim method to ensure the ergodicity, and thus, convergence, of the MCMC algorithm [9, p. 161f.]. Furthermore, the fact that the seeds $\{\boldsymbol{\theta}_{k-1,\text{seed}}^{(j)} : j = 1, \ldots, n_c\}$, used for generating the conditional samples in the MCMC algorithm (Step 9 in Algorithm 2.3), are discarded after use provides confidence in the failure probability estimation and reduces the correlation between samples [9, p. 162]. Take into account that the seeds are chosen as those samples, which are closest to the failure domain (Step 5 in Algorithm 2.3).

It should be noted that there are essentially two options on how to conduct the SubSim in real applications: The first option is to choose the intermediate failure events in (2.40) before making the SubSim and create samples within the resulting conditional failure

Algorithm 2.3 Basic algorithm used for a subset simulation (after [9, p. 158ff.]).

Require:

Number of samples per level n_s.
Conditional probability p_0.
Critical threshold b.
Maximal MCMC level $n_{ss,\max}$.
Counter variable: $k = 1$

1: Calculate the number of Markov chains $n_c = p_0 \cdot n_s$ and the number of samples $n_{sc} = {p_0}^{-1}$ for each of the chains.
2: Initialize the SubSim by creating the random sample set $\{\boldsymbol{\theta}_0^{(i)} : i = 1, \ldots, n_s\}$.
3: Calculate the output set $\{\mathbf{y}_0^{(i)}\left(\mathbf{z}; \boldsymbol{\theta}_0^{(i)}\right) : i = 1, \ldots, n_s\}$ related to $\{\boldsymbol{\theta}_0^{(i)} : i = 1, \ldots, n_s\}$.
4: Normalize the output set $\{\mathbf{y}_0^{(i)}\left(\mathbf{z}; \boldsymbol{\theta}_0^{(i)}\right) : i = 1, \ldots, n_s\}$ such that larger values indicate samples closer to or in the failure domain.
5: Sort $\{\mathbf{y}_0^{(i)}\left(\mathbf{z}; \boldsymbol{\theta}_0^{(i)}\right) : i = 1, \ldots, n_s\}$ in ascending order to create $\{\mathbf{b}_0^{(i)} : i = 1, \ldots, n_s\}$. Here, $\mathbf{b}_0^{(i)}$ is an estimate of the exceedance probability $\mathbb{P}\left[\mathbf{y}\left(\mathbf{z}; \boldsymbol{\theta}\right) > b\right] = \frac{n_s - i}{n_s}, i = 1 \ldots, n_s$.
6: Set $b_0 = \mathbf{b}_0^{(n_s - n_c)}$ and $\{\boldsymbol{\theta}_{0,\text{seed}}^{(j)} : j = 1, \ldots, n_c\}$ corresponding to $\{\mathbf{b}_0^{(n_s - n_c + j)} : j = 1, \ldots, n_c\}$ as the threshold and the seeds for the next level respectively
7: Estimate the current failure probability using the indicator function in (2.44) by: ${p_f}^{(0)} = \frac{1}{n_s} \sum_{i=1}^{n_s} \widetilde{\mathcal{I}}\left[\mathbf{y}_0^{(i)}\left(\mathbf{z}; \boldsymbol{\theta}_0^{(i)}\right)\right]$

Algorithm 2.3 Basic algorithm used for a subset simulation (after [9, p. 158ff.]) (continued).

8: **while** $k \leq n_{ss,\max}$ & $\sum_{i=1}^{n_s} \widetilde{\mathcal{I}}\left(\mathbf{y}_{k-1}^{(i)}\left(\mathbf{z};\boldsymbol{\theta}_{k-1}^{(i)}\right)\right) < p_0 n_s$ **do**

9: Use e.g., the MMHA (Algorithm 2.2) to generate the samples $\{\boldsymbol{\theta}_k^{(j)(i)} : i = 1,\ldots,n_{sc}\}$ of the conditional PDF $\rho_j^*(.|\mathcal{F}_{k-1})$ for each seed $\{\boldsymbol{\theta}_{k-1,\text{seed}}^{(j)} : j = 1,\ldots,n_c\}$. This creates n_c Markov chains with n_{sc} samples.

10: Calculate the output set $\{\mathbf{y}_k^{(j)(i)}\left(\mathbf{z};\boldsymbol{\theta}_k^{(j)(i)}\right) : j = 1,\ldots,n_c,\ i = 1,\ldots,n_{sc}\}$ related to $\{\boldsymbol{\theta}_k^{(j)(i)} : j = 1,\ldots,n_c,\ i = 1,\ldots,n_{sc}\}$.

11: Normalize the output set $\{\mathbf{y}_k^{(j)(i)}\left(\mathbf{z};\boldsymbol{\theta}_k^{(j)(i)}\right) : j = 1,\ldots,n_c,\ i = 1,\ldots,n_{sc}\}$ such that larger values indicate samples closer to or in the failure domain.

12: Sort the normalized set $\{\mathbf{y}_k^{(j)(i)}\left(\mathbf{z};\boldsymbol{\theta}_k^{(j)(i)}\right) : j = 1,\ldots,n_c,\ i = 1,\ldots,n_{sc}\}$ in ascending order to create $\{\mathbf{b}_k^{(i)} : i = 1,\ldots,n_s\}$. Here, $\{\mathbf{b}_k^{(i)}\}$ is an estimate of the exceedance probability $\mathbb{P}[\mathbf{y}(\mathbf{z};\boldsymbol{\theta}) > b] = {p_0}^k \frac{n_s - i}{n_s}, k = i,\ldots,n_s$.

13: Set $b_k = \mathbf{b}_k^{(n_s - n_c)}$ and $\{\boldsymbol{\theta}_{k+1,\text{seed}}^{(j)} : j = 1,\ldots,n_c\}$ corresponding to the set $\{\mathbf{b}_k^{(n_s - n_c + j)} : j = 1,\ldots,n_c\}$ as the threshold and the seeds for the next level.

14: Estimate the current failure probability using the indicator function in (2.44) and the failure probability estimation in (2.43).

15: Increase level counter $k + 1$.

16: **end while**

17: Set the number of SubSim level: $n_{ss} = k - 1$.

18: Calculate the failure probability p_f based on (2.43) or (2.47).

19: Calculate the coefficient of variation c_V based on (2.45).

20: **return** Failure probability p_f and coefficient of variation (CoV) c_V. Number of SubSim levels n_{ss}.

domains. As this procedure, and especially the sampling in the conditional failure domain, is not straightforward [7], the opposite procedure is generally applied (this procedure is also already introduced in Algorithm 2.3): Instead of defining the intermediate failure events in advance, they are chosen during the SubSim evaluation, while the conditional probabilities are set to a fixed, pre-defined value (i.e., $p_0 \in]0,1[$). Therefore, the system responses are appropriately normalized, such that larger values indicate output responses closer to the failure domain, and sorted in ascending order (Step 12 in Algorithm 2.3). Then, the $(n_s - n_c)-$th value is the intermediate failure event threshold b_k for the next SubSim level (i.e., it defines the intermediate failure event). The last n_c values in the sorted normalized vector provide the corresponding seeds (samples) for the next SubSim level (Step 13 in Algorithm 2.3).

This choice gives exactly the p_0 percentile of the samples that define the next conditional failure domain (i.e., those samples lie in the conditional failure domain). Thus, the failure domains and conditional samples are adaptively chosen to follow the conditional probability chain of the SubSim in (2.40). A fairly often used, and normally very efficient conditional probability value, is $p_0 = 0.1$ [7]. This value comes from a theoretical minimization of the CoV by choosing an appropriate SubSim level probability [9, p. 175ff.]. The CoV here is a measure of the confidence in the results, where a smaller value indicates a higher confidence (see (2.45)). Further research in study [148] also shows that choosing $0.1 \leq p_0 \leq 0.3$ results in similar efficiency and thus, specific fine-tuning of the conditional probability level is not necessary in most cases.

The introduced procedure based on choosing the intermediate failure events adaptively, yields the following conditional probability estimation (based on (2.40)) [9, p. 163]:

$$\begin{aligned} \mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) > b_0\right] &\approx \mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) > b_1 | \mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) > b_0\right] \\ &\approx \ldots \approx \mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) > b_{n_{ss}} | \mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) > b_{n_{ss}-1}\right] \approx p_0 \end{aligned} \tag{2.42}$$

Here, b_k, $k = 0, \ldots, n_{ss}$ are the calculated SubSim thresholds (Step 13 in Algorithm 2.3). As seen, this procedure makes the SubSim probability calculation in (2.40) straightforward to evaluate as the conditional failure probabilities are equal to the defined threshold p_0. Thus, it is straightforward to calculate the failure probability of the rare-event, which is introduced in the next subsection.

2.2.4.4 Statistical Analysis

Generally, the failure probability of the SubSim with respect to the desired $b$and all intermediate thresholds b_k can be estimated by using the threshold and system response sets from Steps 12 and 13 in Algorithm 2.3 as follows [9, p. 179]:

$$\begin{aligned} {p_f}^{(0)} &= \mathbb{P}\left[\mathcal{F}_0\right] = \frac{1}{n_s}\sum_{i=1}^{n_s} \widetilde{\mathcal{I}}\left[\mathbf{y}_0^{(i)}\left(\mathbf{z};\boldsymbol{\theta}_0^{(i)}\right)\right], \quad b < b_0 \\ {p_f}^{(k=1,\ldots,n_{ss}-1)} &= \mathbb{P}\left[\mathcal{F}_{k=1,\ldots,n_{ss}-1}\right] = \frac{1}{n_s} {p_0}^k \sum_{j=1}^{n_c}\sum_{i=1}^{n_{sc}} \widetilde{\mathcal{I}}\left[\mathbf{y}_k^{(j)(i)}\left(\mathbf{z};\boldsymbol{\theta}_k^{(j)(i)}\right)\right], \quad b_{k-1} < b < b_k \\ p_f &= \mathbb{P}\left[\mathcal{F}\right] = \frac{1}{n_s} {p_0}^{n_{ss}} \sum_{i=1}^{n_c}\sum_{j=1}^{n_{sc}} \widetilde{\mathcal{I}}\left[\mathbf{y}_{n_{ss}}^{(j)(i)}\left(\mathbf{z};\boldsymbol{\theta}_{n_{ss}}^{(i)(j)}\right)\right], \quad b > b_{n_{ss}} \end{aligned} \tag{2.43}$$

In (2.43), $\mathbf{y}_0^{(i)}$ is the system responses of the random samples $\boldsymbol{\theta}_0^{(i)}$ given by the MCA solution (i.e., the zeroth subset), while $\mathbf{y}_k^{(j)(i)}$ is the system response of the j-th MCMC element with the i-th random sample of this chain element within the k-th SubSim run (Algorithm 2.3). Furthermore, n_{ss} is the number of SubSim levels required for the SubSim to converge (Step 17 in Algorithm 2.3). Additionally, $\widetilde{\mathcal{I}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]$ is the indicator function, which indicates whether or not the system response is in the failure domain (i.e., it makes a logic decision), for the SubSim defined as:

$$\widetilde{\mathcal{I}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \begin{cases} 1, & \text{for } \mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \in \mathcal{F} \\ 0, & \text{else} \end{cases} \tag{2.44}$$

It should be noted that (2.43) and (2.44) give a failure probability estimate that can be regarded as one realization of the real rare-event probability. This is due to the fact that there is only limited information available to calculate the value (e.g., a limited number of samples) and thus, the probability will not resemble the real value exactly [148, p. 292]. For instance, changing the evaluation samples will lead to a (slightly) different failure estimate (due to the CLT and the associated CIs). Thus, the failure probability can be treated as a RV itself and its relative plausibility and accuracy can be assessed by statistical analysis [148, p. 292].

As already mentioned: To assess this accuracy of the SubSim, the CoV, which connects the mean and the standard deviation of the failure probability, can be used [9, p. 25f.]. This is generally necessary because only looking at the standard deviation is no longer informative when considering small failure probabilities because those must be seen in relation to each other. Thus, the CoV is defined as follows:

$$c_V\left[p_f\right] = \frac{\sigma\left[p_f\right]}{\mathbb{E}\left[p_f\right]} \tag{2.45}$$

In general, the goal is to have a small CoV as then the dispersion of the data is small and the certainty to fulfill the rare-event probability is high. Thus, a value of $c_V \ll 1$ should be achieved by the SubSim. Indeed it can be shown that the CoV can never be larger than one, if there is at least one sample within the failure domain [81, p. 17]. In this context in can be stated that the SubSim result obtained by Algorithm 2.3 is to be taken with caution if the CoV is greater than one.

To calculate this CoV, an estimate of the standard deviation of the failure probability is required as the mean value is known from (2.43). Therefore, the method introduced in [148] can be used: Here, the posterior/marginal PDF of the SubSim is estimated by a Beta PDF, which is proven to coincide with the first two moments of the exact failure RV (i.e., it returns the exact mean and standard deviation). In other words, the PDF of the failure probability is estimated by a Beta PDF. By this, an estimate of the mean failure probability is available, which should coincide with the probability calculated in (2.43), and an estimate of its standard deviation, which can be used to calculate the CoV. The shape parameter α and rate parameter β of the Beta PDF, $\mathcal{B}\left(a=0, b=1, \alpha, \beta\right)$, are defined as follows [148, p. 293]:

$$
\begin{aligned}
\alpha &= c_1 \cdot \frac{1-c_2}{c_2-c_1} \\
\beta &= (1-c_2) \cdot \frac{1-c_2}{c_2-c_1} \\
\text{with} \quad c_1 &= (p_0 \cdot n_s + 1)^{n_{ss}} \cdot \frac{\sum_{i=1}^{n_s} \widetilde{\mathcal{I}}\left[\mathbf{y}_{n_{ss}}^{(i)}\left(\mathbf{z}; \boldsymbol{\theta}_{n_{ss}}^{(i)}\right)\right] + 1}{(n_s+2)^{n_{ss}+1}} \\
\text{with} \quad c_2 &= (p_0 \cdot n_s + 2)^{n_{ss}} \cdot \frac{\sum_{i=1}^{n_s} \widetilde{\mathcal{I}}\left[\mathbf{y}_{n_{ss}}^{(i)}\left(\mathbf{z}; \boldsymbol{\theta}_{n_{ss}}^{(i)}\right)\right] + 2}{(n_s+3)^{n_{ss}+1}}
\end{aligned}
\tag{2.46}
$$

Again, n_{ss} is the number of SubSim levels that were required for the SubSim to converge. It should be taken into account that here the sum over all samples n_s is taken for simplicity as it is known that the MCMC, based on its definition in Step 1 of Algorithm 2.3, creates exactly this amount of samples.

Then, the mean and standard deviation of the BETA PDF are calculated, based on the definitions in (2.46), as follows [148, p. 293]:

$$
\begin{aligned}
\mathbb{E}\left[p_f\right] &= \frac{\alpha}{\alpha+\beta} \\
\sigma\left[p_f\right] &= \sqrt{\frac{\alpha \cdot \beta}{(\alpha+\beta)^2 \cdot (\alpha+\beta+1)}}
\end{aligned}
\tag{2.47}
$$

Thus, (2.47) provides the possibility to calculate and check the CoV after the SubSim (Algorithm 2.3) is finished using (2.45), in order to check the convergence and significance of the results. In addition, by comparing the resulting CoV from Algorithm 2.3 with e.g., literature values [97], the viability of calculated SubSim results can be assessed. In general, the magnitude of the CoV in SubSim has been proven to be influenced by the subset probability p_0 [9, p. 175ff.]. Thus, this value must be chosen with care and adapted, if necessary, to get a suitable CoV.

2.2.4.5 Illustrative Example

To show how the SubSim, and especially the MMHA sampling, works, the following illustrative example is discussed (similar to [102]): Here, the two uncertainties are distributed with a UNIFORM PDF on a two-dimensional grid in the interval $[-10; 10]$, i.e., $\theta_1 \sim \mathcal{U}(a=-10, b=10)$ and $\theta_2 \sim \mathcal{U}(a=-10, b=10)$. In the left lower corner of this rectangle, a quarter of a circle with radius $r = 1.0$ depict one failure region. A second failure region is given by a full circle at the position $\begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^\mathsf{T} = \begin{bmatrix}1 & 2\end{bmatrix}^\mathsf{T}$ with the same radius of $r = 1.0$. This example shows the capabilities of the SubSim with MMHA to distribute samples to different failure domain of different size, as well as to unconnected failure domains. Thus, it provides a viable example to understand the working principle of the algorithms.

It is possible to analytically evaluate the probability to be within the failure regions as: $p_f{}^* = \mathbb{P}[\mathcal{F}] = \frac{5}{4}\frac{\pi \cdot r^2}{20^2} \approx 0.982\%$ (which is not a rare-event but used here for the purpose of demonstration and the possibility of visualization). In the following, two conditional probability values are used: $p_0 = 0.01$ with $n_c = 50$ Markov chains and $n_{sc} = 100$ elements in each chain ($n_s = 5000$; Figure 2.9; "first case") and $p_0 = 0.125$ with $n_c = 125$ Markov chains and $n_{sc} = 8$ elements in each chain ($n_s = 1000$; Figure 2.8; "second case"). This depicts different SubSim parameter settings and thus, different convergence behaviors of the SubSim. The first case shows an example with a small number of Markov chains, but a large number of samples within those chains. The second case depicts a SubSim with a larger number of Markov chains and a smaller number of samples within these chains.

The results for the first case are shown in Figure 2.8, while Figure 2.9 gives an overview of the second case. Generally, green crosses mark samples in the allowed domain, while red crosses mark samples in the failure domain. The failure domains are depicted by dashed red lines. SubSim level 0 is the initial MCA solution, while all subsequent levels show the results when applying Algorithm 2.2 (with the MMHA) to choose new samples. Here, it should be noted that the blue circles around the failure domains depict the threshold values b_k, $k = 0, \ldots, n_{ss}$ for the different SubSim simulation levels. In this example, the normalized threshold is defined as the minimal distance to the closest circle centers, i.e., the center of the failure domain. This is exactly what is achieved by the normalizing, sorting, and seed choosing in Step 5 of Algorithm 2.3. It can be directly seen that each new SubSim level only creates new samples that are within this conditional failure domain as required by the SubSim.

Another property that can be observed directly in Figures 2.8 and 2.9 is that the first example (with $p_0 = 0.01$) converges with one less level. Overall, both examples distribute the samples to the failure domain fairly good and it can be seen that all SubSim levels fulfill the requirement that the generated samples are within the conditional failure set, i.e., the thresholds. Additionally, more and more samples are created in the actual failure domain, which is achieved using the MMHA in Algorithm 2.2.

Then, applying the method in Subsection 2.2.4.4, the following failure probabilities and CoVs can be estimated. The first case ($p_0 = 0.01$) results in:

$$p_f = \mathbb{P}[\mathcal{F}] \approx 1.017\%, \qquad c_V[p_f] \approx 13.93\% \tag{2.48}$$

The second case ($p_0 = 0.125$) yields:

$$p_f = \mathbb{P}[\mathcal{F}] \approx 0.911\%, \qquad c_V[p_f] \approx 12.11\% \tag{2.49}$$

It is evident that both examples (see (2.48) and (2.49)) give a good, although not perfect, approximation of the failure probability. This is to be expected as the SubSim algorithm was not specifically tuned for the "frequent" failure probability looked at in this example.

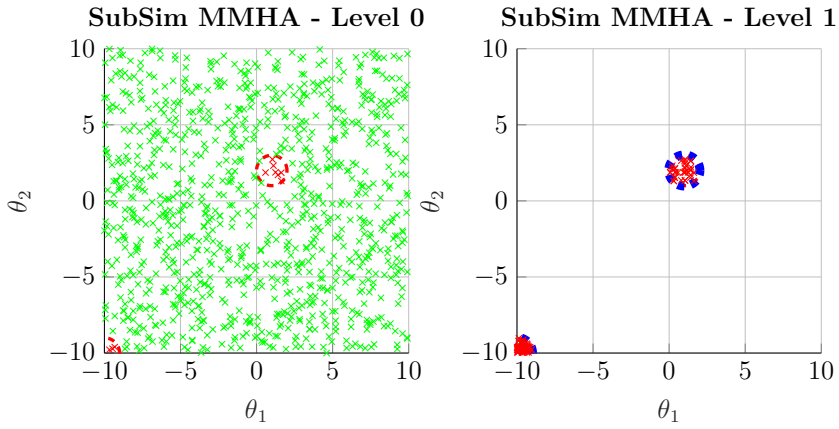

Figure 2.8: *General behavior of subset simulation with* $p_0 = 0.01$ *and movement of the samples* ($n_s = 1000$*; red: in failure domain, green: in allowed domain) over the different subset simulation runs (blue circles are conditional failure regions based on threshold) applying the modified Metropolis-Hastings algorithm in Algorithm 2.2.*

Overall, the first example has a larger CoV, while the second example has a worse approximation of the true failure probability. Thus, it is always necessary to try different conditional probability levels with different number of samples to check for the appropriate convergence and magnitude of the CoV. Additionally, (2.49) shows in application, why subset thresholds in between $p_0 = [0.1, \ldots, 0.3]$ have proven to be viable, especially to achieve a small CoV [9, p. 175ff.].

Although not a standard application of CIs, (2.38) can be used to estimate the CI for the first and second example (here with $\xi = 0.05$):

$$\begin{aligned} p_0 = 0.010: &\quad \mathbb{P}\left[0.739\% \leq {p_f}^* = 0.982\% \leq 1.294\%\right] \approx 95\% \\ p_0 = 0.125: &\quad \mathbb{P}\left[0.694\% \leq {p_f}^* = 0.982\% \leq 1.127\%\right] \approx 95\% \end{aligned} \tag{2.50}$$

Thus, a 95%-CI with both chosen probability levels p_0 is fulfilled. As it was already seen in (2.48) and (2.49), the bounds for $p_0 = 0.125$ are tighter due to the smaller CoV.

This basic SubSim method is extended in Chapter 6 to the needs of this thesis and the general gPC collocation framework (Chapter 5) with CCs. Here, the efficient solution of the SubSim within the OCP context is of paramount importance and achieved by the gPC method introduced next.

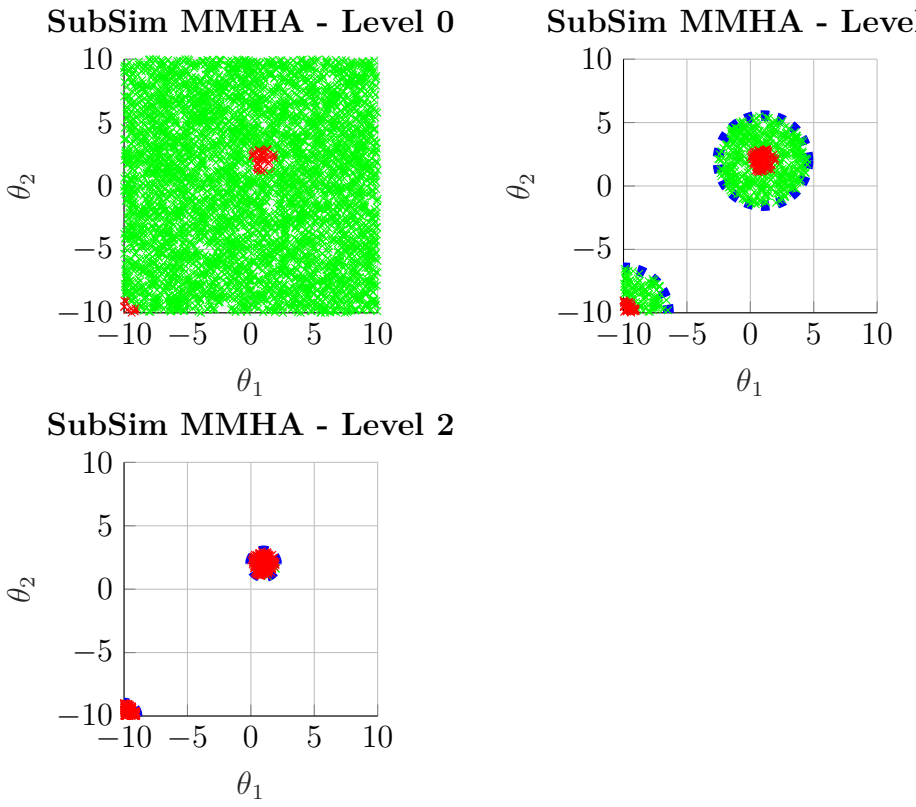

Figure 2.9: *General behavior of subset simulation with $p_0 = 0.125$ and movement of the samples ($n_s = 5000$; red: in failure domain, green: in allowed domain) over the different subset simulation runs (blue circles are conditional failure regions based on threshold) applying the modified Metropolis-Hastings algorithm in Algorithm 2.2.*

2.3 Generalized Polynomial Chaos

In contrast to the classic sampling methods for the uncertainty description, introduced in Section 2.2, generalized polynomial chaos (gPC) does rely on the approximation of the uncertain system response by analytic functions (in this case: orthogonal polynomials). Thus, gPC fits orthogonal polynomials to the actual response of the system and is therefore said to be a NON-SAMPLING METHOD. This non-sampling based procedure by a response surface function approximation is also visualized in Figure 2.10. Here, it can be seen that the gPC method transcribes the general (unknown) nonlinear model response ("output") with uncertainties into an equivalent finite sum expansion formula. One benefit of this new representation is that the influence of the uncertainties is decoupled from the deterministic parameters. This makes the stochastic analysis simple and straightforward.

The origin of the gPC methodology was introduced by WIENER in 1938 to model GAUSSIAN uncertainties [135]. In 2002, XIU AND KARNIADAKIS extended this original methodology to general orthogonal polynomials using the WIENER-ASKEY SCHEME, calling the method GENERALIZED POLYNOMIAL CHAOS [143]. This generalization allowed the use of not only the GAUSSIAN, but also further uncertainty PDFs. Some

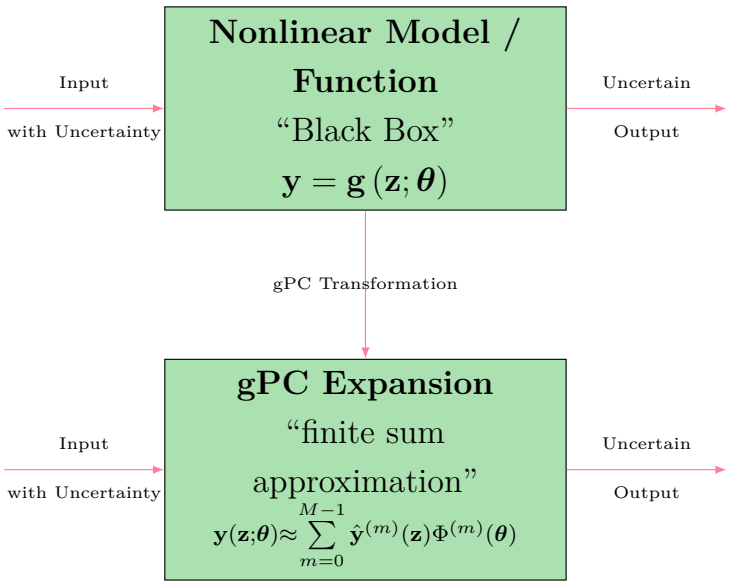

Figure 2.10: *General procedure of generalized polynomial chaos chaos transcription method for nonlinear model with multiple uncertainties resulting in a finite sum approximation.*

comprehensive reviews, which are the basis for the following description, have previously been conducted [36, 140, 141]. Additional information on required probability theory is given in Appendix B, while Appendix C deals with the basics of orthogonal polynomials, which are required in this thesis for the gPC expansion. The following subsections introduce the general gPC expansion procedure, starting with the gPC expansion for a scalar RV in Subsection 2.3.1.

2.3.1 Expansion with Scalar Random Variable

Within this subsection, the gPC expansion for a scalar RV Θ (Appendix B.1) with probability distribution function $F_\Theta(\theta) = \mathbb{P}(\Theta \leq \theta)$ (Definition B.3; this is the well-known cumulative distribution function, see (B.2)) is considered. Here, the scalar random parameter is denoted by θ. This is done to give an insight into the general procedure. An extension to multiple dimensions is straightforward and conducted in Subsection 2.3.2.

First of all, the RV must have finite second order, i.e., even, moment (Subsection B.2.2) for the gPC expansion to be applicable, meaning that the following condition must be fulfilled [141, p. 58]:

$$\mathbb{E}\left[|\Theta|^{2m}\right] = \int_\Omega |\theta|^{2m} \underbrace{\mathrm{d}F_\Theta(\theta)}_{=\rho_\Theta(\theta)\,\mathrm{d}\theta,\ \text{after (B.2)}} < \infty, \quad m \in \mathbb{N}_0 = \{0, 1, \ldots\} \tag{2.51}$$

Thus, the statistical moments of the random parameter must exist. This is generally applicable to all physical systems that are considered in this thesis and thus, gPC can be applied. It should be noted that Ω is the support of the uncertainty/random space (Appendix B), i.e., where it is defined.

If (2.51) is fulfilled, the d^{th}-order gPC expansion of the random solution $y(\mathbf{z};\theta) \in \mathbb{R}$ (sensitive parameters $\mathbf{q}$ are omitted for the sake of brevity) is given by the orthogonal projection operation $P^{(d)}$ of order d onto the polynomial space $W^{(d)}$ (Definition 2.2) as follows [141, p. 31]:

$$y(\mathbf{z};\theta) \approx P^{(d)}y = y^{(d)}(\mathbf{z};\theta) = \sum_{m=0}^{d-1} \hat{y}^{(m)}(\mathbf{z})\,\phi^{(m)}(\theta) \tag{2.52}$$

Take into account that (2.52) is basically a FOURIER decomposition of the original stochastic problem in an uncertain and a non-uncertain/deterministic part. Here, d is the degree of the polynomial (also called "expansion order" in the following) and $\hat{y}^{(m)} \in \mathbb{R}$ is an expansion coefficient (deterministic part) of order m (also called "FOURIER coefficient"). The orthogonal polynomials (uncertain part; also called "basis functions") are represented by $\phi^{(m)} \in \mathbb{R}$ and describe an orthogonal polynomial of order m (i.e., it has m as the largest polynomial exponent; Appendix C).

The polynomial space $W^{(d)}$ of these orthogonal polynomials is then defined by a so-called orthogonal polynomial measure: This measure is the orthogonality with respect to the probability measure, i.e., the PDF $\rho_\Theta(\theta)$, of the RV as follows (after [140, p. 251f.]):

Definition 2.2 (Continuous Orthogonal Space)**.** *A one–dimensional, continuous orthogonal polynomial space with respect to the measure (PDF) $\rho_\Theta(\theta)$ in Ω is defined by:*

$$W^{(d)} \equiv \left\{ v : \Omega \to \mathbb{R} : v \in \operatorname{span}\left\{ \phi^{(m)}(\theta),\, m = 0, \ldots, d-1 \right\} \right\}$$

Here, $\left\{ \phi^{(m)}(\theta), m = 0, \ldots, d-1 \right\}$ are a set of orthogonal polynomials that satisfy the orthogonality relation in the L^2-Hilbert space [5, p. 1073] defined by:

$$\mathbb{E}\left[\phi^{(m)}(\theta)\phi^{(n)}(\theta)\right] = \int_\Omega \phi^{(m)}(\theta)\phi^{(n)}(\theta)\rho_\Theta(\theta)\,\mathrm{d}\theta = \left[h^{(m)}\right]^2 \delta_{mn}, \quad m,n \in \mathbb{N}_0$$

The Kronecker delta function is denoted by $\delta_{mn} = \begin{cases} 1, & \text{if} \quad m = n \\ 0, & \text{else} \end{cases}$, *while*

$$\left[h^{(m)}\right]^2 = \int_\Omega \left[\phi^{(m)}(\theta)\right]^2 \rho_\Theta(\theta)\,\mathrm{d}\theta$$

is the normalization constant to get an orthonormal polynomial basis.

Within this work, only orthonormal polynomials are used, i.e., they are normalized using $\left[h^{(m)}\right]^2$ in the sense of Definition 2.2. Therefore, the terms orthogonal and orthonormal are synonymous throughout this thesis. Appendix C gives a more detailed insight into the properties of orthogonal polynomials and defines the relevant ones for this work. A summary is given in Table 2.2 with corresponding PDFs and supports. It should be

noted that Table 2.2 introduces the common relations of orthogonal polynomials and PDFs in the WIENER-ASKEY SCHEME. Extensions, using the theory of arbitrary polynomial chaos [98, 136] or GAUSSIAN mixture models [103, 129] are also available. These are not specifically treated in this thesis as the introduced PDFs in Table 2.2 are sufficient for the considered problems. Nonetheless, the developed methods can be directly extended using the aforementioned theories, as shown e.g., by the author in [103].

Table 2.2: *Continuous probability density function-orthogonal polynomial connection for standard generalized polynomial chaos and scalar random variable (after [140]).*

Probability Distribution	**Probability Density Function** $\rho_\Theta(\theta)$	**Support** Ω	**Symbol**	**Orthogonal Polynomial**
GAUSSIAN/NORMAL	$\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\theta^2}{2}\right)$	$]-\infty,\infty[$	$\mathcal{N}(\mu=0,\sigma=1)$	Hermite (Definition C.3)
GAMMA	$\frac{\theta^\alpha \exp(-\theta)}{\Gamma(\alpha+1)}$	$[0,\infty[$	$\gamma(\mu=\alpha,\sigma=\sqrt{\alpha},\alpha)$	Laguerre (Definition C.4)
BETA	$\frac{\Gamma(\alpha+\beta+2)}{2^{\alpha+\beta+1}\Gamma(\alpha+1)\Gamma(\beta+1)}(1-\theta)^\alpha(1+\theta)^\beta$	$[-1,1]$	$\mathcal{B}(a=-1,b=1,\alpha,\beta)$	Jacobi (Definition C.5)
UNIFORM	$\frac{1}{2}$	$[-1,1]$	$\mathcal{U}(a=-1,b=1)$	Legendre (Definition C.2)

A basic example of the idea behind the gPC expansion in (2.52) is given in Figure 2.11: Here, an uncertain parameter is influencing the output by an exponential function dependence (this function is shown in solid red). The parameter is distributed according to a standard UNIFORM PDF, i.e., $\theta \sim \mathcal{U}(a=-1,b=1)$, i.e., the lower bound of the PDF support is $a=-1$ and the upper bound is $b=1$. The gPC expansion in (2.52), which is fitted to the exact exponential functions, is shown for the orders $d=2,\ldots,5$. Here, the LEGENDRE polynomials (Appendix C), as suggested in Table 2.2, are applied as the orthogonal polynomials in (2.52). It can be seen that the gPC expansion yields a bad approximation for only a second order expansion (solid cyan), which is a first-order, linear fit, while the approximation accuracy of the system response by gPC is already significantly increased with a third order expansion (dashed green; quadratic fit). Starting from the fourth order expansion (dash-dotted blue; cubic fit), there is virtually no difference between the exact and the gPC expansion solution. Thus, the gPC expansion converges to the exact solution fast, as already noted before. This is especially the case if the function, which is to be approximated, is smooth. It should be noted here, without further details at this point, that the determination of the expansion coefficients, required in (2.52) to plot Figure 2.11, is based on the stochastic collocation (SC) approach introduced in Subsection 2.3.3.

Looking at a mathematical description, the convergence and accuracy of (2.52) can also be described by the CAMERON-MARTIN THEOREM [26] and the WEIERSTRASS THEOREM (Theorem C.6) as follows [141, p. 59]:

$$||y(\mathbf{z};\theta) - P^{(d)}y||_{L_d^2} \to 0, \quad d \to \infty \tag{2.53}$$

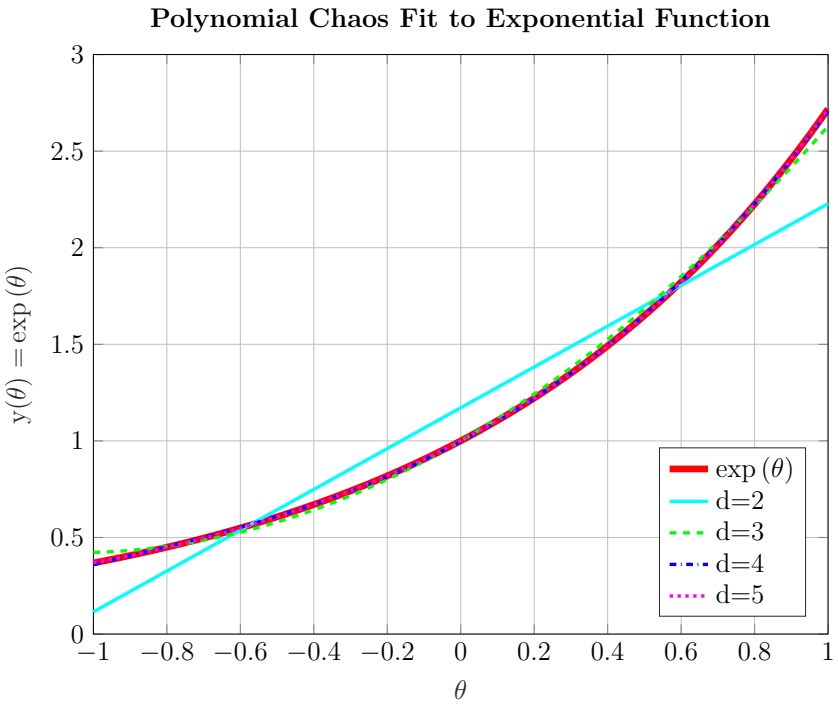

Figure 2.11: *Polynomial chaos approximation of exponential function by different generalized polynomial chaos expansion orders with Legendre polynomials.*

This convergence type is often referred to as "mean-square convergence" [141, p. 59] and states that the gPC approximation converges to the exact solution for a sufficiently large expansion order. As Figure 2.11 has shown, the necessary expansion order depends on the nonlinearity, the uncertainty is introducing in the system response.

For (2.53) to hold, the function must be square-integrable, i.e., its squared value must be integrable over the random space, which is a general requirement for the gPC method (see (2.51)). It is important to note in this context that the convergence rate of (2.52) depends on the smoothness of the function with respect to the uncertain parameters [141, p. 59f.]. Generally, it can be stated that a smoother function can be approximated by the gPC method with a lower expansion order. Take into account that this also directly implies a strategy on how to choose the expansion order, as the uncertain parameter influence on the system output is functionally known. Thus, a good "guess" on the smoothness and the nonlinearity of the response can be made and an appropriate expansion order d can be chosen. It should further be noted that the convergence in (2.53) implies that the gPC expansion converges in probability (Definition B.10) [141, p. 59], i.e., it is a consistent estimator that converges to the true function when using a sufficiently large expansion order.

Take into account that (2.52) can also be written as a matrix-vector operation:

$$\underbrace{y^{(d)}(\mathbf{z};\theta)}_{\in\mathbb{R}} \approx \underbrace{\left[\hat{y}^{(0)}(\mathbf{z}) \quad \dots \quad \hat{y}^{(d-1)}(\mathbf{z})\right]}_{\in\mathbb{R}^{1\times d}} \cdot \underbrace{\begin{bmatrix} \phi^{(0)}(\theta) \\ \vdots \\ \phi^{(d-1)}(\theta) \end{bmatrix}}_{\in\mathbb{R}^{d}} \tag{2.54}$$

Here, it becomes clear that there are d expansion coefficients that have d corresponding orthogonal polynomials to create the scalar random output response. This also shows that sampling the response surface is possible by a linear mapping ("matrix-vector operation"), which is later used in the CC-OC framework (Chapter 6).

As observed in Table 2.2, the orthogonal polynomials in (2.52) are defined by the chosen PDF for the RV. Thus, the remaining issue in (2.52) is the calculation of the expansion coefficients. Therefore, an error ε is defined as follows that should ultimately approach zero (i.e., the gPC expansion should approximate the actual response):

$$\varepsilon = y(\mathbf{z};\theta) - \sum_{m=0}^{d-1} \hat{y}^{(m)}(\mathbf{z})\,\phi^{(m)}(\theta) \to 0 \tag{2.55}$$

Looking at $\varepsilon = 0$, a GALERKIN PROJECTION [5, p. 1094] with the orthogonal polynomials and the PDF can be made. Therefore, (2.55) is multiplied by an n-th order orthogonal polynomial $\phi^{(n)}(\theta)$ ("shape function") and integrated over the probability space as follows:

$$\int_\Omega y(\mathbf{z};\theta)\,\phi^{(n)}(\theta)\,\rho_\Theta(\theta)\;\mathrm{d}\theta - \int_\Omega \sum_{m=0}^{d-1} \hat{y}^{(m)}(\mathbf{z})\,\phi^{(m)}(\theta)\,\phi^{(n)}(\theta)\,\rho_\Theta(\theta)\;\mathrm{d}\theta \equiv 0 \tag{2.56}$$

From Definition 2.2, it is known that the second integral in (2.56) is only non-zero if $m = n$ (orthogonality property in Definition 2.2). Therefore, only this case can be used to calculate the expansion coefficients and consequently, all n are set to m in (2.56). As the m-th orthogonal polynomial is now part of the first integrand as well, the sum operation can consequently be extended over both integrals, yielding:

$$\sum_{m=0}^{d-1}\left[\int_\Omega y(\mathbf{z};\theta)\,\phi^{(m)}(\theta)\,\rho_\Theta(\theta)\;\mathrm{d}\theta - \hat{y}^{(m)}(\mathbf{z})\underbrace{\int_\Omega \phi^{(m)}(\theta)\,\phi^{(m)}(\theta)\,\rho_\Theta(\theta)\;\mathrm{d}\theta}_{\left[h^{(m)}\right]^2,\ \text{after Definition 2.2}}\right] \equiv 0 \tag{2.57}$$

Additionally, it is used in (2.57) that the expansion coefficients are independent of the integration over the PDF support as they are specifically independent of the RVs. Therefore, they can be put in front of the integration.

Then, (2.57) can be solved for each expansion coefficients individually as the sum is generally only zero if all addends are zero. Furthermore, Definition 2.2 for the normalization constant can be applied resulting in [140, p. 260]:

$$\hat{y}^{(m)}(\mathbf{z}) = \frac{1}{\left[h^{(m)}\right]^2}\int_\Omega y(\mathbf{z};\theta)\,\phi^{(m)}(\theta)\,\rho_\Theta(\theta)\;\mathrm{d}\theta, \qquad m = 0,\dots,d-1 \tag{2.58}$$

It is clear that (2.58) cannot be solved directly as the expansion coefficients depend on the system response $y(\mathbf{z};\theta)$ (also: "response surface"), which is to be approximated by gPC. Thus, it is necessary to find a way to calculate/approximate the integral in (2.58) without requiring any knowledge on the response surface over its (complete) support. Within this thesis, the SC approach to calculate the integral in (2.58) is used. The procedure is introduced in more detail in Subsection 2.3.3, while at first the multivariate gPC expansion is introduced in Subsection 2.3.2.

2.3.2 Expansion with Multivariate Random Variable

Definition 2.2 provided a description of the orthogonal polynomial space for a scalar RV. This subsection gives an extension to multiple dimensions by using the tensor product of one-dimensional grids. Thus, it holds for the N-dimensional orthogonal space (i.e., there are N RVs) [140, p. 252f.]:

Definition 2.3 (N-dimensional Continuous Orthogonal Space)**.** *An N-dimensional orthogonal space is defined by a tensor grid expansion of one-dimensional orthogonal polynomial spaces in the following manner:*

$$W_N^{(D)} \equiv \bigotimes_{|\mathbf{d}|\leq D} W^{(d_i)}$$

Here, $|\mathbf{d}|$ is a multi-index operator defined by $|\mathbf{d}| = \sum_{i=1}^{N} d_i$.

Definition 2.3 defines a combination of one-dimensional orthogonal spaces in a manner that a tensor grid of maximum order D is achieved. Figure 2.12 shows a visualization of a two-dimensional tensor grid: Here, the red crosses give the positions of the values in the created tensor grid from the one-dimensional grids ($\theta_1 \sim \mathcal{N}(\mu = 0, \sigma = 1)$ and $\theta_2 \sim \mathcal{U}(a = -1, b = 1)$). These are depicted by the blue (first uncertain parameter) and green circles (second uncertain parameter) and provide the basic values. Here, the one-dimensional grid of the first uncertain parameter has a dimension of four ($d_1 = 4$), while the second one has a dimension of two ($d_2 = 2$). This shows the possibility to combine different order one-dimensional grids as well as how the tensor grid is created from them by applying Definition 2.3.

A general prerequisite of the gPC expansion in the N-dimensional case is now that a properly defined probability space is used, as given in e.g., Definition B.2: Here, all RVs must be mutually independent (Section B.5). Take into account that this independence can be achieved using e.g., a Karhunen-Loève transformation [68], if necessary. This transformation is essentially a principle axis transformation.

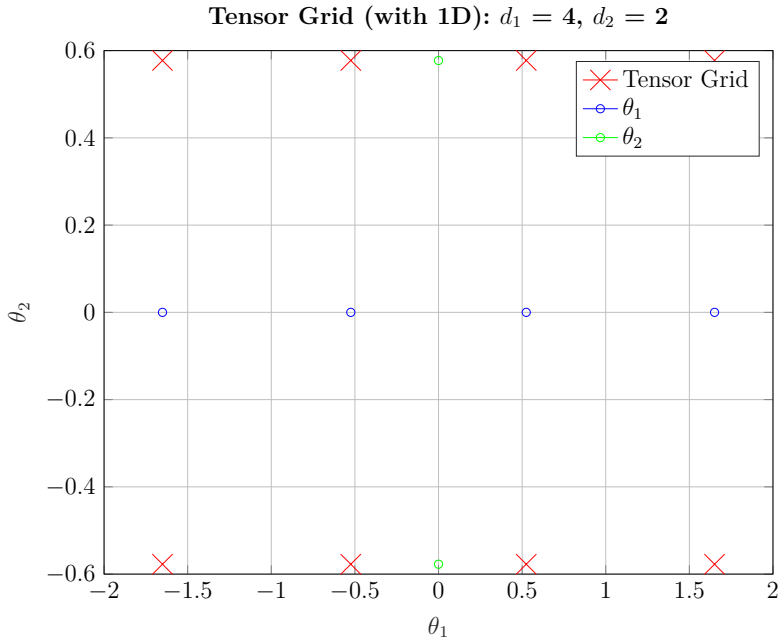

Figure 2.12: *Visualization of two-dimensional tensor grid based on one-dimensional grids and full expansion.*

Having defined a proper probability space as well as secured mutual independence, all PDFs of the scalar RVs and their respective supports can be multiplied according to (B.2) as follows [140, p. 249]:

$$\rho_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \prod_{i=1}^{N} \rho_{\Theta_i}(\theta_i) \in \mathbb{R}, \qquad \boldsymbol{\Omega} = \prod_{i=1}^{N} \Omega_i \subset \mathbb{R}^N \tag{2.59}$$

With (2.59), the N-dimensional orthogonal polynomial space is defined by multiplying all one-dimensional orthogonal polynomials as follows [140, p. 253]:

$$\Phi^{(m)}(\boldsymbol{\theta}) = \prod_{i=1}^{N} \phi^{(m_i)}(\theta_i) \in \mathbb{R}, \qquad m_1 + \ldots + m_N \leq D \tag{2.60}$$

It should be noted that the multivariate expansion order of the orthogonal polynomial is D (see Definition 2.3). Additionally, the normalization constants of the orthogonal polynomials can also be multiplied like the orthogonal polynomials in (2.60).

Using (2.59) and (2.60), (2.52) can directly be expanded to the N-dimensional space as follows [140, p. 254]:

$$\mathbf{y}(\mathbf{z};\boldsymbol{\theta}) \approx P_N^{(D)}\mathbf{y} = \mathbf{y}_N^{(D)}(\mathbf{z};\boldsymbol{\theta}) = \sum_{m=0}^{M-1} \hat{\mathbf{y}}^{(m)}(\mathbf{z})\, \Phi^{(m)}(\boldsymbol{\theta}), \qquad M - 1 = \binom{N+D}{N} \tag{2.61}$$

Here, M is the expansion order of the gPC and $\hat{\mathbf{y}}^{(m)}(\mathbf{z}) \in \mathbb{R}^{n_y}$ are the multi-dimensional expansion coefficients matching the size of the output vector. It should be noted that the full tensor grid, i.e., the multi-index operator in Definition 2.3 for the tensor grid

allows all polynomial order combinations, is also sometimes used [140, p. 253]. This yields an expansion order of $M - 1 = D^N$, which is normally much larger than the binomial coefficient in (2.61) and does not improve the accuracy significantly [140, p. 253ff.]. Thus, this thesis applies the tensor grid and multivariate gPC expansion with the corresponding expansion order as introduced in Definition 2.3 and (2.61) respectively.

Further take into account that, just like for the scalar case in (2.54), (2.61) can be rewritten as a matrix-vector operation:

$$\underbrace{\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)}_{\in\mathbb{R}^{n_y}} \approx \underbrace{\left[\hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right) \quad \dots \quad \hat{\mathbf{y}}^{(M-1)}\left(\mathbf{z}\right)\right]}_{\in\mathbb{R}^{n_y\times M}} \cdot \underbrace{\begin{bmatrix} \Phi^{(0)}\left(\boldsymbol{\theta}\right) \\ \vdots \\ \Phi^{(M-1)}\left(\boldsymbol{\theta}\right) \end{bmatrix}}_{\in\mathbb{R}^{M}} \tag{2.62}$$

Here, it is again clear that there exist M expansion coefficients with M corresponding orthogonal polynomials. Further, it can be seen that each of the output expansion coefficients is multiplied using the same series of orthogonal polynomials because the uncertainties acting on the dynamic system are the same.

The expansion coefficients themselves are then also directly obtained by expanding (2.58) as follows [140, p. 254]:

$$\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right) = \frac{1}{\left[h^{(m)}\right]^2} \int_{\boldsymbol{\Omega}} \mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \Phi^{(m)}\left(\boldsymbol{\theta}\right) \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right) \, \mathrm{d}\boldsymbol{\theta}, \qquad m = 0, \dots, M-1 \tag{2.63}$$

Take into account that the orthogonal space defined in Definition 2.3 grows rapidly with the number of uncertainties due to the fact that a standard tensor product is used ("curse of dimensionality"). To overcome this issue, sparse grids can be implemented that use a suitable subset of tensor grids to achieve a good approximation of the tensor grid with fewer required expansion coefficients. As a rule of thumb, a sparse grid is generally required starting from $N \geq 5$ and the origins of this method date back to the Russian mathematician Smolyak. Within this thesis, a sparse grid is defined as follows [142, p. 1127]:

Definition 2.4 (Smolyak Sparse Grid)**.** *A sparse collocation grid, in the sense of* Smolyak*, is defined as a subset of standard tensor products in the following manner:*

$$W_N^{(D)} = \sum_{k_{sg}+1\leq|\mathbf{d}|\leq k_{sg}+N} (-1)^{k_{sg}+N-|\mathbf{d}|} \binom{N-1}{N+k_{sg}-|\mathbf{d}|} \left(W^{(d_1)} \otimes \dots \otimes W^{(d_N)}\right)$$

Here, k_{sg} is the approximation level of the Smolyak *grid. Again, $|\mathbf{d}|$ symbolizes the multi-index operator.*

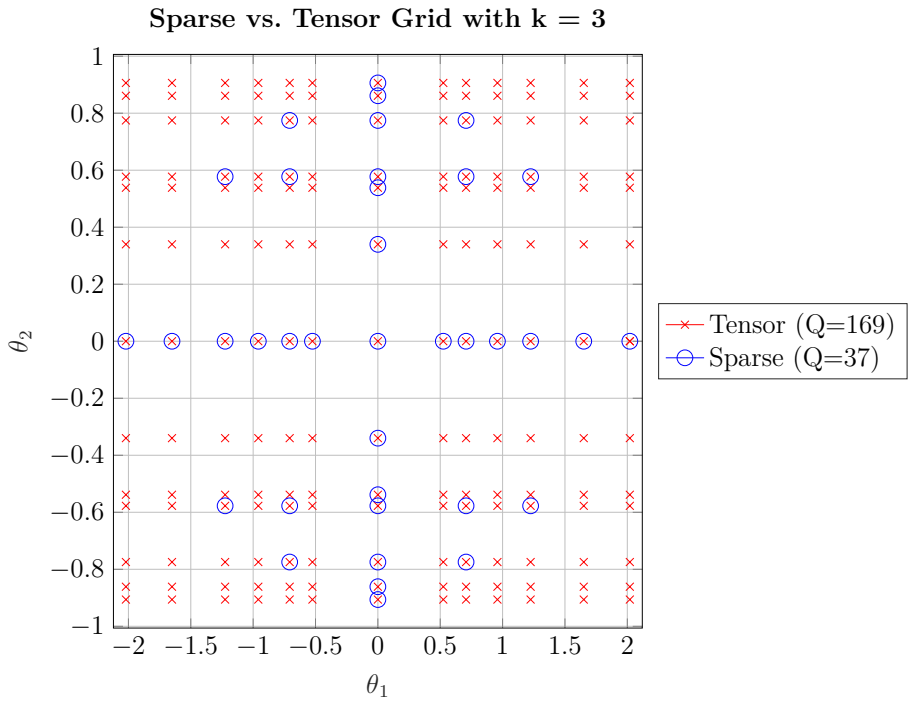

Figure 2.13: *Visualization of two-dimensional sparse grid based on Smolyak rule in comparison to full tensor grid based on one-dimensional grids from sparse grid algorithm.*

Definition 2.4 states that an arbitrary accurate representation of a tensor grid can be achieved by a sparse grid using a sufficiently large approximation level k_{sg} [142].

The sparse grid given in Definition 2.4 can also be visualized as shown in Figure 2.13. The same definition for the uncertainties ($\theta_1 \sim \mathcal{N}(\mu = 0, \sigma = 1)$; $\theta_2 \sim \mathcal{U}(a = -1, b = 1)$) as in Figure 2.12 is applied. The approximation order of the sparse grid is chosen to be $k_{\text{sg}} = 3$. In Figure 2.13, the blue circles depict the created sparse grid (number of points: $Q = 37$), while the red crosses denote the tensor grid (number of points: $Q = 169$) that is created from the one-dimensional grids that are used to create the sparse grid. A first observation in Figure 2.13 is that the sparse grid only requires 22% of the points that the tensor grid requires. Thus, the sparse grid is more efficient in the approximation. In addition, it is clear that the sparse grid mainly uses points that are on the coordinate axes or close to the coordinate axes. This is due to the fact that points close to the coordinate axes normally have an increased influence on the response surface compared to points far away. Thus, it is reasonable to "cluster" points around the center. Further examples and comparisons of sparse and tensor grids, especially regarding the efficiency with a large number of uncertainties, can also be found in study [142].

With the gPC expansion defined for the scalar as well as the multivariate case, the next subsection deals with the calculation of the expansion coefficients, which requires the approximation of the integral in (2.63).

2.3.3 Stochastic Collocation

This subsection introduces one possible method to calculate an approximation for the integral in e.g., (2.63), and thus the expansion coefficients, by means of STOCHASTIC COLLOCATION. This is also known as discrete expansion. The definition of the stochastic collocation (SC) method in a single dimension, in the context of the gPC framework, is given as follows (after [141, p. 79]):

Definition 2.5 (Stochastic Collocation)**.** *Let* $\Theta_Q = \left\{\theta^{(j)},\, j = 1, \ldots, Q\right\} \subset I_\Theta \subset \mathbb{R}^Q$ *be a set of (prescribed) collocation nodes in the random space, where* $Q \geq 1$ *is the number of nodes, and let* $y(\Theta_Q) = \left\{y^{(j)},\, j = 1, \ldots, Q\right\}$ *be the set of solutions obtained from e.g., the OCP (see* (2.1)*), at those nodes* Θ_Q*. Then find* $w(\Theta_Q) \in \Pi(\Theta_Q)$ *in a proper polynomial space* $\Pi(\Theta_Q)$ *such that* $w(\Theta_Q)$ *is an approximation of the true solution* $y(\Theta_Q)$ *in the sense that* $\|w(\Theta_Q) - y(\Theta_Q)\|$ *is sufficiently small in a strong norm defined in* I_Θ*, i.e.,*

$$\|w(\Theta_Q) - y(\Theta_Q)\| \to 0, \qquad Q \to \infty$$

The norm is to be determined by the approximation approach and is typically a L^p *norm.*

Take into account that posterior error estimations on the approximation accuracy of the SC method are also available from e.g., [56].

It should be noted that in the context of gPC the full tensor grid for the SC, i.e., the multi-index operator in Definition 2.3 allows all polynomial order combinations, is normally used [140, p. 253]. This yields a number of nodes of $Q = D^N$. This expansion on the full grid is generally required to have the correct weighting of the solutions and is also employed in this thesis for the tensor grid examples (sparse grids, as defined in Definition 2.4, can be applied analogously to reduce the number of required nodes).

Thus, the basic problem is to find nodes that yield a good approximation of the response surface to calculate the integral in (2.58) or (2.63) respectively: As orthogonal polynomials and their corresponding PDFs, according to Table 2.2, are used in the gPC expansion coefficient calculation, a best fit approximation is given by GAUSSIAN QUADRATURE (also: "pseudo-spectral projection"). This is due to the fact that GAUSSIAN QUADRATURE is defined with respect to the used PDF as follows [5, p. 1184]:

$$\int_\Omega y(\mathbf{z};\theta)\,\rho_\Theta(\theta)\,\mathrm{d}\theta \approx \sum_{j=1}^{Q} \underbrace{y\left(\mathbf{z};\theta^{(j)}\right)}_{y^{(j)}} \alpha^{(j)} \tag{2.64}$$

The expansion formula in (2.64) is exact for all polynomial integrands up until degree $2Q-1$ [5, p. 1184]. Thus, even for a small number of nodes a high accuracy is achieved. In (2.64), $\alpha^{(j)}$ are the normalized quadrature weights and $\theta^{(j)}$ are the quadrature nodes ("sampling

points"). These are chosen according to Table 2.2, i.e., in connection with the orthogonal polynomials and the PDF. It should be noted that the quadrature weights must be normalized, i.e., $\sum_{j=1}^{Q} \alpha^{(j)} \equiv 1$, to fulfill that the integral over the PDF is one (see (B.3)). This is different compared to a standard GAUSSIAN QUADRATURE [5, p. 1184].

Take into account that the quadrature nodes in (2.64) are the zeros of the orthogonal polynomial of order d, i.e.:

$$\Phi^{(d)}\left(\theta^{(j)}\right) \equiv 0, \quad j = 1, \ldots, Q \tag{2.65}$$

There are then always $Q = d$ distinct, real-valued nodes located in the support of Ω for a scalar orthogonal polynomial [2, p. 787]. Thus, it is secured that the model is evaluated within the specified bounds.

The weights for the GAUSSIAN quadrature are calculated by integrating LAGRANGE polynomials of the quadrature nodes from (2.65) over the uncertainty domain weighted by the PDF. For the scalar quadrature node this is defined as follows [141, p. 40]:

$$\alpha^{(j)} = \int_{\Omega} \rho_{\Theta}(\theta) \prod_{\substack{i=1 \\ i \neq j}}^{Q} \frac{\theta - \theta^{(i)}}{\theta^{(j)} - \theta^{(i)}} \, \mathrm{d}\theta, \quad j = 1, \ldots, Q \tag{2.66}$$

It should be mentioned here that it is important to calculate the nodes in (2.65) and the weights in (2.66) with respect to the PDF as defined in Table 2.2. Otherwise, the SC in (2.68) returns wrong expansion coefficients. This may require adapting readily implemented GAUSSIAN quadrature schemes [30, p. 127].

Some node and weight combinations for GAUSSIAN quadrature with Hermite polynomials (GAUSSIAN PDF) and Legendre polynomials (UNIFORM PDF) are listed in Table 2.3 and Table 2.4 respectively. It should be noted that the tables merely provide an overview and should not be seen as a complete reference. Here, it can be observed that the nodes and weights are distributed symmetrically for these symmetric PDFs. Additionally, it is clear that nodes with a higher probability in the PDF (here e.g., in the context of the center values in the GAUSSIAN PDF) are weighted with a larger magnitude than nodes at the edges (i.e., far away from the center). Furthermore, it can also be seen that the weights are indeed normalized. It should additionally be noted in Table 2.3 and Table 2.4 that the SC nodes are deterministic values, which can be used to solve deterministic OCPs. This is a major benefit of the generalized polynomial chaos-stochastic collocation framework (gPC-SC) approach, as it does not require changes in the deterministic model formulation. Finally, the nodes in Table 2.3 and Table 2.4 are not "clustered" around the center, but actually they are "clustered" at the boundaries of the random space. This is specifically seen for the Legendre polynomials in Table 2.4. Although this seems to be non-intuitive in the first place, this behavior has the specific reason to cancel the RUNGE'S PHENOMENON [114]: This phenomenon occurs when increasing the number of SC points on an equidistant grid and results in oscillations close to the bound values.

Table 2.3: *Nodes and weights for Hermite polynomials/*Gaussian *uncertainty in generalized polynomial chaos stochastic collocation for* Gaussian *quadrature.*

Order d	Nodes $\theta^{(j)}$	Weights $\alpha^{(j)}$
2	$\begin{bmatrix}\theta^{(1)}\\ \theta^{(2)}\end{bmatrix} \approx \begin{bmatrix}-0.7071\\ 0.7071\end{bmatrix}$	$\begin{bmatrix}\alpha^{(1)}\\ \alpha^{(2)}\end{bmatrix} = \begin{bmatrix}0.5\\ 0.5\end{bmatrix}$
3	$\begin{bmatrix}\theta^{(1)}\\ \theta^{(2)}\\ \theta^{(3)}\end{bmatrix} \approx \begin{bmatrix}-1.2247\\ 0\\ 1.2247\end{bmatrix}$	$\begin{bmatrix}\alpha^{(1)}\\ \alpha^{(2)}\\ \alpha^{(3)}\end{bmatrix} \approx \begin{bmatrix}0.1667\\ 0.6666\\ 0.1667\end{bmatrix}$
4	$\begin{bmatrix}\theta^{(1)}\\ \theta^{(2)}\\ \theta^{(3)}\\ \theta^{(4)}\end{bmatrix} \approx \begin{bmatrix}-1.6507\\ -0.5246\\ 0.5246\\ 1.6507\end{bmatrix}$	$\begin{bmatrix}\alpha^{(1)}\\ \alpha^{(2)}\\ \alpha^{(3)}\\ \alpha^{(4)}\end{bmatrix} \approx \begin{bmatrix}0.0459\\ 0.4541\\ 0.4541\\ 0.0459\end{bmatrix}$

Table 2.4: *Nodes and weights for Legendre polynomials/*Uniform *uncertainty in generalized polynomial chaos stochastic collocation for* Gaussian *quadrature.*

Order d	Nodes $\theta^{(j)}$	Weights $\alpha^{(j)}$
2	$\begin{bmatrix}\theta^{(1)}\\ \theta^{(2)}\end{bmatrix} \approx \begin{bmatrix}-0.5774\\ 0.5774\end{bmatrix}$	$\begin{bmatrix}\alpha^{(1)}\\ \alpha^{(2)}\end{bmatrix} = \begin{bmatrix}0.5\\ 0.5\end{bmatrix}$
3	$\begin{bmatrix}\theta^{(1)}\\ \theta^{(2)}\\ \theta^{(3)}\end{bmatrix} \approx \begin{bmatrix}-0.7746\\ 0\\ 0.7746\end{bmatrix}$	$\begin{bmatrix}\alpha^{(1)}\\ \alpha^{(2)}\\ \alpha^{(3)}\end{bmatrix} \approx \begin{bmatrix}0.2778\\ 0.4445\\ 0.2778\end{bmatrix}$
4	$\begin{bmatrix}\theta^{(1)}\\ \theta^{(2)}\\ \theta^{(3)}\\ \theta^{(4)}\end{bmatrix} \approx \begin{bmatrix}-0.8611\\ -0.34\\ 0.34\\ 0.8611\end{bmatrix}$	$\begin{bmatrix}\alpha^{(1)}\\ \alpha^{(2)}\\ \alpha^{(3)}\\ \alpha^{(4)}\end{bmatrix} \approx \begin{bmatrix}0.1739\\ 0.3261\\ 0.3261\\ 0.1739\end{bmatrix}$

These oscillations can be reduced by "clustering" points at the interval edges, i.e., by using a non-equidistant grid, and thus, is generally used in polynomial interpolation/integration methods, for which gPC-SC is an example.

As a reminder, Table 2.2 introduced the orthogonal polynomials for gPC within their standardized support and parameter setting (e.g., the PDF of the Hermite polynomials is the standard Gaussian PDF). Thus, especially the nodes are also calculated with respect to this standard setting (Tables 2.3 and 2.4). If a non-standard PDF support or parameter setting should be used, an iso-probabilistic interval transformation can be conducted [30, p. 89]. The transformed parameter is then used as the sampling point for the model (i.e., to calculate $y\left(\theta^{(j)}\right)$), but not for the orthogonal polynomials (as these are defined with respect to the standard domain). As an example, the iso-probabilistic transformation for the uniform PDF from its basic domain (basic), $[-1;1]$, to a transformed domain (tra), $[a;b]$, is given as follows:

$$\theta_{\text{tra}}^{(j)} = \frac{b-a}{2} \cdot \theta_{\text{basic}}^{(j)} + \frac{b+a}{2}, \qquad j = 1, \ldots, Q \tag{2.67}$$

This transformation can also be conducted for other PDFs of Table 2.2 [30, p. 89 & 142ff.]. It should be noted that it is not specifically distinguished between basic and transformed domain in the following for the sake of readability. But, it can be stated that the orthogonal polynomials are always evaluated with the samples from the basic domain, while the system response is generally evaluated in the transformed domain.

Furthermore, it is important to note that it is sufficient to look at the scalar case for the calculation of quadrature nodes by (2.65) and weights by (2.66), due to the fact that multi-variate gPC expansions are generally created using a tensor (Definition 2.3) or a sparse grid (Definition 2.4). As these merely combine one-dimensional (scalar) grids, (2.64) can directly be used in (2.63) (or equivalently in (2.58)) as follows to approximate the integral by Gaussian quadrature (normalized orthogonal polynomials according to Definition 2.2 are assumed from now on and therefore, the normalization constant $\left[h^{(m)}\right]^2$ is left out of the equations for the sake of simplicity):

$$\hat{\mathbf{y}}^{(m)}(\mathbf{z}) \approx \sum_{j=1}^{Q} \underbrace{\mathbf{y}\left(\mathbf{z}; \boldsymbol{\theta}^{(j)}\right)}_{\mathbf{y}^{(j)}} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \tag{2.68}$$

Here, (2.68) states that it is only necessary to solve the OCP, defined in (2.1), on the set of independent quadrature nodes and weights $\left\{\boldsymbol{\theta}^{(j)}, \alpha^{(j)}\right\}$ to get the desired system response $\mathbf{y}^{(j)} = \mathbf{y}\left(\mathbf{z}; \boldsymbol{\theta}^{(j)}\right)$ and be able to calculate the expansion coefficients $\hat{\mathbf{y}}^{(m)}(\mathbf{z})$. Therefore, only deterministic OCPs must be solved, in order to get a stochastic representation of the system in the form of (2.61) (or equivalently (2.52)). Once more, this shows the huge benefit of the gPC-SC, as it does not require the alteration of the deterministic OCP that is normally extensively tested and therefore, can be assumed to be a good representation of the real system behavior (except for the uncertainties).

Take into account that the multi-dimensional case in (2.68) allows the combination of different uncertain parameter PDFs (as long as these are mutually independent) as well as different expansion orders for each of the uncertain parameters. This allows a tailored adaptation to the problem structure.

Finally, it is once more reminded that the approach applied in this thesis uses the connection of the PDF to an orthogonal polynomial from the Wiener-Askey-Scheme (Table 2.2). Nonetheless, extensions to arbitrary PDFs using arbitrary polynomial chaos [136] or Gaussian mixture models [103, 129] can be applied as well within the methods developed in this thesis, which was already shown by the author in [103].

2.3.4 Statistical Information from Generalized Polynomial Chaos

A major advantage of the gPC method is the fact that (2.52), and consequently also (2.61) for the multivariate case, become an analytic representation of the uncertain system (or at least a good approximation). Here, it is only necessary to consider enough expansion coefficients to reach a desired accuracy. This makes it simple to calculate statistical information from the gPC representation, and thus, the system. A more detailed derivation of the formulas given in the following can be found in Appendix D. The definition of general statistical moments is given in Subsection B.2.2.

For the expected/mean value, the calculation using the gPC expansion in (2.61) is as follows [141, p. 254]:

$$\begin{aligned}\mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] &= \mu\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] \approx \mathbb{E}\left[\mathbf{y}_N^{(D)}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] \\ &= \int_{\boldsymbol{\Omega}} \left(\sum_{m=0}^{M-1} \hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right) \Phi^{(m)}\left(\boldsymbol{\theta}\right)\right) \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\, \mathrm{d}\boldsymbol{\theta} = \hat{\mathbf{y}}^{(0)}\end{aligned} \tag{2.69}$$

The variance, i.e., the square of the standard deviation σ, has an equally simple representation (again using (2.61)) [141, p. 255]:

$$\mathrm{var}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \sigma^2\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \mathbb{E}\left[\left(\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) - \mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\right)^2\right] \approx \sum_{m=1}^{M-1} \left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2 \tag{2.70}$$

Using (2.69) and (2.70), the statistical properties of optimal trajectories can easily be calculated. It should be noted that both results only depend on the expansion coefficients, which makes it convenient to evaluate them. On the other hand, further statistical moments, e.g., covariance, skewness, and kurtosis, can also be derived and used. These are then generally depending on the orthogonal polynomials, in addition to the expansion coefficients. Their formulas are given in Appendix D.

The applications within this thesis concentrate on mean values and variances, mainly because the direct computation of higher order moments become quite expensive (and can also be done via sampling the gPC expansion in (2.52)), the formulas given in (2.69) and (2.70) are most relevant.

As already noted, for the SubSim methodology (Subsection 2.2.4) the variance in (2.70) is no longer a suitable measure to describe the dispersion of the data for small failure probabilities. Therefore, the CoV, c_V, which is a standardized measure of dispersion for statistical moments can be used. Compared to the standard deviation, it is non-dimensional and independent of the original data magnitude. In the gPC framework, the CoV, using the results of (2.69) and (2.70), is as follows [9, p. 25]:

$$c_V\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \frac{\sigma\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]}{\mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]} \approx \frac{\sqrt{\sum_{m=1}^{M-1}\left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2}}{\hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right)} \tag{2.71}$$

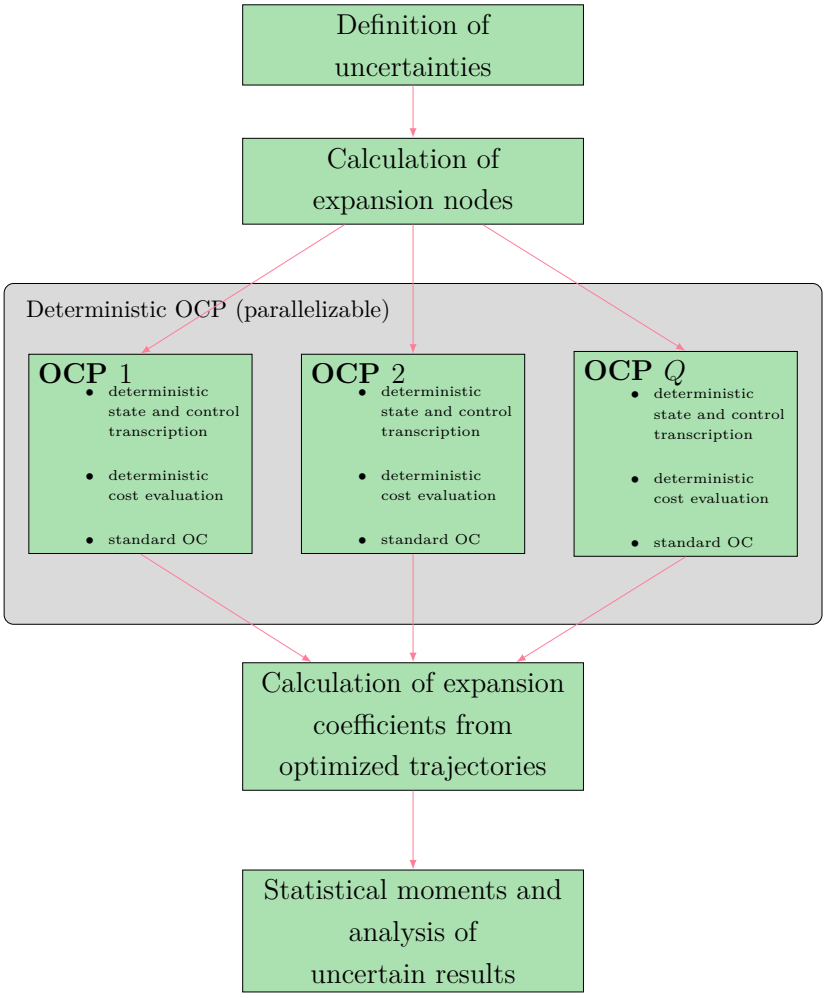

Figure 2.14: *General, basic procedure of combining generalized polynomial chaos method using stochastic collocation with an optimal control framework to calculate response surface of system to uncertain parameters.*

As mentioned, this standardization is especially important with the rare-events considered in this study, as the mean value will be small and therefore, the standard deviation must be seen with respect to this small value.

2.3.5 Generalized Polynomial Chaos in Optimal Control

This subsection gives an overview on the basic idea of incorporating the gPC method (Section 2.3) in an OCP (Section 2.1). Here, Figure 2.14 shows the idea of the fundamental framework: It can be seen that each OCP is solved in an independent way at the SC nodes, provided by the generalized polynomial chaos-stochastic collocation framework (gPC-SC). The responses are afterward used to calculate the gPC expansion. This fundamental framework does not calculate robust trajectories, but only a response surface of the uncertain trajectories. In order to calculate robust trajectories, the framework is elaborated on in the thesis.

Additionally, Figure 2.15 gives a formal overview on the parts of the framework, their dependencies developed throughout this chapter, and how they interact with each other. This should give a complete overview on how the gPC framework with OC is set up and what parts are necessary to get to the finite sum gPC approximation. It is important that the gPC-SC provides the samples to evaluate the OCP, while neither changing the OCP itself nor the underlying dynamic model. This means that the model is evaluated in the deterministic domain as desired.

Overall, it can be stated that the incorporation of the gPC-SC in the OCP formulation is straightforward because gPC can be considered as a wrapper around the already implemented FALCON.m OC framework in its basic form. This is due to the fact that only deterministic OCPs must be solved, which is what FALCON.m is designed to do. Therefore, it is easy to use and does not interfere with the core functionalities of the OC software, in this case FALCON.m.

Finally, an important aspect of the discussion of gPC in OC is the question if it leads to biased statistical moments like the MCA (Figure 2.4; e.g., for the application in CCs). The general idea of approaching the question remains similar to the MCA case and is visualized in Figure 2.16: Again, the feasible function values are shown in green and the infeasible in red. The blue error bar is, once more, obtained from the constrained MCA (i.e., discarding infeasible samples). The black circles denote the SC nodes used in a fifth-order gPC expansion. It should be noted that these are all located in the feasible domain. By the gPC-SC procedure the black error bar is calculated, which matches the exact error bar (around magenta cross in black square) both in mean as well as standard deviation very well. Thus, the gPC expansion recovers the actual solution without bias due to the fact that it approximates the uncertain output response as a whole. Therefore, as long as all SC node NLP solutions are successful, it can be assumed that the gPC-SC method gives a good approximation of the response surface, which can be used for detailed analysis of also the infeasible regions, e.g., in the context of CCs (Subsection 2.4.4).

2.3.6 Accuracy and Convergence Properties of Generalized Polynomial Chaos Expansion

This final section of the introduction to gPC deals with a convergence analysis of the gPC expansion. It covers both a comparison to MCA ($n_s = 1000$) as well as a comparison of different gPC expansion orders. The main idea is to get an overview on the convergence properties of the statistical moments, in this case for mean value and standard deviation. Mathematical concepts and derivations of this convergence properties can e.g., also be found in [38, 138].

As an example, a first order lag with uncertainty in the time constant T is considered (u is the command, y the measured output with corresponding dynamics, and t the time; see [21, p. 161]):

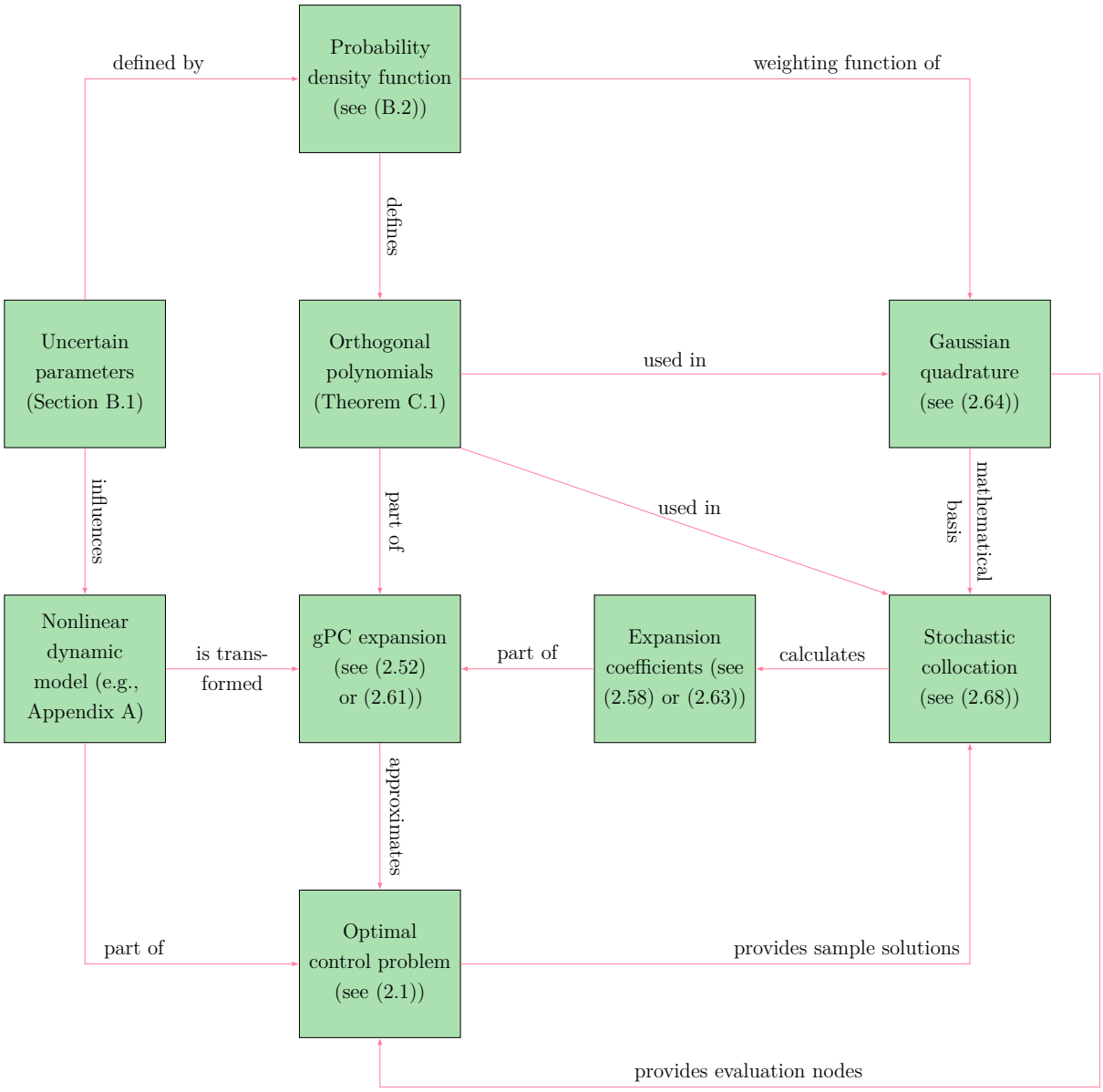

Figure 2.15: *Dependencies between different parts of the optimal control problem, generalized polynomial chaos, as well as the dynamic model and how they are connected with as well as influenced by each other.*

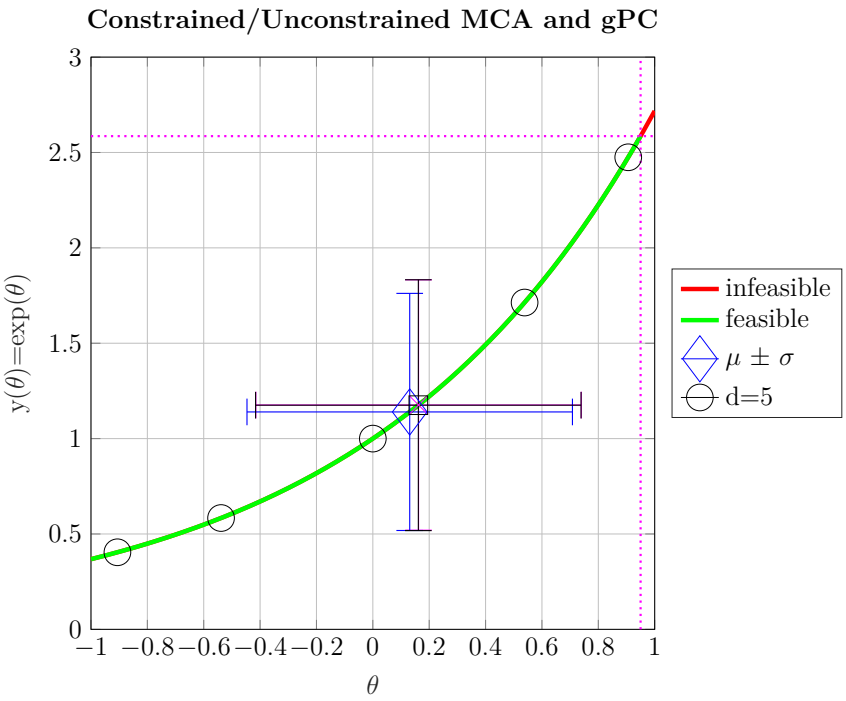

Figure 2.16: *Comparison of constrained and unconstrained Monte Carlo analysis as well as generalized polynomial chaos for calculated mean and standard deviation.*

$$\dot{y}\left(t;T\right)=\frac{1}{T}\left(u-y\right) \tag{2.72}$$

The time constant T is defined using a UNIFORM PDF as follows:

$$T\sim\mathcal{U}\left(a=0.5s,b=1.5s\right),\qquad\rho_{\Theta}\left(\theta\right)=1 \tag{2.73}$$

This choice ensures a simpler form of the emerging integrals for the statistical moments as the PDF has a simple representation.

A step response of this first-order lag system is considered as the response example and defined as follows [21, p. 200]:

$$y\left(t;T\right)=1-\exp\left(\frac{t}{T}\right) \tag{2.74}$$

Using (B.4) (definition of the mean value) and the PDF in (2.73), a semi-analytic solution for the mean value can be derived (Appendix E):

$$\mathbb{E}\left[y\right]=\left[\mathrm{Ei}\left(-2t\right)-\mathrm{Ei}\left(-\frac{2t}{3}\right)\right]t+\frac{\exp\left(-2t\right)}{2}-\left[1.5\cdot\exp\left(-\frac{2t}{3}\right)\right]+1 \tag{2.75}$$

Here, the exponential integral Ei is defined as follows [2, p. 228]:

$$\mathrm{Ei}\left(t\right)=\int_{\tau=-\infty}^{t}\frac{\exp\left(\tau\right)}{\tau}\,\mathrm{d}\tau,\qquad t>0 \tag{2.76}$$

Similarly, a formula for the standard deviation can be derived that is not stated here for the sake of brevity, but given in Appendix E.

The time development of the gPC solution (solid lines) for the mean value and the standard deviation of different gPC expansion orders ($d = M = Q = 2, \ldots, 10$) with respect to the semi-analytic solution (dashed red line) can be seen in Figure 2.17. It is clear that there are basically no differences between the gPC solutions of different order for the mean value. The standard deviation, however, is not accurate with only a second order expansion but also gets very accurate when increasing the expansion order.

The convergence history of gPC and MCA ($n_s = 1000$), in comparison to the semi-analytic solution for mean and standard deviation, is depicted in Figure 2.18. Furthermore, Figure 2.19 shows the same results as in Figure 2.18 zoomed-in for the initial samples to better compare the convergence properties for only few samples. Both figures depict the semi-analytic solution in dashed-red, the gPC solution in solid green, and the MCA solution ($n_s = 1000$) in solid blue. The figures show the step response result at $t = 1s$: This is the time after which the step response for the mean system (i.e., with $T = 1s$) reaches approximately 63% of the stationary value and thus, is still in the transient phase (see Figure 2.17). Therefore, this point can be seen as a good indicator for the convergence properties and accuracy of the different approximation algorithms.

Figure 2.18 shows that the MCA converged to the semi-analytic solution after approximately 100 samples for both mean value and standard deviation, while gPC shows almost no difference to the semi-analytic solution even with only two samples. This is especially evident by looking at the zoomed-in plot given in Figure 2.19. Here, the mean value shows a very good accordance with two samples, while the standard deviation shows a good convergence starting from three samples on. This was also already clear from Figure 2.17. It is additionally clear that a MCA with this number of samples is still fairly far off from the semi-analytic solution and thus, not as efficient in the approximation like the gPC method.

Overall, this section, together with literature results as in e.g., [38, 138], justifies the use of gPC in OCPs, as it provides a very fast and efficient method to calculate the statistical properties of an uncertain system accurately.

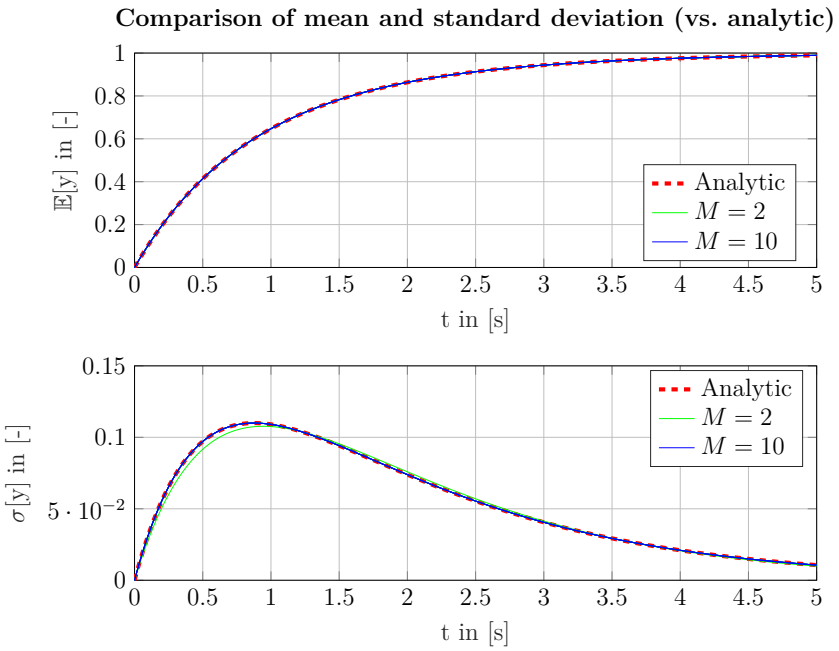

Figure 2.17: *Comparison of mean value and standard deviation convergence for different generalized polynomial chaos expansion orders to analytic solution over step time.*

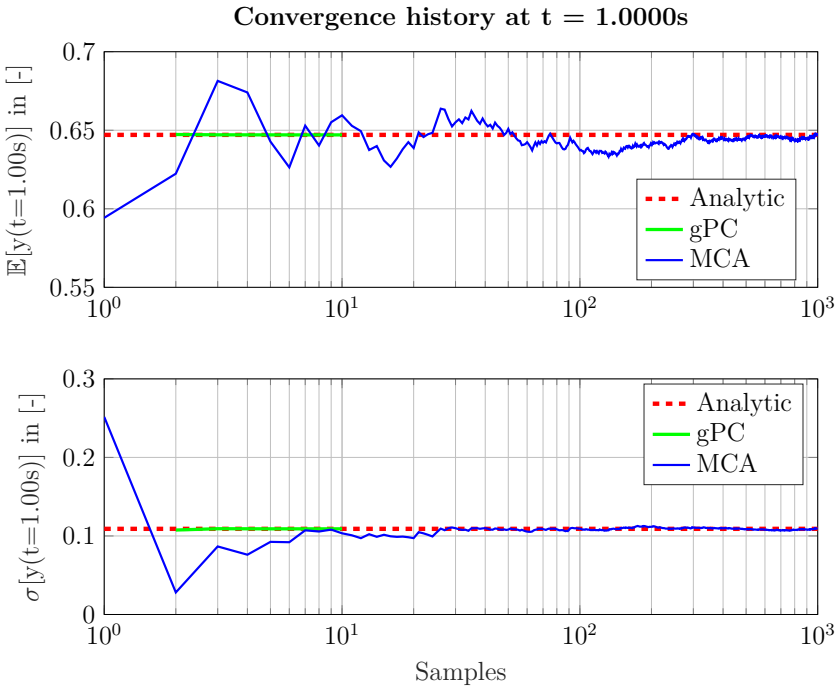

Figure 2.18: *Comparison of mean value and standard deviation convergence for generalized polynomial chaos and Monte Carlo analysis to semi-analytic solution one second after step initiation.*

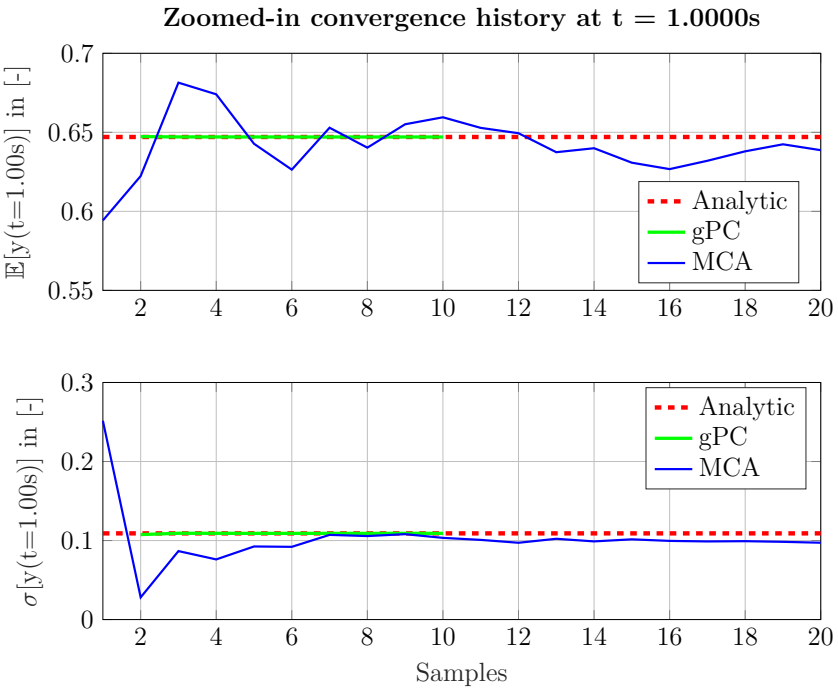

Figure 2.19: *Comparison of mean value and standard deviation convergence for generalized polynomial chaos and Monte Carlo analysis to semi-analytic solution zoomed-in for the first 20 samples one second after step initiation.*

2.4 Advanced Optimization Methods

This section gives an overview on advanced optimization methods to conduct a detailed analysis of the dynamic system and its optimal result, which is required in this thesis. These methods rely on the NLP transcription and solution (Subsection 2.1.2) and thus, provide add-ons to this formulation. Here, Subsection 2.4.1 gives an overview on Pareto analysis techniques, i.e., methods to calculate optima for multiple (opposing) objectives in the OCP formulation. Then, Subsection 2.4.2 introduces methods for sensitivity analysis. These methodologies are used to calculate the influence of an invariant parameter on the optimal solution when varying it ("sensitive parameter"). Thus, the methods can be used to update trajectories based on varying parameters online, as they represent a first-order Taylor series expansion [45, p. 295]. Subsection 2.4.3 gives an overview on bi-level OC methods. These methods are often used to decompose a single OCP into multiple, smaller OCPs, which are dependent on a dynamic model, and (generally) one parameter optimization problem that shapes their behavior. Afterward, Subsection 2.4.4 introduces the concept of OC with chance constraints (CCs). These CCs are then used to assign probabilistic constraints within the OCP formulation. Finally, Subsection 2.4.5 introduces the concept of distributed open-loop direct optimal control (DOC) that provides an efficient way to split the introduced gPC-SC framework (Subsection 2.3.5) by solving

the different sub-problems independently of each other, and finally combining them in a connection level using connection variables. These ensure that the original OCP is still solved.

Especially the final two sections introduce basic methods that are required and applied in the following chapters to calculate robust, optimal trajectories.

2.4.1 Pareto Analysis

A Pareto analysis (also: Pareto optimization) is conducted when there are multiple (opposing) objectives within an OCP, specifically the cost function. It is therefore also called multi-criterion optimization. The resulting optimal points create a so-called Pareto frontier, i.e., a boundary where an improvement of one cost function part would automatically lead to a deterioration of another cost function part. Comprehensive introductions and reviews have recently been conducted in e.g., [28, 69, 75].

Pareto problems are very common in OC, although they are seldom analyzed in the way of a Pareto analysis, because most often only one point on the Pareto frontier is calculated. Figure 2.20 shows the general Pareto problem with two opposing objectives in the cost function. The blue line is the Pareto frontier, while the green crosses are sub-optimal solutions. In this example, solutions left of the Pareto frontier cannot be reached as they are unphysical ("utopia points"; e.g., if J_1 is the flight time between two points even with maximal speed there is a minimal time required to travel between them; this would then also increase the J_2, which might be the fuel consumption). Thus, each change on the Pareto frontier normally benefits one cost function part, while it is non-beneficial for the other part(s). Here, it is up to the user to calculate the Pareto point in a manner that it suits the application requirements well. This is normally done by weighting the cost function influences appropriately.

One straightforward way of calculating a Pareto frontier is thus by a so-called "weighted sum scalarization" [17, p. 90]: Here, the different cost function parts are summed up and weighted to create a scalar cost function. Depending on the weights, different points on the Pareto frontier can be calculated. Mathematically, the Pareto problem can then be stated as follows [17, p. 90]:

$$\begin{aligned} \min \quad & J_{\text{sum}} = \sum_{i=1}^{n} w_i \cdot J_i \\ \text{s.t.} \quad & w_i > 0, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} w_i = 1 \end{aligned} \tag{2.77}$$

Here, the Pareto problem in (2.77) creates two issues: First of all, non-convex Pareto frontiers cannot be covered by this approach and second of all, the choice of the weights to calculate an accurate representation of the Pareto frontier is not straightforward [17, p. 91ff.]. Therefore, multiple developments have been made to this approach [17, p. 91ff.],

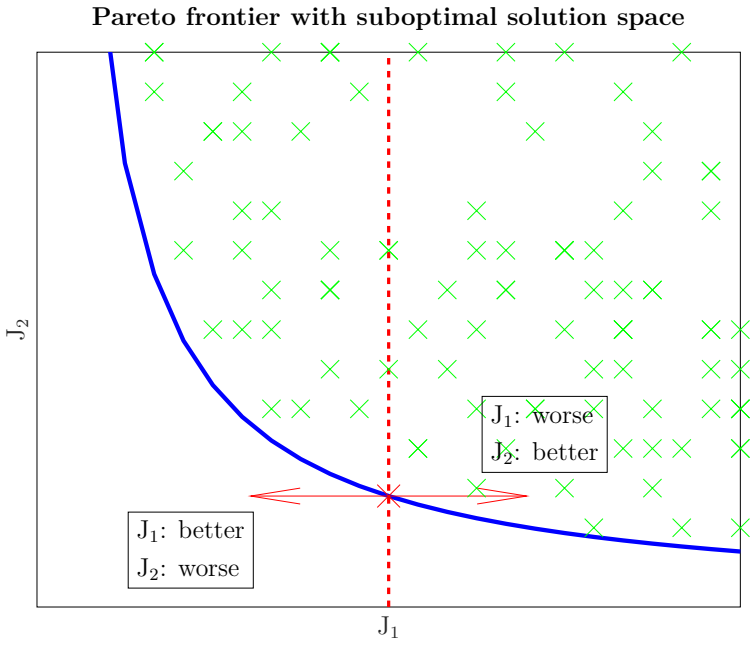

Figure 2.20: *Visualization of Pareto frontier (solid blue line) with two different, opposing cost function influences (each of the cost influences seeks to be minimal), the sub-optimal solution space above the Pareto frontier (green crosses), and infeasible region below Pareto frontier ("utopia points").*

which are not required in this thesis. Still, (2.77) provides the most common Pareto frontier approximation techniques that can be used to e.g., calculate the Pareto frontier in Figure 2.20.

2.4.2 Sensitivity Analysis

This section covers techniques for sensitivity analysis by means of post-optimal as well as integrator sensitivities.

Post-optimal Sensitivity

First of all, the post-optimal sensitivity analysis is introduced to analyze the optimal solution of an OCP, when invariant ("sensitive") parameters in the OCP are perturbed at this optimal point. It relies on the implicit function theorem (IFT) given as follows (using the required definitions for sensitivities; after [52, p. 28], [45], [23, p. 23]):

Theorem 2.6 (Implicit Function and Sensitivity Theorem)**.** *Let* $\mathbf{w} = \begin{bmatrix}\mathbf{z} & \boldsymbol{\mu} & \boldsymbol{\lambda}_{\mathbb{A}}\end{bmatrix}^{\mathsf{T}}$ *(here* $\boldsymbol{\lambda}_{\mathbb{A}}$ *are the* LAGRANGE *multipliers of the active inequality constraints) be implicitly given by the* $\mathcal{C}^1$ *function (i.e., it is continuously differentiable at least once) [5, p. 290]:*

$$\mathbf{R}(\mathbf{w};\mathbf{q}_0) = \mathbf{0}, \text{ with } \nabla_{\mathbf{w}}\mathbf{R}(\mathbf{w};\mathbf{q}_0) \text{ full rank}$$

A full rank is guaranteed if the constraint qualifications as well as sufficient optimality conditions hold. Then, for any $\mathbf{q}_0$ *("linearization point") there is a* $\mathcal{C}^1$ *function* $\boldsymbol{\xi}(\mathbf{q})$ *such that*

$$\mathbf{R}(\boldsymbol{\xi}(\mathbf{q});\mathbf{q}) = \mathbf{0}$$

holds in a neighborhood of $\mathbf{q}_0$*.*
That means it is $\mathbf{w}(\mathbf{q}) = \boldsymbol{\xi}(\mathbf{q})$ *around* $\mathbf{q}_0$*. Consequently, this means that* $\mathbf{w}(\mathbf{q})$ *is locally well defined and differentiable and that sensitivities can be calculated.*

From the IFT, a sensitivity equation can be derived by differentiating the implicit function with respect to the invariant (sensitive) parameters as follows:

$$\frac{\mathrm{d}\mathbf{R}(\mathbf{w};\mathbf{q})}{\mathrm{d}\mathbf{q}} = \underbrace{\frac{\partial\mathbf{R}(\mathbf{w};\mathbf{q})}{\partial\mathbf{w}}}_{\nabla_{\mathbf{w}}\mathbf{R}(\mathbf{w};\mathbf{q})}\frac{\partial\mathbf{w}}{\partial\mathbf{q}} + \frac{\partial\mathbf{R}(\mathbf{w};\mathbf{q})}{\partial\mathbf{q}} = \mathbf{0} \tag{2.78}$$

Solving (2.78) gives the sensitivities of the optimization parameters and constraint multipliers with respect to the sensitive parameters as follows:

$$\frac{\partial\mathbf{w}}{\partial\mathbf{q}} = -\left[\frac{\partial\mathbf{R}(\mathbf{w};\mathbf{q})}{\partial\mathbf{w}}\right]^{-1}\frac{\partial\mathbf{R}(\mathbf{w};\mathbf{q})}{\partial\mathbf{q}} \tag{2.79}$$

Thus, the remaining issue is to define the matrices in (2.79) for the OCP and check if they are invertible. In case they are, the equation can be solved for the sensitivities.

In the OC context, the post-optimal sensitivity equation is calculated by applying the IFT on the KKT conditions (Definition 2.1). This defines an implicit function, with only equality constraints for the sake of simplicity, as follows [23, p. 22f.], [45]:

$$\begin{aligned}\mathbf{R}(\mathbf{w};\mathbf{q}) &= \begin{bmatrix}\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z},\boldsymbol{\mu};\mathbf{q})\\ \boldsymbol{\psi}(\mathbf{z};\mathbf{q})\end{bmatrix} = \mathbf{0}\\ \frac{\partial\mathbf{R}(\mathbf{w};\mathbf{q})}{\partial\mathbf{w}} &= \begin{bmatrix}\mathbf{H}(\mathbf{z},\boldsymbol{\mu};\mathbf{q}) & \nabla_{\mathbf{z}}\boldsymbol{\psi}(\mathbf{z};\mathbf{q})^{\mathsf{T}}\\ \nabla_{\mathbf{z}}\boldsymbol{\psi}(\mathbf{z};\mathbf{q}) & \mathbf{0}\end{bmatrix}\end{aligned} \tag{2.80}$$

As the derivative of the implicit function in (2.80) is indeed the KKT matrix that is e.g., used to calculate the NEWTON step to update the OCP in (2.23), the matrix is invertible (assuming that the regularity conditions mentioned in Section 2.1 hold). Therefore, (2.79) provides the sensitivities of the OCP. The required properties of the KKT matrix and a more detailed derivation of the sensitivity equation for IPOPT can also be found in [106].

Take into account that $\mathbf{H}$ in (2.80) is the Hessian matrix of the OCP. Thus, a sensitivity analysis is only meaningful if the exact Hessian or a fairly good approximation is available and assessible (e.g., from a converged BFGS update and by directly using the factorized KKT matrix of the NLP from e.g., IPOPT [133]).

It should be further noted that the derivative of the implicit function, with respect to the sensitive parameters in (2.79), is also straightforward to calculate as follows:

$$\frac{\partial \mathbf{R}\left(\mathbf{w};\mathbf{q}\right)}{\partial \mathbf{q}} = \begin{bmatrix} \nabla_{\mathbf{z},\mathbf{q}} \mathcal{L}\left(\mathbf{z},\boldsymbol{\mu};\mathbf{q}\right) \\ \nabla_{\mathbf{q}} \boldsymbol{\psi}\left(\mathbf{z};\mathbf{q}\right)^{\mathsf{T}} \end{bmatrix} \tag{2.81}$$

The same derivation can as well be made for OCPs with inequality constraints. Here, a major part is the split up into the active and inactive constraints and considering the complementary condition. The derivation is therefore not done in this work and it is referred to the literature, where both the derivation as well as the assumptions are stated [23, p. 22f.], [45]. As a result, one gets the KKT matrix of the OCP with inequality constraints that is also invertible (assuming the fulfillment of the regularity and sufficient optimality conditions). Therefore, it is possible to calculate sensitivities with inequality constraints as well.

Integrator Sensitivity

In case (2.80) cannot be evaluated, because the Hessian is not available or only a simulation instead of an OCP is solved, a further option to calculate sensitivities is given by propagating an additional first order differential equation. It should be noted that this method does not result in the same sensitivities as for the post-optimal method, as e.g., the constraints are not considered.

For integrator sensitivities, the EoMs are generally derived with respect to the sensitive parameters and the chain rule is used [17, p. 69f.]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Bigg(\underbrace{\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{q}}}_{\mathbf{S}}\Bigg) = \underbrace{\frac{\mathrm{d}\dot{\mathbf{x}}\left(\mathbf{x};\mathbf{q}\right)}{\mathrm{d}\mathbf{q}}}_{\dot{\mathbf{S}}} = \underbrace{\frac{\partial \dot{\mathbf{x}}\left(\mathbf{x};\mathbf{q}\right)}{\partial \mathbf{x}}}_{\mathbf{J}_{\mathbf{x}}} \underbrace{\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{q}}}_{\mathbf{S}} + \underbrace{\frac{\partial \dot{\mathbf{x}}\left(\mathbf{x};\mathbf{q}\right)}{\partial \mathbf{q}}}_{\mathbf{J}_{\mathbf{q}}} \tag{2.82}$$

Here, $\mathbf{S}$ is the matrix that contains the desired (state) sensitivities. It should be noted that (2.82) is derived with respect to the states and sensitive parameters only for the sake of brevity. The case with controls and optimizable parameters is straightforward to calculate as well by suitably applying the chain rule.

Using (2.82), the sensitivity EoM system can be written as follows:

$$\begin{aligned} \dot{\mathbf{S}} &= \mathbf{J_x} \cdot \mathbf{S} + \mathbf{J_q} \\ \mathbf{S}\left(t_0\right) &= \mathbf{S}_0 \end{aligned} \tag{2.83}$$

Therefore, knowing the initial sensitivity of the system (which is e.g., zero with a fixed initial condition), it is possible to propagate the sensitivity based on the knowledge of the model Jacobian (e.g., from FALCON.m).

2.4.3 Bi-level Optimal Control

Bi-level OC is a form of OC that separates different objectives (e.g., as in Pareto problems) into two levels. These levels are normally organized in one upper level problem and multiple lower level problems. In this work, the lower level contains multiple OCPs, while the upper level contains a single parameter optimization problem, i.e., a problem with no state dynamics.

Bi-level (or more general: multi-level) OC is frequently used in technical applications. The following references provide an overview on the application areas: A general introduction to the topic is given in [10, 29, 145]. Aircraft related topics are dealt with in [46, 47].

Figure 2.21 depicts the general structure of a bi-level OCP: The upper level problem is typically a (small scale) parameter optimization problem, i.e., it does not have a dynamic model and only few constraints. The lower level is combined from multiple OCPs that are solved at the parameter set provided by the upper level. The lower level gives back the optimal trajectories to the upper level that are used to update the parameter values. Take into account that the upper level should also be a Newton-type optimization to be efficient. Therefore, at least a gradient must be provided by the lower levels, which is normally calculated using the already introduced (post-optimal) sensitivities (Subsection 2.4.2).

A general workflow for a bi-level optimization is also given in Algorithm 2.4. It is clear that multiple OCPs must be solved within each iteration of the upper level. This means that the OCPs must be implemented efficiently as well as parallelization techniques should be used. The formal bi-level optimization problem can then be stated as follows:

$$\begin{aligned} \min_{\mathbf{z}_{\mathrm{UL}}} \quad & J_{\mathrm{UL}}\left(\mathbf{z}_{\mathrm{UL}}; \mathbf{z}_{\mathrm{LL}}^{(i)}\right) \\ \text{s.t.} \quad & \mathbf{z}_{\mathrm{UL,lb}} \leq \mathbf{z}_{\mathrm{UL}} \leq \mathbf{z}_{\mathrm{UL,ub}}, \\ & \mathbf{c}(\mathbf{z}_{\mathrm{UL}}; \mathbf{z}_{\mathrm{LL}}^{(i)}) \leq \mathbf{0}, \\ & \boldsymbol{\psi}(\mathbf{z}_{\mathrm{UL}}; \mathbf{z}_{\mathrm{LL}}^{(i)}) = \mathbf{0} \\ & \text{with } \mathbf{z}_{\mathrm{LL}}^{(i)} = \arg\min \quad J_{\mathrm{LL}}^{(i)}\left(\mathbf{z}_{\mathrm{LL}}^{(i)}; \mathbf{z}_{\mathrm{UL}}\right), \qquad \forall i \\ & \qquad\qquad \text{s.t.} \quad \mathbf{z}_{\mathrm{LL,lb}}^{(i)} \leq \mathbf{z}_{\mathrm{LL}}^{(i)} \leq \mathbf{z}_{\mathrm{LL,ub}}^{(i)}, \\ & \qquad\qquad \mathbf{f}^{(i)}\left(\mathbf{z}^{(i)}; \boldsymbol{\theta}^{(i)}, \mathbf{z}_{\mathrm{UL}}\right) = \dot{\mathbf{x}}^{(i)}, \\ & \qquad\qquad \mathbf{c}(\mathbf{z}_{\mathrm{LL}}^{(i)}; \mathbf{z}_{\mathrm{UL}}) \leq \mathbf{0}, \\ & \qquad\qquad \boldsymbol{\psi}(\mathbf{z}_{\mathrm{LL}}^{(i)}; \mathbf{z}_{\mathrm{UL}}) = \mathbf{0} \end{aligned} \tag{2.84}$$

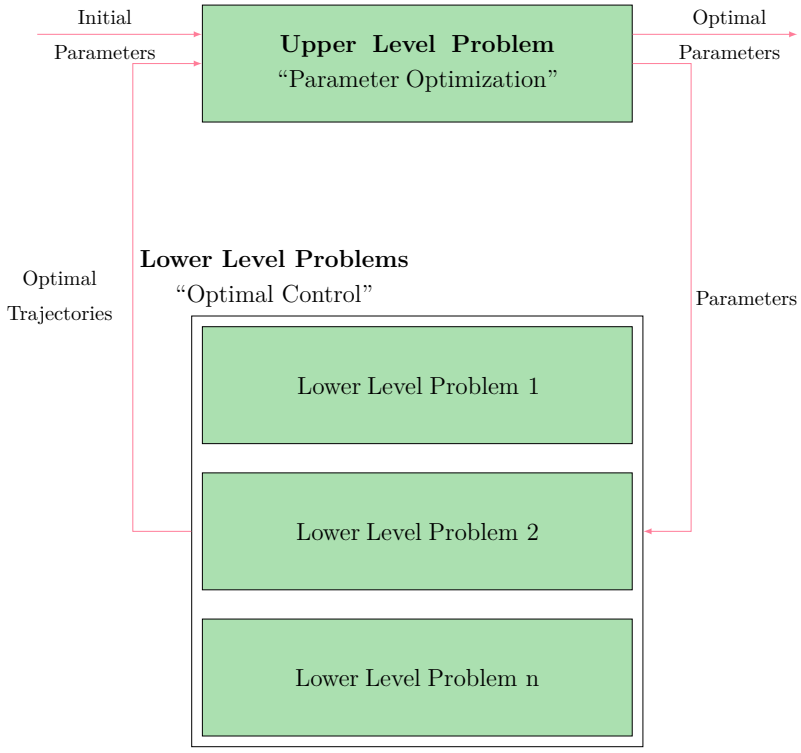

Figure 2.21: *General structure of a bi-level optimal control framework.*

Here, UL denotes quantities of the upper level and LL of the lower level. It is clear that the upper level assigns its optimization parameters to the lower level as additional non-optimizable parameters, which shape the optimal trajectories. In turn, the upper level is provided the lower level trajectories, as non-optimizable parameters again, and updated according to these. Thus, the cooperation character ("master-slave" or "leader-follower") of the bi-level OCP is clear, while it is also imminent that the solution process is quite time consuming due to the fact that, especially, the lower level OCPs must be solved multiple times in each iteration of the upper level to provide the new response.

Algorithm 2.4 Basic algorithm implemented in a bi-level optimal control framework.

Require:

Initial guess for bi-level parameters $\mathbf{z}_{\mathrm{UL}}$.

Initial guess for lower level optimization parameters $\mathbf{z}_{\mathrm{LL}}$.

1: **while** Upper level not converged **do**
2: **for all** i **do**
3: Solve OCP in (2.1) (e.g., FALCON.m) at each parameter set in parallelized manner, i.e., independent of each other, with current bi-level parameter values.
4: Calculate the (post-optimal) sensitivities (Subsection 2.4.2) of the OCPs with respect to the bi-level parameters.
5: **end for**
6: Calculate the upper level cost and constraints based on all lower level OCP solutions.
7: Update the bi-level parameters using an optimization algorithm for the cost and constraints applying e.g., the post-optimal sensitivities.
8: **end while**

9: **return** Optimal bi-level parameters $\mathbf{z}_{\mathrm{UL}}$ and optimal trajectories $\mathbf{z}_{\mathrm{LL}}^{(i)}$.

2.4.4 Chance-constrained Optimization

The usage of chance constraints (CCs) is frequently required in robust programming and optimization strategies [79, 130, 137]. The basic idea is to add a constraint to the general OCP formulation in (2.1) that takes the form of a probability as follows:

$$\mathbb{P}\left[\mathbf{y} \in \mathcal{P}\right] \geq \xi \tag{2.85}$$

Here, $\mathbb{P}$ is the probability operator (i.e., the probability of the condition in brackets is calculated) and $\mathcal{P}$ is the domain in which the probability of the CC must be fulfilled with the confidence/probability level ξ. It should be noted that CCs are generally defined with respect to the system outputs in this thesis as this includes the optimization parameters as well as dependent values according to (2.12). Thus, using the outputs provides a more general statement than, for instance, just using the optimization parameters.

Take into account that (2.85) can also be described in the context of a failure event $\mathcal{F}$ (e.g., required for the SubSim in Subsection 2.2.4) with the failure probability p_f as follows:

$$\mathbb{P}\left[\mathbf{y} \in \mathcal{F}\right] = 1 - \mathbb{P}\left[\mathbf{y} \in \mathcal{P}\right] = p_f \leq 1 - \xi \tag{2.86}$$

Due to the fact that (2.85) normally provides a better scaled OCP than (2.86) (in terms of the magnitude of the values), the formulation in (2.85) is used in the following.

For chance-constrained open-loop direct optimal control (CC-OC), the standard OCP formulation in (2.1) is thus adapted to:

$$\begin{aligned} \min_{\mathbf{z}} \quad & J(\mathbf{z};\mathbf{q}) \\ \text{s.t.} \quad & \mathbf{z}_{\text{lb}} \leq \mathbf{z} \leq \mathbf{z}_{\text{ub}}, \\ & \mathbf{f}(\mathbf{z};\boldsymbol{\theta},\mathbf{q}) = \dot{\mathbf{x}}, \\ & \mathbf{c}(\mathbf{z};\mathbf{q}) \leq \mathbf{0}, \\ & \boldsymbol{\psi}(\mathbf{z};\mathbf{q}) = \mathbf{0}, \\ & \mathbb{P}\left[\mathbf{y} \in \mathcal{P}\right] \geq \xi \end{aligned} \tag{2.87}$$

It should be noted that it is deliberately distinguished between probabilistic and deterministic constraints in (2.87). This is due to e.g., the fact that the state integration is generally carried out in the deterministic domain (in our case, the specific evaluation nodes provided by the gPC-SC theory are used), while there are also constraints that are specifically designed in the probabilistic domain (i.e., the CCs).

It is clear that (2.85) and (2.87) are largely depending on the functional dependence for the output PDF that is normally not assessable directly using e.g., the gPC method. Thus, the probability in (2.85) must either be approximated by probability inequalities, as given in Appendix B.8, or the PDF must be approximated as a whole using e.g., the maximum entropy principle [113]. Additionally, the probability can be approximated by sampling, e.g., MCA.

General methods to solve the chance-constrained open-loop direct optimal control problem (CC-OCP) are normally based on sampling techniques that try to approximate the probability in (2.85): At first, the already introduced MCA is a very simple method (Subsection 2.2.1). Further, more sophisticated methods include e.g., importance sampling [22] like the MHA [112, 118]. In order to use these methods, normally some additional information on the system, e.g., a general knowledge on the shape of the output PDF, must be applied. Therefore, the approach used in this thesis to approximate CCs is by directly applying the gPC expansion (see (2.52)) as a sampling alternative. This approach of sampling can be mainly used if the probability of failure occurrence is still fairly frequent (e.g., 1%).

In cases, where a rare event (e.g., $10^{-6}\%$) should be optimized, different methods must be applied, as the standard sampling methods cannot handle these. One example of these specialized methods is the SubSim (Subsection 2.2.4) that uses a MCMC MMHA relying on conditional probability. Within this approach, the MCMC basically moves from frequent event probabilities to rare-event probabilities by a chain and calculates the desired rare-event probability based on the conditional probability of this chain. The necessary sampling can again be done using the gPC expansion.

The general method developed in this thesis to incorporate CCs in OCPs, using the gPC expansion, is introduced in Chapter 6 and based on the formulation in (2.87).

2.4.5 Distributed Optimal Control

A fairly well-known and often used method in OC, and here especially in model predictive control (MPC), is distributed open-loop direct optimal control (DOC). DOC is based on splitting up a large-scale OCP into smaller OCPs that are easier to solve. These smaller OCPs are solved independently of each other. For this, a small set of connection variables must be introduced to ensure that the original large-scale OCP is still satisfied and solved. This is also depicted in Figure 2.22: Here, the upper level provides the connection variables $\boldsymbol{\nu}$ to the lower levels, while the lower level OCPs provide their optimal results to the upper level. Again, (post-optimal) sensitivities (Subsection 2.4.2) can be used to update the NEWTON-type NLP of the upper level.

Formally, DOC can directly be derived from the KKT conditions in Definition 2.1 [78]. A derivation of this, to prove the viability of the distributed open-loop direct optimal control problem (DOCP) of this thesis, is also done in Section 4.4. From a methodological point of view, the implementation and solution of DOCPs is very similar to the bi-level OCP methods of Subsection 2.4.3. Overall, the implementation of the robust DOC framework using gPC is introduced in Chapter 4, while the general methodology can be described as follows (similar to (2.84)):

$$
\begin{aligned}
\min_{\boldsymbol{\nu}} \quad & J_{\mathrm{CP}}\left(\boldsymbol{\nu}; \mathbf{z}^{(j)}\right) \\
\text{s.t.} \quad & \boldsymbol{\nu}_{\mathrm{lb}} \leq \boldsymbol{\nu} \leq \boldsymbol{\nu}_{\mathrm{ub}}, \\
& \mathbf{c}_{\mathrm{CP}}(\boldsymbol{\nu}; \mathbf{z}^{(j)}) \leq \mathbf{0}, \\
& \boldsymbol{\psi}_{\mathrm{CP}}(\boldsymbol{\nu}; \mathbf{z}^{(j)}) = \mathbf{0} \\
& \text{with } \mathbf{z}^{(j)} = \arg\min \quad J^{(j)}\left(\mathbf{z}^{(j)}; \boldsymbol{\nu}\right), \qquad \forall j \\
& \qquad\qquad \text{s.t.} \quad \mathbf{z}^{(j)}_{\mathrm{LL,lb}} \leq \mathbf{z}^{(j)} \leq \mathbf{z}^{(j)}_{\mathrm{LL,ub}}, \\
& \qquad\qquad\qquad \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \boldsymbol{\nu}\right) = \dot{\mathbf{x}}^{(j)}, \\
& \qquad\qquad\qquad \mathbf{c}(\mathbf{z}^{(j)}; \boldsymbol{\nu}) \leq \mathbf{0}, \\
& \qquad\qquad\qquad \boldsymbol{\psi}(\mathbf{z}^{(j)}; \boldsymbol{\nu}) = \mathbf{0}
\end{aligned}
\tag{2.88}
$$

Here, $\boldsymbol{\nu}$ are the connection variables and the index CP denotes the connection problem. It is clear that the DOCP is very similar to the bi-level OC framework in the way that there is parameter exchange between the lower, distributed levels and the upper, connection level. The main difference is the way how the upper level cost function is formulated (generally, the connection error should be reduced) and that the lower levels must be split up, i.e., distributed. This has to be done in a very specific way to recreate the originally desired OCP and is introduced in detail in Chapter 4 in the context of ROC with gPC. It is important to state that, in general, DOC makes no simplifying assumptions and thus, solves the original large-scale OCP.

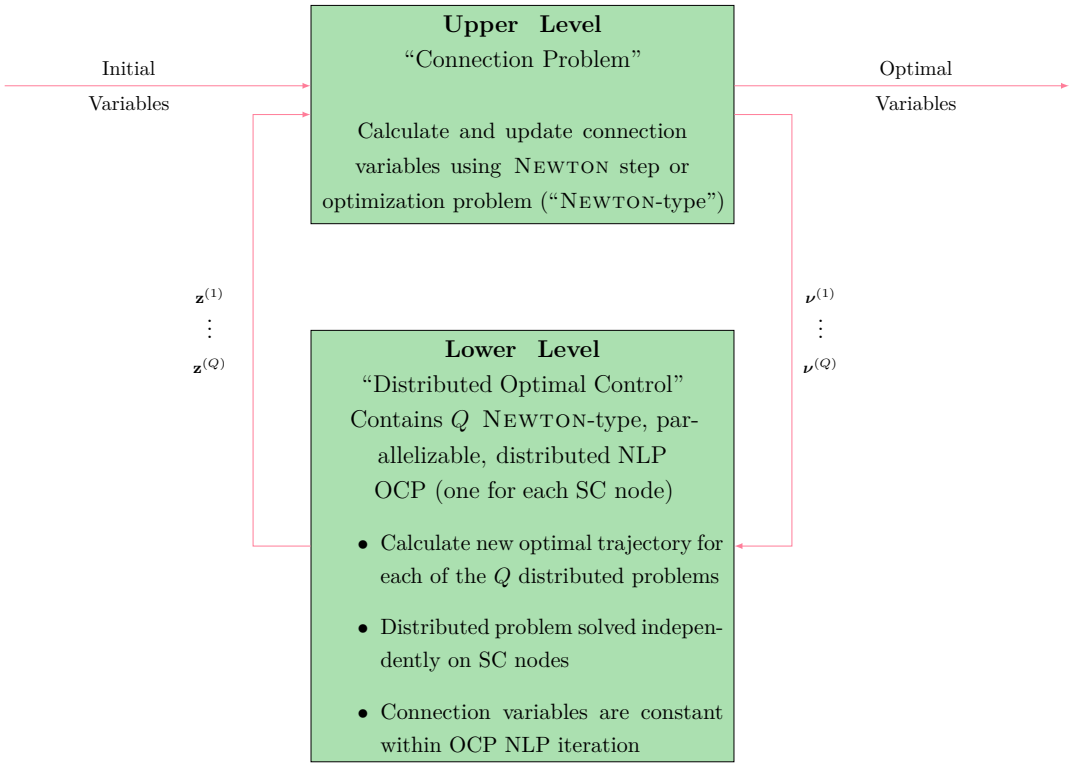

Figure 2.22: *General structure of distributed optimal control problem framework with connection problem in upper level, exchange of parameters, and distributed optimal control in lower level.*

Chapter 3

Robust Bi-level Optimal Control

This chapter introduces the developed robust bi-level open-loop direct optimal control (OC) framework that is related to Contribution 2 of this thesis. Here, Section 3.1 gives an overview on the bi-level approach, using the generalized polynomial chaos-stochastic collocation framework (gPC-SC) (Section 2.3) in the lower level OC nonlinear programs (NLPs) and a differential evolution algorithm (DEA) (a genetic algorithm) in the upper level problem. Then, Section 3.2 introduces the use of sensitivities within the gPC-SC bi-level approach to speed up the solution process, especially within the upper level problem. It should be noted that a similar bi-level structure was already introduced in [109], while results by the author of this work with similar structures were already published in [104, 105]. This work extends on the previous studies by introducing constraints, especially chance constraints (CCs), and giving a clearer picture on the combination of sensitivities and generalized polynomial chaos (gPC).

3.1 Bi-level Framework using Generalized Polynomial Chaos and Stochastic Collocation

This section introduces the bi-level OC framework used in this work to calculate robust optimal results e.g., for the controller gain design example in Chapter 7. The general structure of the framework is illustrated in Figure 3.1: It is depicted that the lower level uses the gPC formulation and the SC approach to calculate statistical moments of the optimal trajectories. Here, NEWTON-type OC techniques are used to calculate the non-robust, optimal trajectories (Subsection 2.1.2.1). This structure was initially proposed by the author in [104, 105].

The resulting trajectory representation by statistical moments is then exchanged with the upper level that in turn tries to optimize the uncertainty influence on the trajectory to e.g., be minimal (i.e., minimizing the standard deviation of a quantity). For this, an optimization based on a DEA is proposed in this section. The upper level ultimately assigns parameters, $\mathbf{q}_1, \ldots, \mathbf{q}_{n_{\text{pop}}}$ (e.g., control authority factors or controller gain parameters),

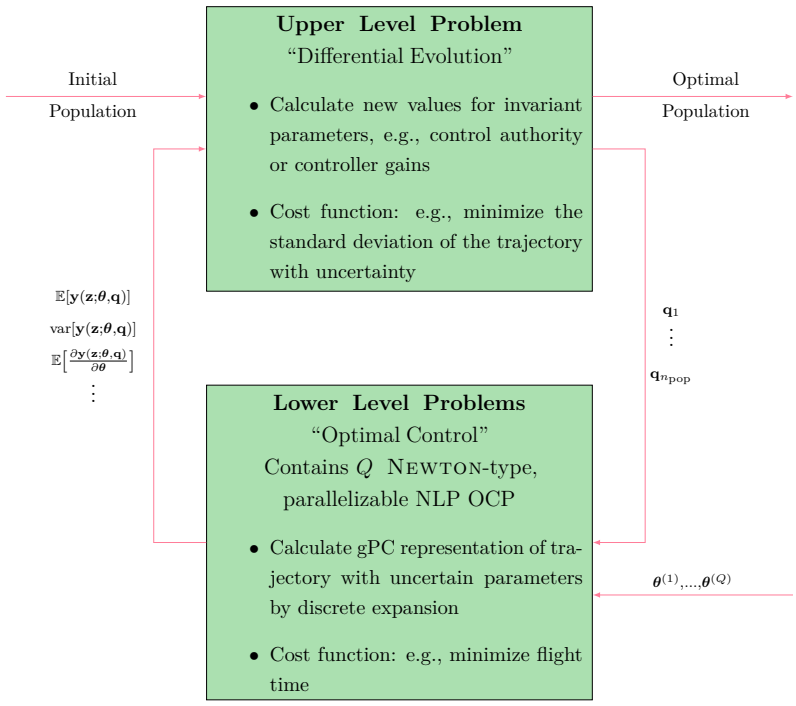

Figure 3.1: *Structure of bi-level optimal control problem with lower level generalized polynomial chaos expansion calculation by optimal control using discrete expansion and upper level parameter optimization.*

denoting the current population of the DEA, to the lower levels that shape the trajectory by influencing e.g., the state dynamics and thus, changing the statistical moments of the optimal trajectory. It should be noted that these parameters can either be constant along the trajectory, i.e., a trade-off for the whole trajectory is sought or they can vary over time, i.e., an adaptation for different trajectory states is possible.

The overall optimization procedure can be summarized as follows: The upper level problem optimizes the parameters $\mathbf{q}_1, \ldots, \mathbf{q}_{n_{\mathrm{pop}}}$ that are e.g., control effectiveness parameters, and are able to shape the trajectory (e.g., by influencing the state dynamics). These parameters are assigned to the lower level open-loop direct optimal control problems (OCPs) that are evaluated with these upper level optimization parameters at the stochastic collocation (SC) nodes to create the gPC expansion. This yields the response of the aircraft to uncertain parameters. Finally, this response is used to calculate the statistical moments of the uncertain trajectory (e.g., mean and standard deviation) that are then again provided to the upper level problem and its cost function. This cost function can be e.g., the minimization of the standard deviation of the optimal trajectory. Overall, this procedure leads to a robust trajectory, as the optimized trajectory is subject to a reduced uncertainty influence based on the upper level cost function and the optimal invariant parameters.

In the following, a detailed overview of the structural parts of the bi-level framework depicted in Figure 3.1 is given:

Lower Level Problem Formulation The lower level OCP in the bi-level OC framework is relying on NEWTON-type OC. The problem formulation is given as in (2.1), discretized as in Subsection 2.1.2.1, and optimized using a NEWTON-type optimization strategy from Subsection 2.1.3 or 2.1.4.

The optimization model is based on the equation of motion (EoM) representation introduced in the examples' chapters. The lower level OCP is e.g., optimized with respect to a minimal flight time, i.e., $J_{\mathrm{LL}} = t_f$ (LL denotes the lower level), in a deterministic way and evaluated according to the gPC expansion (Section 2.3) using the SC method (Subsection 2.3.3). The statistical moments are calculated according to Subsection 2.3.4.

Thus, the lower level OCP provides a non-robust, optimal response surface with respect to the defined uncertainties to the upper level. This response surface is calculated by the gPC-SC methodology.

Upper Level Problem Formulation The upper level problem is a parameter optimization problem with no dynamic model and only few additional constraints. Initially, it uses a DEA strategy [44] to optimize the statistics of the lower level OCPs, by applying an own upper level cost function J_{UL} (UL denotes the upper level). Take into account that the upper level optimization parameter vector is constructed as: $\mathbf{z}_{\mathrm{UL}} = \begin{bmatrix} \mathbf{q}_1^{\mathsf{T}} & \dots & \mathbf{q}_{n_{\mathrm{pop}}}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$. The upper level provides these parameters, which are able to influence the optimal trajectory, to the lower levels and therefore, robustifies the solution.

The structure of the implemented DEA, also shown in Figure 3.2, is introduced in the following paragraph:

Differential Evolution Algorithm This algorithm is part of the broader range of genetic algorithms that can be used in bi-level OC [145]. The general procedure of the DEA is visualized in Figure 3.2 and described in Algorithm 3.1. Compared to standard NEWTON-type OC algorithms (e.g., Subsections 2.1.3 or 2.1.4), the DEA does not rely on the gradient to solve the optimization problem. It rather is a search algorithm within the solution space and tries to find the best overall cost. Therefore, the algorithm normally converges globally compared to the only local results of NEWTON-type algorithms. As a drawback, the DEA is generally very slow in convergence and requires a population n_{pop} of solution points that are adapted throughout the optimization (see e.g., Step 1 in Algorithm 3.1). This means that significant computational overhead is introduced, as the solution of multiple OCPs at optimization parameters, which are later on discarded because they are not optimal, is required. Additionally, the introduction and fulfillment of constraints is not trivial as e.g., a choice for a new point normally does not fulfill these (Steps 9–11 in Algorithm 3.1).

Because of this, the DEA is only used within this thesis for small parameter optimization problems that are present in the upper level of the bi-level OCP (Subsection 2.4.3). Here, the benefits of a global optimal solution, which is possible to achieve by the DEA [44,

Algorithm 3.1 Description of differential evolution algorithm.

Require:

Population size: n_{pop}

Backtrack steps: n_{bt}

Maximum number of iterations $k_{\max}$ and optimality ϵ_{opt} as well as feasibility ϵ_{feas} tolerance

Define cost function J, bounds on optimization parameters $\mathbf{z}_{\text{lb}}$ and $\mathbf{z}_{\text{ub}}$ as well as nonlinear constraint functions $\mathbf{c}$ and $\boldsymbol{\psi}$.

Define initial population: $\mathbf{z}_{\text{init}} \in \mathbb{R}^{n_z \times n_{\text{pop}}}$ (e.g., by random sampling in optimization parameter bounds).

Calculate initial cost $\mathbf{J}_0$ and constraint values $\mathbf{c}_0, \boldsymbol{\psi}_0$ for all population members.

Punish population members that do not fulfill the desired constraints for $\mathbf{c}_0$ and $\boldsymbol{\psi}_0$ (e.g., logarithmic barrier to increase cost function).

Get the best cost function, i.e., the lowest cost value: $J_{0,\text{best}} = \min \mathbf{J}_0$

Set the counter and the initial tolerance: $k = 1$, $\epsilon_{\text{DE,opt}} > \epsilon_{\text{opt}}$, $\epsilon_{\text{DE,feas}} > \epsilon_{\text{feas}}$

while $k \leq k_{\max}$ & $\epsilon_{\text{DE,opt}} > \epsilon_{\text{opt}}$ & $\epsilon_{\text{DE,feas}} > \epsilon_{\text{feas}}$ **do**

  Mutate the previous population $\mathbf{z}_{k-1}$ to create a new population $\mathbf{z}_{k,\text{donor}}$ by random sampling of new population members ("donor vector").

  Ensure that the donor vector $\mathbf{z}_{k,\text{donor}}$ fulfills the constraints $\mathbf{z}_{\text{lb}}$ and $\mathbf{z}_{\text{ub}}$.

  Crossover of previous population $\mathbf{z}_{k-1}$ and donor population $\mathbf{z}_{k,\text{donor}}$ by binomial crossover (i.e., random selection of parameters below "crossover" margin) to create $\mathbf{z}_{k,\text{trial}}$ ("trial vector").

  Calculate trial cost $\mathbf{J}_{k,\text{trial}}$ and constraints $\mathbf{c}_{k,\text{trial}}$ and $\boldsymbol{\psi}_{k,\text{trial}}$ for all trial population members $\mathbf{z}_{k,\text{trial}}$.

  Punish trial population members that do not fulfill the desired constraints for $\mathbf{c}_{k,\text{trial}}$ and $\boldsymbol{\psi}_{k,\text{trial}}$ (e.g., by logarithmic barrier to increase cost function).

  Select the new population members $\mathbf{z}_k$ by a tournament selection of the previous population $\mathbf{z}_{k-1}$ and trial population $\mathbf{z}_{k,\text{trial}}$, i.e., by selecting the n_{pop} samples corresponding to the n_{pop} best cost functions of both previous and trial population vector.

  Calculate current cost $\mathbf{J}_k$ and constraints $\mathbf{c}_k, \boldsymbol{\psi}_k$ for all population members $\mathbf{z}_k$.

  Get the best cost function, i.e., the lowest cost: $J_{k,\text{best}} = \min \mathbf{J}_k$

  Calculate current optimality: $\epsilon_{\text{DE,opt}} = ||J_{k,\text{best}} - J_{k-n_{\text{bt}}+1,\text{best}}||_\infty$

  Calculate current feasibility: $\epsilon_{\text{DE,feas}} = \max\{||\mathbf{c}_k||_\infty, ||\boldsymbol{\psi}_k||_\infty\}$

  Increase counter: $k = k + 1$

end while

---

return Best cost function $J_{k-1,\text{best}}$ with corresponding best population member $\mathbf{z}_{\text{opt}}$.

p. 1f.], are desired. Additionally, the convergence speed in the bi-level context is normally dependent on the convergence speed of the lower level OCPs rather than the upper level. Furthermore, the computation of sensitivities (Subsection 2.4.2) for uncertain trajectories to update the upper level can be more complicated and not trivial for OCPs, whose (analytic) Hessian is costly to evaluate and non-smooth (e.g., in cases where large look-up tables are used in the dynamic model). Therefore, the DEA is of benefit for the convergence in such cases and might even just enable them.

In addition, the DEA can be used to restrict the original search space to the area of the global optimum. Subsequently, a NEWTON-type NLP technique with sensitivities can be applied to find the exact optimum. This is described in Section 3.2.

Finally, the optimization problem for the bi-level structure can be formulated as follows (after (2.84)):

$$
\begin{aligned}
\min_{\mathbf{z}_{\mathrm{UL}}} \quad & J_{\mathrm{UL}}\left(\mathbf{z}_{\mathrm{UL}}; \mathbf{z}_{\mathrm{LL}}^{(j)}\right) \\
\text{s.t.} \quad & \mathbf{z}_{\mathrm{UL,lb}} \leq \mathbf{z}_{\mathrm{UL}} \leq \mathbf{z}_{\mathrm{UL,ub}}, \\
& \mathbf{c}(\mathbf{z}_{\mathrm{UL}}) \leq \mathbf{0}, \\
& \boldsymbol{\psi}(\mathbf{z}_{\mathrm{UL}}) = \mathbf{0}, \\
& \mathbb{P}\left[\mathbf{y}_{\mathrm{UL}} \in \mathcal{P}\right] \geq \boldsymbol{\xi}_{\mathrm{UL}} \\
& \text{with } \mathbf{z}_{\mathrm{LL}}^{(j)} = \arg\min \quad J_{\mathrm{LL}}^{(j)}\left(\mathbf{z}_{\mathrm{LL}}^{(j)}; \mathbf{z}_{\mathrm{UL}}\right), \qquad \forall j \\
& \qquad\qquad \text{s.t.} \quad \mathbf{z}_{\mathrm{LL,lb}}^{(j)} \leq \mathbf{z}_{\mathrm{LL}}^{(j)} \leq \mathbf{z}_{\mathrm{LL,ub}}^{(j)}, \\
& \qquad\qquad \quad \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{z}_{\mathrm{UL}}\right) = \dot{\mathbf{x}}^{(j)}, \\
& \qquad\qquad \quad \mathbf{c}(\mathbf{z}_{\mathrm{LL}}^{(j)}; \mathbf{z}_{\mathrm{UL}}) \leq \mathbf{0}, \\
& \qquad\qquad \quad \boldsymbol{\psi}(\mathbf{z}_{\mathrm{LL}}^{(j)}; \mathbf{z}_{\mathrm{UL}}) \leq \mathbf{0}, \\
& \qquad\qquad \quad \mathbb{P}\left[\mathbf{y}_{\mathrm{LL}}^{(j)} \in \mathcal{P}\right] \geq \boldsymbol{\xi}_{\mathrm{LL}}^{(j)}
\end{aligned}
\tag{3.1}
$$

Problem (3.1) is the basic formulation of a bi-level NLP and is used for both the DEA-based framework of this subsection as well as the sensitivity-based framework introduced in the next subsection. It should be noted that (3.1) directly includes CCs (Subsection 2.4.4) that can be used to robustify the bi-level OC solution by constraints rather than in the cost function.

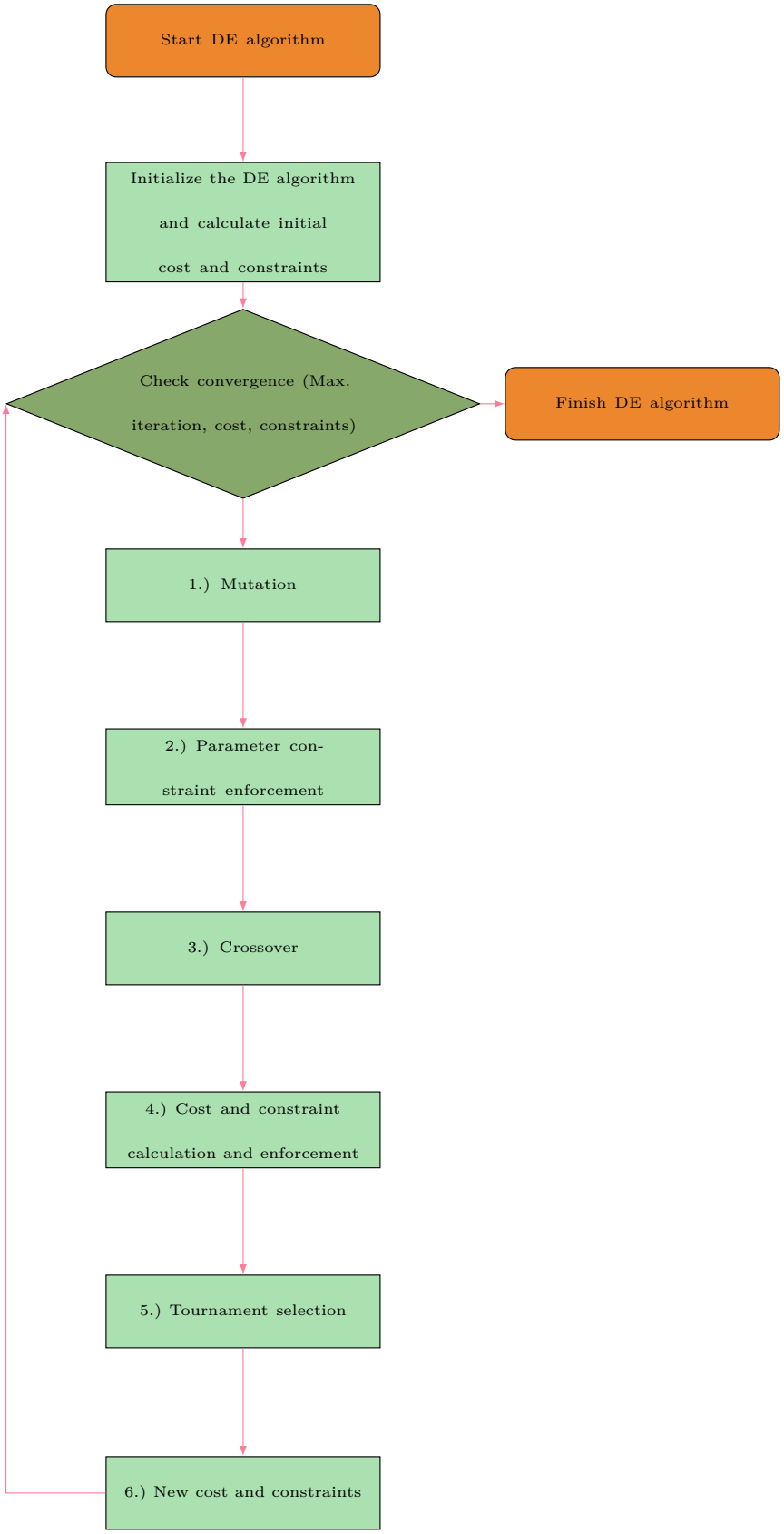

Figure 3.2: *Flowchart of differential evolution algorithm used in this thesis.*

3.2 Bi-level Framework using Generalized Polynomial Chaos Statistics and Stochastic Collocation with Sensitivity Update

This section introduces a bi-level framework with improved efficiency compared to Section 3.1 that is connected by the exchange of the statistics of (post-optimal or simulated) sensitivities (Subsection 2.4.2). The sensitivities give the possibility to use gradient information within the upper level optimization algorithm and thus, solve the bi-level OCP in (3.1) more efficiently. This leads to a faster convergence than with the DEA approach in Section 3.1. The structure of the bi-level framework with sensitivities is illustrated in Figure 3.3.

The main development of the proposed approach is that by using the sensitivities not only the mean or standard deviation values of a quantity are provided to the upper level, but also the derivative of the mean and the standard deviation with respect to the optimization parameters. These values are calculated as follows: At each of the SC nodes the (post-optimal or simulated) sensitivities are calculated as given in (2.79) or (2.83). For the optimization states, controls, or further outputs, the corresponding expansion coefficients can be calculated using the SC in (2.68). Consequently, the statistics for the sensitivities are also given as described in Subsection 2.3.4 using the SC. Thus, using sensitivities is very convenient from an implementation point of view, as only already available information is used to calculate the sensitivities and afterward, the already implemented gPC-SC can be used to evaluate the statistics. Take into account that, as it is generally difficult to calculate a second order sensitivity (i.e., the Hessian), the upper level normally only runs the NLP solution only with the Jacobian and gets the Hessian by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update.

It should be noted that the bi-level OCP remains as defined in (3.1) with the upper level optimization parameters now given by $\mathbf{z}_{\mathrm{UL}} = \mathbf{q}$. Thus, no population but only a single parameter set is required. As now a NEWTON-type NLP scheme is also applied in the upper level, the chain rule for the statistical moments of the sensitivity must be applied. Assuming a mean value for the cost function in the upper level, the gradient can be calculated as follows:

$$\begin{aligned}\mathbb{E}\left[\frac{\mathrm{d}J_{\mathrm{UL}}}{\mathrm{d}\mathbf{z}_{\mathrm{UL}}}\right] &= \mathbb{E}\left[\frac{\partial J_{\mathrm{UL}}}{\partial \mathbf{z}_{\mathrm{UL}}} + \frac{\partial J_{\mathrm{UL}}}{\partial \mathbf{z}_{\mathrm{LL}}} \cdot \frac{\partial \mathbf{z}_{\mathrm{LL}}}{\partial \mathbf{z}_{\mathrm{UL}}}\right] \\ &= \mathbb{E}\left[\frac{\partial J_{\mathrm{UL}}}{\partial \mathbf{z}_{\mathrm{UL}}}\right] + \mathbb{E}\left[\frac{\partial J_{\mathrm{UL}}}{\partial \mathbf{z}_{\mathrm{LL}}}\right] \cdot \mathbb{E}\left[\frac{\partial \mathbf{z}_{\mathrm{LL}}}{\partial \mathbf{z}_{\mathrm{UL}}}\right] + \operatorname{cov}\left[\frac{\partial J_{\mathrm{UL}}}{\partial \mathbf{z}_{\mathrm{LL}}}, \frac{\partial \mathbf{z}_{\mathrm{LL}}}{\partial \mathbf{z}_{\mathrm{UL}}}\right]\end{aligned} \tag{3.2}$$

Here, all mean values (see (2.69)) as well as the covariance (cov; see (D.3)) can be directly calculated using the gPC expansion coefficients for the sensitivity. Thus, (3.2) is applicable in the context of bi-level OC. Take into account that for most applications, the derivative of the cost function with respect to the upper level parameters is zero.

Take into account that (3.2) can be extended to other statistical moments as well by calculating suitable higher order joint statistical moments. For instance, the co-kurtosis is required for the standard deviation. As seen in Appendix D, the calculation of higher order moments for the gPC expansion becomes non-trivial. Thus, the best option in these cases is to sample the required higher order statistical moments using the gPC expansion.

It should be further noted that the approach in (3.2) is different from the iterative approaches in e.g., [139, p. 497] or [140, p. 255], which calculate the sensitivity based on deriving the gPC response surface directly. Thus, they do not consider constraints that change the sensitivity. These are considered in the proposed approach, because gPC is used to calculate a response surface for the (post-optimal) sensitivity that considers the constraint.

It is once more reminded that the upper level parameters are invariant parameters in the lower level. Therefore, the second part of the product in the second line of (3.2) is easily calculated using the (post-optimal or simulated) sensitivities in Subsection 2.4.2. The statistics are then available using the gPC-SC method. The first part of the product in the second line of (3.2) is normally readily available by calculating the analytic derivative of the upper level cost function. Finally, the covariance is given by (D.3) for gPC and therefore, also directly available. Thus, the bi-level OCP can be solved efficiently using the gPC-SC methodology and (post-optimal) sensitivities. Take into account that the equations for constraint gradients in the upper level can be derived analogous to (3.2).

Summarizing, the frameworks introduced in Sections 3.1 and 3.2 provide one possible way of calculating robust trajectories. Although easy to implement and based on deterministic OCP evaluation, a drawback is the necessity to introduce artificial parameters that are able to shape the OC trajectories. These are not trivial to define and different choices might lead to different robust, optimal solutions. Additionally, the proposed method is computationally high-demanding as the upper level requires the solution of multiple full OCPs in each iteration. Thus, the proposed framework of the previous sections is evolved in the following to e.g., increase the efficiency.

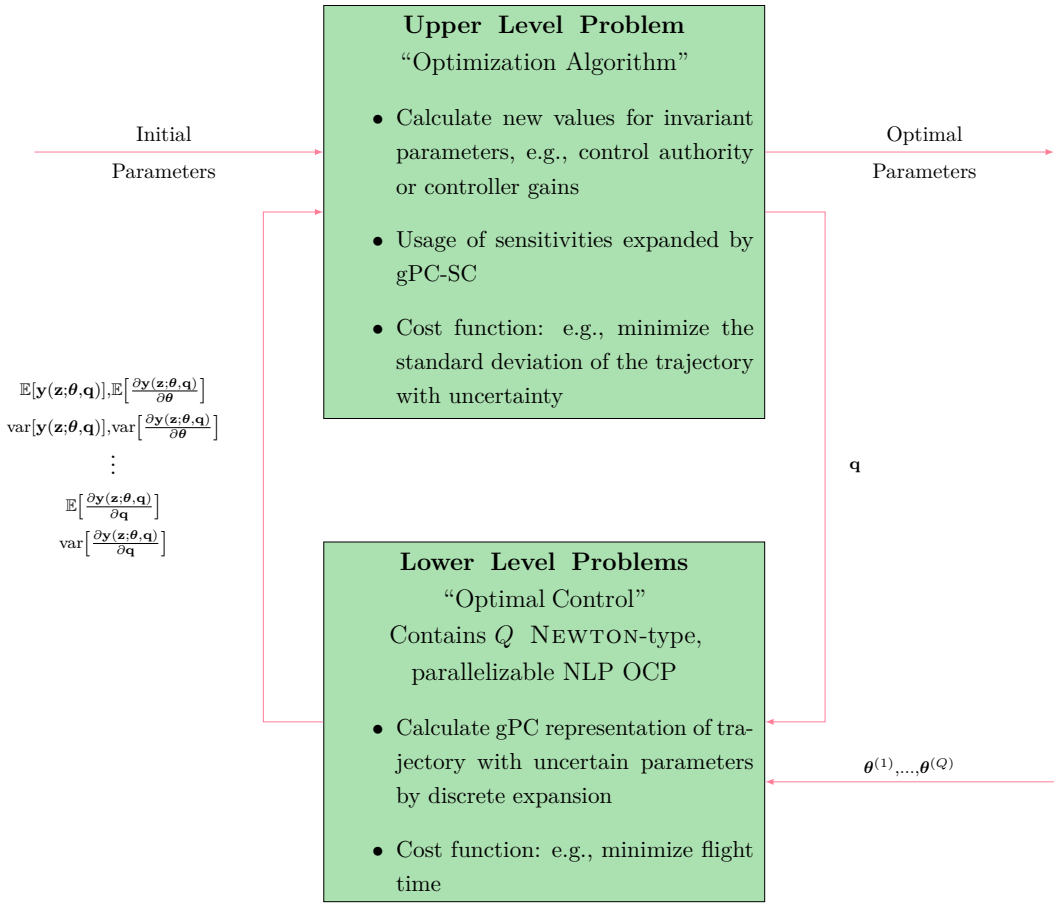

Figure 3.3: *Structure of cooperative bi-level optimal control problem with lower level generalized polynomial chaos expansion calculation and upper level parameter optimization using sensitivity updates.*

Chapter 4

Distributed Optimal Control with Generalized Polynomial Chaos

This chapter introduces the developed distributed open-loop direct optimal control (DOC) framework of Contribution 3 in this thesis. Within this framework the standard, large-scale open-loop direct optimal control problem (OCP) that arises when using a generalized polynomial chaos (gPC) transcription with stochastic collocation (SC) on the deterministic OCP is reduced to multiple smaller OCPs that are then evaluated in a distributed way. The framework is based on the standard gPC expansion with SC, introduced in Section 2.3, and the basic distributed open-loop direct optimal control problem (DOCP), introduced in Subsection 2.4.5. Here, the general idea of the method is similar to the bi-level framework in Chapter 3 with a specific mathematical derivation of the upper level optimization parameters.

It should be noted that the connection of DOC and gPC was already studied in e.g., [32, 37, 67]. The main difference of the studies compared to the thesis at hand is the fact that the studies rely on intrusive changes to the OCP by the gPC expansion. This thesis, on the other hand, uses the SC method and thus, conserves the deterministic baseline structure of the OCP. The initial idea of this framework was published by the author in [101]. Within this thesis, the published method is extended by a proof of convergence based on the Karush-Kuhn-Tucker (KKT) conditions as well as a distributed solution of constraints, e.g., in the context of chance constraints (CCs).

4.1 Distributed Optimal Control

Generally, Chapter 3 already introduced a robust open-loop direct optimal control (ROC) framework, based on bi-level open-loop direct optimal control (OC), that can be used to calculate robust, optimal trajectories. A drawback of this method was the fact that artificial parameters had to be introduced in the OC procedure that shape a robust trajectory and are assigned by the upper level. Although the introduction of these parameters is

generally always possible, it might be difficult to find meaningful parameters. Additionally, the introduction of artificial parameters might constrain the robust open-loop direct optimal control problem (ROCP) inherently and thus, different parameter sets can lead to different robust, optimal solutions.

To mitigate the issues arising with the formulation in Chapter 3, this chapter introduces a direct formulation of the ROCP, which is solved by distributing a number of unconnected OC sub-problems based on the gPC expansion. The general goal is to solve the following large-scale OCP in the context of the generalized polynomial chaos-stochastic collocation framework (gPC-SC) for ROC:

$$
\begin{aligned}
\min_{\bar{\mathbf{z}}} \quad & J\left(\bar{\mathbf{z}};\mathbf{q}\right) \\
\text{s.t.} \quad & \bar{\mathbf{z}}_{\text{lb}} \leq \bar{\mathbf{z}} \leq \bar{\mathbf{z}}_{\text{ub}}, \\
& \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right) = \dot{\mathbf{x}}^{(j)}, \quad j=1,\ldots,Q, \\
& \mathbf{c}^{(j)}\left(\mathbf{z}^{(j)};\mathbf{q}\right) \leq \mathbf{0}, \quad j=1,\ldots,Q, \\
& \boldsymbol{\psi}^{(j)}\left(\mathbf{z}^{(j)};\mathbf{q}\right) = \mathbf{0}, \quad j=1,\ldots,Q, \\
& \bar{\mathbf{c}}\left(\bar{\mathbf{z}};\mathbf{q}\right) \leq \mathbf{0}, \\
& \bar{\boldsymbol{\psi}}\left(\bar{\mathbf{z}};\mathbf{q}\right) = \mathbf{0}, \\
& \mathbb{P}\left[\bar{\mathbf{y}} \in \mathcal{P}\right] \geq \boldsymbol{\xi}
\end{aligned}
\tag{4.1}
$$

Here, $\bar{\mathbf{z}} = \left[\left[\mathbf{z}^{(1)}\right]^{\mathsf{T}} \quad \ldots \quad \left[\mathbf{z}^{(Q)}\right]^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{(Q\cdot n_z)}$ is the vector of decision variables containing all states and controls at each SC node (i.e., the deterministic trajectories). It should be noted that there are constraints in (4.1) that are merely depending on each SC node individually (e.g., state dynamics), while other constraints as well as the cost function can depend on all decision variables (e.g., probabilistic constraints; "robustification"). By the definition in (4.1), the expansion coefficients of the gPC expansion are available within the ROCP, because the deterministic OCPs are solved at each SC node, which gives the expansion coefficients using the SC formula in (2.68). Thus, (4.1) can directly optimize statistical moments and therefore produces a robust, optimal trajectory representation.

Generally, the ROCP given in (4.1) could be solved like a standard OCP as it is a deterministic representation of a large-scale OCP due to the SC. But, as already mentioned in Subsection 2.3.2, the required number of SC nodes and thus, required number of decision variables, grows rapidly with the expansion order and the number of uncertainties (Definition 2.3). Thus, the ROCP in (4.1) might become too large to be solvable by a nonlinear program (NLP) optimization scheme. To resolve this issue, and also utilize the structure of the ROCP in (4.1), in which most of the constraints are unconnected, i.e., independent of each other, the following paragraphs show the derivation of a DOCP structure that parallelizes most of the required calculations, while still solving the original ROCP in (4.1) without simplification (basic idea: Subsection 2.4.5).

Generally, the ROCP in e.g., (4.1) is distributed, i.e., divided into multiple smaller OCPs that can be solved independently. This independence must be secured by introducing connection variables $\boldsymbol{\nu}^{(i)} \in \mathbb{R}^{n_\nu}$ that enforce constraints or mitigate an interaction between the DOCPs to ensure that the original large-scale OCP in (4.1) is still satisfied. These connection variables are constants within the DOCPs (i.e., the "lower" levels) and the overall strategy is also referred to as primal decomposition. The connection problem, i.e., the calculation of the connection variables, is solved separately from the DOCPs in an upper level connection problem. Take into account that the introduction of connection variables and the separate solution of unconnected OCPs is a deliberate relaxation of the OC formulation. The applicability of this approach is proven in the application examples of Chapter 8 as well as in the KKT derivation of Section 4.4.

It should be noted that connection variables are used as the sensitive parameters in the DOCPs, i.e., the sensitivities (Subsection 2.4.2) can be calculated and used within the connection problem. Thus, it is $\mathbf{q} = \left[\left[\boldsymbol{\nu}^{(1)}\right]^\mathsf{T} \quad \dots \quad \left[\boldsymbol{\nu}^{(Q)}\right]^\mathsf{T}\right]^\mathsf{T}$ (take into account that this is already defined in the context of gPC-SC with Q SC nodes). Overall, the j-th DOCP, i.e., the OCP solved at the gPC-SC node $\boldsymbol{\theta}^{(j)}$, can then be stated according to (2.1) and (4.1) as follows:

$$\begin{aligned}
\min_{\mathbf{z}^{(j)}} \quad & J^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{J}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right) \\
\text{s.t.} \quad & \mathbf{z}_{\text{lb}}^{(j)} \leq \mathbf{z}^{(j)} \leq \mathbf{z}_{\text{ub}}^{(j)}, \\
& \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}\right) = \dot{\mathbf{x}}^{(j)}, \\
& \mathbf{c}^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{\mathbf{c}}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right) \leq \mathbf{0}, \\
& \boldsymbol{\psi}^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{\boldsymbol{\psi}}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right) = \mathbf{0}, \\
& \mathbb{P}\left[\mathbf{y}^{(j)} \in \mathcal{P}\right] \geq \boldsymbol{\xi}^{(j)}
\end{aligned} \tag{4.2}$$

Here, a $\widetilde{\cdot}$ over a function denotes dependencies on the connection variables, i.e., the sensitive parameters. Thus, (4.2) solves a deterministic OCP with the optimization variables $\mathbf{z}^{(j)}$ and external influences $\mathbf{q}$. Remember that the connection variable vector is not a part of the optimization parameters in the single DOCP, but is adapted according to a connection problem to ensure that the original solution of the large-scale, connected OCP (see (4.1)) is calculated. Overall, the connection variables merely provide an additional gradient shaping, i.e., descent direction, in the DOC context for the lower level OCPs. Take into account that the discretized form of (4.2) can be derived as for (2.13), which is omitted here because it is no new information.

As stated, the DOCPs in (4.2) must fulfill the original connected OCP formulation in (4.1). Thus, e.g., the cost function must e.g., fulfill the following condition:

$$\sum_{j=1}^{Q}\left[J^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{J}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right)\right] \equiv J \tag{4.3}$$

The same also applies for the constraints in (4.2). Take into account that (4.3) also directly implies a strategy to update the connection variables in the connection problem such that the original, connected OCP is solved.

Overall, a general connection problem, i.e., the update of the connection parameters to recover the original connected OCP, can be stated for gPC-SC as follows:

$$\boldsymbol{\upsilon}^{(j)} = \boldsymbol{\nu}^{(j)} - \underbrace{\mathbf{y}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{q}\right)}_{\mathbf{y}^{(j)}} \stackrel{!}{=} \mathbf{0} \in \mathbb{R}^{n_y}, \quad j = 1, \ldots, Q \tag{4.4}$$

Here, $\boldsymbol{\upsilon}^{(j)}$ is the error between the connection variables and the physical trajectories of the considered output quantity $\mathbf{y}^{(j)}$ that is used in the robustification. It should be noted that the connection variable errors for all connection problems are combined in the vector $\boldsymbol{\upsilon} = \left[\left[\boldsymbol{\upsilon}^{(1)}\right]^\mathsf{T} \quad \ldots \quad \left[\boldsymbol{\upsilon}^{(Q)}\right]^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{(n_y \times Q)}$. It is further important to note that (4.4) updates the j-th connection variable with the j-th OCP result. This means that the updated variables are given with respect to the j-th normalized time grid (see (2.5)). Thus, an interpolation might be required if different time grids are used. In addition, even if the same normalized time grid is used, the trajectories are given on a different physical time grid, as the initial and final time are generally optimization parameters and change the physical time grid (see (2.7)). Therefore, if e.g., a robust LAGRANGE cost is used, i.e., a function that requires an integration over the physical time, an appropriate interpolation (e.g., time-based, distance-to-end-point-based, covered-distance-based,...) must be conducted. This ensures that the time-based function is evaluated correctly using the connection variables. Take into account that this interpolation is automatically defined using a collocation method by using the applied collocation scheme.

Overall, the connection variables mitigate the solutions of other DOCPs, required for the robustification, to the currently considered DOCP. This makes it possible to calculate e.g., statistical moments by gPC-SC. As seen in the following, the choice of connection variables is directly depending on the statistical moment calculation of gPC (Subsection 2.3.4). Thus, the exact relations for the connection problem in (4.4) are based on the specifics of the implemented DOCP. Here, Section 4.2 derives the distributed formulation of common statistical moments, i.e., mean and variance in gPC, and gives an example on their distribution in the DOC context.

Further take into account that (4.4) can be used as long as the connection variables do not have to fulfill any constraints or bounds. If such should be considered or the efficiency of an NLP solver is desired in the connection problem, the following formulation can be used:

$$
\begin{aligned}
\min_{\boldsymbol{\nu}} \quad & J_{\mathrm{CP}} = \boldsymbol{\upsilon}^{\mathsf{T}}\boldsymbol{\upsilon} \\
\text{s.t.} \quad & \boldsymbol{\nu}_{\mathrm{lb}} \leq \boldsymbol{\nu} \leq \boldsymbol{\nu}_{\mathrm{ub}}, \\
& \mathbf{c}_{\mathrm{CP}}\left(\boldsymbol{\nu}\right) \leq \mathbf{0}, \\
& \boldsymbol{\psi}_{\mathrm{CP}}\left(\boldsymbol{\nu}\right) = \mathbf{0}, \\
& \mathbb{P}\left[\boldsymbol{\nu} \in \mathcal{P}_{\mathrm{CP}}\right] \geq \boldsymbol{\xi}_{\mathrm{CP}}, \\
& \boldsymbol{\upsilon}^{\mathsf{T}}\boldsymbol{\upsilon} \leq \delta
\end{aligned}
\tag{4.5}
$$

Here, the subscript CP denotes the connection problem. It should be noted that in (4.5), the NLP is adapting the optimization parameters $\boldsymbol{\nu} = \left[\left[\boldsymbol{\nu}^{(1)}\right]^{\mathsf{T}} \quad \dots \quad \left[\boldsymbol{\nu}^{(Q)}\right]^{\mathsf{T}}\right]^{\mathsf{T}}$ until the connection error $\boldsymbol{\upsilon}$, as defined in (4.4), becomes minimal. Take into account that the multiplication of the connection error $\boldsymbol{\upsilon}$ with itself assures that the minimum is indeed at $\boldsymbol{\upsilon} = \mathbf{0}$ because the cost function is quadratic. Additionally, the connection variables fulfill the desired box as well as nonlinear equality and inequality and probabilistic constraints.

It should be further noted that (4.5) is formulated as a min-max problem, because the connection error should be minimized and thus, reach a value close to zero. As some constraint combinations might lead to optimal solutions with large connection errors, there is a maximal bound δ enforced on the connection error, which ensures that the optimal solution is only feasible close to a vanishing connection error.

Thus, although (4.4) provides the basic connection problem, which also gives a good idea of the characteristics of the connection problem, (4.5) provides a more general definition of the connection problem. Therefore, (4.5) is normally applied, especially as it allows the use of NLP solvers, which are specifically tailored to solve the connection problem formulation efficiently by e.g., sophisticated line-search procedures.

Overall, the DOC formulation used for ROC can be written as follows:

$$
\begin{aligned}
\min_{\boldsymbol{\nu}} \quad & J_{\mathrm{CP}} = \boldsymbol{\upsilon}^{\mathsf{T}}\boldsymbol{\upsilon} \\
\text{s.t.} \quad & \boldsymbol{\nu}_{\mathrm{lb}} \leq \boldsymbol{\nu} \leq \boldsymbol{\nu}_{\mathrm{ub}}, \\
& \mathbf{c}_{\mathrm{CP}}(\boldsymbol{\nu}; \mathbf{z}^{(j)}) \leq \mathbf{0}, \\
& \boldsymbol{\psi}_{\mathrm{CP}}(\boldsymbol{\nu}; \mathbf{z}^{(j)}) = \mathbf{0}, \\
& \mathbb{P}\left[\boldsymbol{\nu} \in \mathcal{P}_{\mathrm{CP}}\right] \geq \boldsymbol{\xi}_{\mathrm{CP}} \\
& \boldsymbol{\upsilon}^{\mathsf{T}}\boldsymbol{\upsilon} \leq \delta \\
& \text{with } \mathbf{z}^{(j)} = \arg\min \quad J^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{J}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right), \qquad j = 1, \dots, Q, \\
& \qquad\qquad \text{s.t.} \quad \mathbf{z}_{\mathrm{lb}}^{(j)} \leq \mathbf{z}^{(j)} \leq \mathbf{z}_{\mathrm{ub}}^{(j)}, \\
& \qquad\qquad \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}\right) = \dot{\mathbf{x}}^{(j)}, \\
& \qquad\qquad \mathbf{c}^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{\mathbf{c}}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right) \leq \mathbf{0}, \\
& \qquad\qquad \boldsymbol{\psi}^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{\boldsymbol{\psi}}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right) = \mathbf{0}, \\
& \qquad\qquad \mathbb{P}\left[\mathbf{y}^{(j)} \in \mathcal{P}\right] \geq \boldsymbol{\xi}^{(j)}
\end{aligned}
\tag{4.6}
$$

Here, the bi-level structure of the DOC is clear (see Figure 2.22). It is reminded that the upper level connection problem gives the connection variables to the lower level. The lower level then provides the updated trajectories that the upper level uses to improve on the connection variables. The procedure stops when the error in the connection variables converged to zero. Thus, the remaining issue in (4.6) is the choice of the connection variables $\boldsymbol{\nu}$, which is introduced in Section 4.2.

Before that, the algorithm that solves the DOC framework, and is used within this thesis, can also be schematically written as given in Algorithm 4.1. The algorithm is written for the general version of the connection problem using the optimization formulation in (4.5). Nonetheless, it can also be directly applied to the case where the simple constraint in (4.4) is used in the connection problem. Once more, it is reminded that, as it is generally difficult to calculate a second order sensitivity (i.e., the Hessian), the connection problem normally only runs the NLP solution with the Jacobian and gets the Hessian by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update.

Algorithm 4.1 Generic distributed optimal control framework algorithm with generalized polynomial chaos.

Require:

Initialize iteration counter $k = 1$.

Set the optimality $\epsilon_{\text{opt,CP},*}$ and feasibility tolerance $\epsilon_{\text{feas,CP},*}$ of the connection problem.

Define maximum number of iterations $k_{\max}$.

Initialize connection variables $\boldsymbol{\nu}_0$ (e.g., physical initial guess).

Calculate optimality $\epsilon_{\text{opt,CP},0}$ and feasibility $\epsilon_{\text{feas,CP},0}$ of initial connection problem.

1: **while** $\epsilon_{\text{opt,CP},k-1} > \epsilon_{\text{opt,CP},*}$ & $\epsilon_{\text{feas,CP},k-1} > \epsilon_{\text{feas,CP},*}$ & $k \leq k_{\max}$ **do**
2: Assign the connection variables $\boldsymbol{\nu}_{k-1}$ to the lower level OCPs.
3: **for all** $\boldsymbol{\theta}^{(j)}$ **do**
4: Solve the DOCPs in (4.2) by e.g., an NLP solver (Algorithm 2.1) at each $\boldsymbol{\theta}^{(j)}$ independent of other DOCPs, with current connection variables $\boldsymbol{\nu}_{k-1}^{(i)}$.
5: Extract the optimal trajectories $\mathbf{y}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{q}\right)$ required for the connection problem.
6: **end for**
7: Get connection cost and constraints as well as their Jacobians.
8: Update the connection variable $\boldsymbol{\nu}_k^{(j)}$ using $\mathbf{y}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{q}\right)$ and (4.5).
9: Check the current optimality tolerance $\epsilon_{\text{opt,CP},k}$ and feasibility tolerance $\epsilon_{\text{feas,CP},k}$.
10: Increase counter: $k = k + 1$
11: **end while**

12: **return** Optimal connection variables $\boldsymbol{\nu}_{\text{opt}}$ and optimal robust trajectory $\mathbf{z}_{\text{opt}}$.

4.2 Definition of Statistical Moments and Connection Variables

Generally, the DOC robustification in e.g., (4.1) is based on the statistical moments and, depending on the formulation, the gPC response surface. Thus, especially the statistical moments must be distributed and connection variables must be defined for these, which is shown in the following.

Evaluating the statistical moments for mean and variance from gPC in the DOC framework requires rewriting (2.69) and (2.70). This is due to the fact that the DOC framework relies on a distributed evaluation of the trajectories at the SC nodes, $\boldsymbol{\theta}^{(j)}$, while the statistical moments are evaluated using the gPC expansion coefficients. To achieve this projection, the SC expansion formula in (2.68) must be inserted in the statistical moment calculation of (2.69) and (2.70). Overall, this then allows the distribution of the statistical moments and the optimization of robust trajectories in DOC. As noted, a similar DOC formulation was already introduced by the author in [102].

At first, the mean value in (2.69) can be rewritten as follows [102]:

$$\mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right)\right] \approx \hat{\mathbf{y}}^{(0)}\left(\mathbf{z}^{(j)},\mathbf{q}\right) \approx \sum_{j=1}^{Q} \mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right) \underbrace{\Phi^{(0)}\left(\boldsymbol{\theta}^{(j)}\right)}_{=1} \alpha^{(j)} \tag{4.7}$$

It should be noted that (4.7) is already the desired distributed representation of the mean value even without introducing connection variables. This is due to the fact that the addends in (4.7) can be solved for each SC node individually. Thus, a DOC setup that only tries to optimize/constrain a mean value is indeed perfectly decoupled and can be solved independently in one step, i.e., without a connection problem. Therefore, only an OCP at each SC node must be solved.

Then, the variance in (2.70) can be written in a distributed form by [102]:

$$\sigma^2\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right)\right] \approx \sum_{m=1}^{M-1}\left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z},\mathbf{q}\right)\right]^2 \approx \sum_{m=1}^{M-1}\left[\sum_{j=1}^{Q} \mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\alpha^{(j)}\right]^2 \tag{4.8}$$

Here, the following binomial formula can be applied [5, p. 69f.]

$$\left[\sum_{j=1}^{Q} x^{(j)}\right]^2 = \sum_{j=1}^{Q}\left[x^{(j)}\right]^2 + \sum_{j\neq i} x^{(i)}x^{(j)} \tag{4.9}$$

Then, the squared sum in (4.8) can be rewritten, based on (4.9), as follows [102]:

$$\begin{aligned}\sigma^2\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right)\right] \approx & \sum_{m=1}^{M-1}\sum_{j=1}^{Q}\left[\mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\right]^2\left[\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\right]^2\left[\alpha^{(j)}\right]^2 \\ & + \sum_{m=1}^{M-1}\left[\sum_{j\neq i}\mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\alpha^{(j)}\mathbf{y}\left(\mathbf{z}^{(i)};\boldsymbol{\theta}^{(i)},\mathbf{q}\right)\Phi^{(m)}\left(\boldsymbol{\theta}^{(i)}\right)\alpha^{(i)}\right]\end{aligned} \tag{4.10}$$

Then, (4.10) can be split in parts that are only dependent on the expansion coefficient summation m and the SC summation j respectively as follows [102]:

$$\begin{aligned}\sigma^2\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right)\right] \approx & \sum_{j=1}^{Q}\left[\mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\right]^2\left[\alpha^{(j)}\right]^2\sum_{m=1}^{M-1}\left[\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\right]^2 \\ & +\sum_{j\neq i}\left[\mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\underbrace{\mathbf{y}\left(\mathbf{z}^{(i)};\boldsymbol{\theta}^{(i)},\mathbf{q}\right)}_{\boldsymbol{\nu}^{(i)}}\alpha^{(j)}\alpha^{(i)}\sum_{m=1}^{M-1}\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\Phi^{(m)}\left(\boldsymbol{\theta}^{(i)}\right)\right]\end{aligned} \tag{4.11}$$

Take into account that the first addend in (4.11) is again decoupled from other SC nodes and can be handled distributed (as the orthogonal polynomials and integration weights are known because they are defined by the uncertainty). On the other hand, the second addend in (4.11) does not exhibit this preferable behavior: Here, a connection between the current DOCP j and all other DOCPs i is clear, because the outputs are exchanged. Thus, when optimizing/constraining the variance the introduction of connection variables in the DOC framework is required. These are defined for the trajectories $\mathbf{y}\left(\mathbf{z}^{(i)};\boldsymbol{\theta}^{(i)},\mathbf{q}\right)$ that are from different DOCPs.

It should be noted in the context of (4.11) that the connection problem still remains easy to solve. To show this, the second line of (4.11) is looked at and the connection variables $\boldsymbol{\nu}^{(i)}$ are used as follows [102]:

$$\widetilde{\sigma}^2\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right)\right] = \sum_{j\neq i}\left[\mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\boldsymbol{\nu}^{(i)}\alpha^{(j)}\alpha^{(i)}\sum_{m=1}^{M-1}\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\Phi^{(m)}\left(\boldsymbol{\theta}^{(i)}\right)\right] \tag{4.12}$$

As already mentioned, this is an exchange of trajectories, and thus, information, between the different DOCPs.

Now, the connection variables for each DOCP j must fulfill the following condition enforced within the connection problem, as already introduced in (4.4) (take into account that the formulation using the OCP in (4.5) is not used due to simplicity of description but is analogous) [102]:

$$\boldsymbol{\upsilon}^{(j)} = \boldsymbol{\nu}^{(j)} - \mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right) \stackrel{!}{=} \mathbf{0} \tag{4.13}$$

In (4.13), $\mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)$ are known values in the connection problem after having solved all DOCPs unconnected. Thus, (4.4) can be solved by e.g., the Newton method. Here, the directional derivative of (4.4) with respect to the connection variables, required for the Newton method, is given as follows [102]:

$$\left(\boldsymbol{\upsilon}^{(j)}\right)' = \frac{\mathrm{d}\boldsymbol{\upsilon}^{(j)}}{\mathrm{d}\boldsymbol{\nu}^{(j)}} = \mathbf{I} - \underbrace{\frac{\mathrm{d}\mathbf{y}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)}{\mathrm{d}\boldsymbol{\nu}^{(j)}}}_{=\mathbf{0}} \tag{4.14}$$

It should be noted that the second addend equates to zero as a direct consequence of (4.11) and (4.12): Here, it is clear that the connection variables only introduce a cross-coupling of the DOCPs, while no direct coupling is introduced. This means that the connection variables are here only evaluated for elements that are not equal to the current DOCP j. Thus, the connection variables $\boldsymbol{\nu}^{(j)}$ are not a part of the j-th DOCP and consequently have no influence on its optimal trajectory.

The structure in (4.14) is desirable, as no sensitivities must be calculated and provided to the upper level connection problem and also justifies the seemingly complex rewriting of (4.8) to (4.11). Then, the update of (4.4), using a NEWTON step [5, p. 294], with the derivative in (4.14) is given as follows [102]:

$$\boldsymbol{\nu}_{new}^{(j)} = \boldsymbol{\nu}^{(j)} - \beta \left[\frac{\boldsymbol{\upsilon}^{(j)}}{\left(\boldsymbol{\upsilon}^{(j)}\right)'} \right] = \boldsymbol{\nu}^{(j)} - \beta \left[\boldsymbol{\nu}^{(j)} - \mathbf{y}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{q}\right) \right] \tag{4.15}$$

Here, $\beta \in]0;1]$ is a step size (line search) parameter to find the best possible update to get as close to the exact problem solution in (4.13) as possible. It should be noted that this line search is crucial for the convergence of the DOC framework. This is a further reason to generally choose the formulation of the connection problem as presented in (4.5), because NLP solvers have sophisticated line search methods internally available.

Overall, also further statistics, e.g., covariance, skewness, and kurtosis, can be derived for the DOC framework and thus, multiple kinds of robustifying cost and constraint functions can be implemented. As the examples in this thesis concentrate on mainly mean and variance, the derived formulas in (4.7) and (4.11) are suitable and show the general procedure.

As an example for the definition of the DOC with statistical moments, the following cost function is looked at:

$$J = \mathbb{E}\left[y\left(\mathbf{z}; \boldsymbol{\theta}, \mathbf{q}\right)\right] + w \cdot \sigma^2\left[y\left(\mathbf{z}; \boldsymbol{\theta}, \mathbf{q}\right)\right] \tag{4.16}$$

Here, y the variable whose mean and variance should be optimized (in the scalar context for simplicity) and w is a scaling factor that weighs the influence of mean and standard deviation (similar to the Pareto optimization in Subsection 2.4.1). Take into account that the mean and standard deviation are not required to be taken for the same value, although this is done here for the simplicity of writing.

Now, (4.7) and (4.11) can be used to rewrite (4.16) as follows:

$$\begin{aligned} J = \sum_{j=1}^{Q} y\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{q}\right) \alpha^{(j)} + w \Bigg\{ & \sum_{j=1}^{Q} \left[y\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{q}\right) \right]^2 \left[\alpha^{(j)}\right]^2 \sum_{m=1}^{M-1} \left[\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \right]^2 \\ & + \sum_{j \neq i} \Big[y\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathbf{q}\right) \underbrace{y\left(\mathbf{z}^{(i)}; \boldsymbol{\theta}^{(i)}, \mathbf{q}\right)}_{\boldsymbol{\nu}^{(i)}} \alpha^{(j)} \alpha^{(i)} \sum_{m=1}^{M-1} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(m)}\left(\boldsymbol{\theta}^{(i)}\right) \Big] \Bigg\} \end{aligned} \tag{4.17}$$

Here, (4.17) directly provides the rule to split up the cost function for the DOCPs (see (4.2)) as the first line is only dependent on the j-th DOCP, while the second line has the additional influence of the connection variables:

$$\begin{aligned} J^{(j)}\left(\mathbf{z}^{(j)}\right) &= y\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\alpha^{(j)} + w\cdot\left[y\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\right]^2\left[\alpha^{(j)}\right]^2\sum_{m=1}^{M-1}\left[\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\right]^2 \\ \widetilde{J}^{(j)}\left(\mathbf{z}^{(j)};\mathbf{q}\right) &= w\cdot\sum_{j\neq i} y\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right)\boldsymbol{\nu}^{(i)}\alpha^{(j)}\alpha^{(i)}\sum_{m=1}^{M-1}\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\Phi^{(m)}\left(\boldsymbol{\theta}^{(i)}\right) \end{aligned} \tag{4.18}$$

Thus, (4.18) is a standardized representation for a mean and variance cost function. It should be noted that (4.18) can directly be extended to further statistical moments in the cost function as well as different or multiple outputs. Therefore, robustness modifications can be introduced as desired by the user.

4.3 Constraints in Distributed Optimal Control

Building upon the derivations in Section 4.2, it is also possible to distribute constraints in the context of a DOCP. Here, a major part is splitting up the respective boundary values of e.g., the box constraint for each DOCP as each sub-problem only fulfills a part of the constraint. To illustrate the procedure, a generic box constraint for the i-th system output is defined as follows:

$$y_{\mathrm{lb},i} \leq y_i\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right) \leq y_{\mathrm{ub},i} \tag{4.19}$$

This constraint can be generally solved in the NLP context using two inequality constraints as introduced in (2.13):

$$\begin{aligned} y_{\mathrm{lb},i} - y_i\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right) &\leq 0 \\ y_i\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right) - y_{\mathrm{ub},i} &\leq 0 \end{aligned} \tag{4.20}$$

Here, $y_i\left(\mathbf{z};\boldsymbol{\theta},\mathbf{q}\right)$ is the random response of the system, which should fulfill the robust constraint, and is e.g., distributed according to (4.7) for the mean value or (4.11) for the variance (or any other combination of statistical moments). Thus, only the j-th part of this variable is available in the j-th DOCP. Consequently, the j-th DOCP also only fulfills a portion of the constraint such that the original constraint is fulfilled in the connected problem, yielding:

$$\begin{aligned} \omega_{\mathrm{lb},i}^{(j)}\cdot y_{\mathrm{lb},i} - y_i\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right) &\leq 0 \\ y_i\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)},\mathbf{q}\right) - \omega_{\mathrm{ub},i}^{(j)}\cdot y_{\mathrm{ub},i} &\leq 0 \end{aligned} \tag{4.21}$$

Here, $\omega_{\mathrm{lb},i}^{(j)}$ and $\omega_{\mathrm{ub},i}^{(j)}$ are the weights for the lower and upper bound of the constraint respectively, which may not be equal (but generally are). These weights are combined in the weight vector defined by $\boldsymbol{\omega} = \begin{bmatrix}\omega_{\mathrm{lb},1,\dots,n_y}^{(1)} & \omega_{\mathrm{ub},1,\dots,n_y}^{(1)} & \dots & \omega_{\mathrm{lb},1,\dots,n_y}^{(Q)} & \omega_{\mathrm{ub},1,\dots,n_y}^{(Q)}\end{bmatrix}^\mathsf{T}$ and are the portion to which the j-th DOCP is fulfilling the constraint. It should be noted that this is very similar to the cost function distribution in (4.18): Here, the addend of the cost function that is related to the j-th DOCP, i.e., does not contain any connection variables, also only minimizes a portion of the cost function. This portion is given by the SC weight $\alpha^{(j)}$. Thus, the SC weight can be viewed as the weight multiplier for the cost, i.e., $\omega_{\mathrm{lb}} = \alpha^{(j)} = \omega_{\mathrm{ub}}$. For the constraint, this procedure is just generalized because the bound value must be distributed such that the nonlinear constraint is fulfilled in the connected problem. As this bound is not given by the gPC expansion, the chosen weight is also not directly obvious and must be optimized accordingly.

Then, the weight vector is added as an additional optimization variable in the connection problem of (4.5), thus, defining an extended version as follows:

$$
\begin{aligned}
\min_{\boldsymbol{\nu},\boldsymbol{\omega}} \quad & J_{\mathrm{CP}} = \boldsymbol{v}^\mathsf{T}\boldsymbol{v} \\
\text{s.t.} \quad & \boldsymbol{\nu}_{\mathrm{lb}} \leq \boldsymbol{\nu} \leq \boldsymbol{\nu}_{\mathrm{ub}}, \\
& \mathbf{c}_{\mathrm{CP}}\left(\boldsymbol{\nu}; \mathbf{z}^{(j)}\right) \leq \mathbf{0}, \\
& \boldsymbol{\psi}_{\mathrm{CP}}\left(\boldsymbol{\nu}; \mathbf{z}^{(j)}\right) = \mathbf{0}, \\
& \mathbb{P}\left[\boldsymbol{\nu} \in \mathcal{P}\right] \geq \xi_{\mathrm{CP}}, \\
& \boldsymbol{v}^\mathsf{T}\boldsymbol{v} \leq \delta, \\
& \sum_{j=1}^{Q} \omega_{\mathrm{lb},i}^{(j)} = 1 \qquad \forall i, \\
& \sum_{j=1}^{Q} \omega_{\mathrm{ub},i}^{(j)} = 1 \qquad \forall i, \\
& \mathbf{0} < \boldsymbol{\omega}_{\mathrm{lb}} \leq \boldsymbol{\omega} \leq \boldsymbol{\omega}_{\mathrm{ub}} < \mathbf{1}
\end{aligned}
\tag{4.22}
$$

Here, the optimization goal still is to minimize the error between the connection variables and the physical trajectories as in (4.5). In addition, it must be considered that the original constraint should be satisfied in the connected problem, which yields the fact that the weightings of the distributed constraint bounds must be equal to one in sum. Furthermore, these constraint weights must be greater than zero and smaller than one, which consequently states that each distributed trajectory must have an influence and the optimizer is not allowed to "remove" particular extreme trajectories in order to fulfill the constraint easier.

To get an understanding of the weights introduced for the distributed constraint in (4.21), the distribution of the mean value in (4.7) is looked at: From Theorem C.1, it is known that the zeroth orthogonal polynomial is always equal to one and thus, the distribution weight in (4.7) can be considered similar to the SC weight $\alpha^{(j)}$. Thus, in a simple

setup $\omega^{(j)} = \alpha^{(j)}$ can be assumed within the connection problem, which would here exactly recover the mean value. Therefore, the weights introduced when distributing a constraint can be viewed as the weights required for the SC to appropriately weigh the trajectories to get the correct statistical moments. In this context, it is also straightforward to define bounds and initial values for the weights $\omega^{(j)}$ as they will be similar to the SC weights $\alpha^{(j)}$. Still, due to the nonlinearity of e.g., the standard deviation, the SC weight $\alpha^{(j)}$ is not always the exact solution obtained from the connection problem. Thus, it is of special importance to choose appropriate weight bounds such that the DOCPs can be solved but the solution quality is still sufficient. This means that all SC trajectories should significantly contribute to the constraint.

Take into account that the connection problem formulation in (4.22) always requires sensitivities from the DOCPs due to the introduced constraint weights. As shown, this is not necessary in some cases with only distributed statistical moments in e.g., the cost function (see (4.14)). Thus, a distribution of constraints comes at an additional cost. It should be further noted that this issue can be mitigated by approximating the full, exact constraint, i.e., non-distributed, in each DOCP. This can be done by using the available connection variables and also removes the necessity to provide suitable bounds for the constraint weights. On the other hand, this might yield to fairly poor convergence due to the "wrong" approximation of the constraint in each DOCP. Generally, bad initial values for the connection variables might not even permit convergence at all. Therefore, the best-suited solution strategy for the DOC must be found individually.

4.4 Derivation of Karush-Kuhn-Tucker Conditions for Distributed Optimal Control

This section covers the derivation of the KKT conditions for the DOC with gPC-SC as developed in this thesis (see (4.1), (4.2), and (4.5)). The goal is to show that DOC still solve the original connected problem in the ROC gPC-SC framework (see (4.1)). It is important to note that the DOCPs are solved individually using an NLP solver (e.g., Interior Point Optimizer (IPOPT) or Sparse Nonlinear Optimizer (SNOPT)). Thus, the DOCPs on their own fulfill the KKT optimality conditions (Definition 2.1), which is a necessary condition. This optimality of the single DOCP is important for the derivation of the optimality of the connected, large-scale OCP. Additionally, the connection problem is considered to be solved and thus, fulfills the KKT conditions. As an example for the derivation of the KKT conditions, the distributed cost function in (4.1) is used. This results in a DOCP as follows:

$$\begin{aligned} \min_{\mathbf{z}^{(j)}} \quad & J^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{J}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right) \\ \text{s.t.} \quad & \mathbf{z}_{\text{lb}}^{(j)} \leq \mathbf{z}^{(j)} \leq \mathbf{z}_{\text{ub}}^{(j)}, \\ & \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}\right) = \dot{\mathbf{x}}^{(j)}, \\ & \mathbf{c}^{(j)}\left(\mathbf{z}^{(j)}\right) \leq \mathbf{0}, \\ & \boldsymbol{\psi}^{(j)}\left(\mathbf{z}^{(j)}\right) = \mathbf{0} \end{aligned} \tag{4.23}$$

It should be noted here that the constraints are not distributed, i.e., they are not coupled by statistical moments for the sake of simplicity.

For the DOCP defined in (4.23), the overall optimization parameter vector is defined as already introduced:

$$\bar{\mathbf{z}} = \left[\left[\mathbf{z}^{(1)}\right]^{\mathsf{T}} \quad \dots \quad \left[\mathbf{z}^{(Q)}\right]^{\mathsf{T}}\right]^{\mathsf{T}} \tag{4.24}$$

Due to the fact that the constraints do not have to be distributed, the LAGRANGE multipliers can be split analogously:

$$\bar{\boldsymbol{\lambda}} = \left[\left[\boldsymbol{\lambda}^{(1)}\right]^{\mathsf{T}} \quad \dots \quad \left[\boldsymbol{\lambda}^{(Q)}\right]^{\mathsf{T}}\right]^{\mathsf{T}}, \quad \bar{\boldsymbol{\mu}} = \left[\left[\boldsymbol{\mu}^{(1)}\right]^{\mathsf{T}} \quad \dots \quad \left[\boldsymbol{\mu}^{(Q)}\right]^{\mathsf{T}}\right]^{\mathsf{T}} \tag{4.25}$$

Generally, the feasibility condition (KKT Condition 3 in Definition 2.1) is directly extended to the concept of DOC. In this context, the extension is according to (4.24) as well as (4.25) and can be stated as follows:

$$\begin{aligned} \mathbf{c}\left(\bar{\mathbf{z}}_{\text{opt}}; \mathbf{q}_{\text{opt}}\right) &= \begin{bmatrix} \underbrace{\mathbf{c}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}\right)}_{\leq \mathbf{0}} + \underbrace{\widetilde{\mathbf{c}}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}; \mathbf{q}_{\text{opt}}\right)}_{\equiv \mathbf{0}} \\ \vdots \\ \underbrace{\mathbf{c}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}\right)}_{\leq \mathbf{0}} + \underbrace{\widetilde{\mathbf{c}}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}; \mathbf{q}_{\text{opt}}\right)}_{\equiv \mathbf{0}} \end{bmatrix} \leq \mathbf{0} \\ \boldsymbol{\psi}\left(\bar{\mathbf{z}}_{\text{opt}}; \mathbf{q}_{\text{opt}}\right) &= \begin{bmatrix} \underbrace{\boldsymbol{\psi}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}\right)}_{= \mathbf{0}} + \underbrace{\widetilde{\boldsymbol{\psi}}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}; \mathbf{q}_{\text{opt}}\right)}_{\equiv \mathbf{0}} \\ \vdots \\ \underbrace{\boldsymbol{\psi}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}\right)}_{= \mathbf{0}} + \underbrace{\widetilde{\boldsymbol{\psi}}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}; \mathbf{q}_{\text{opt}}\right)}_{\equiv \mathbf{0}} \end{bmatrix} = \mathbf{0} \end{aligned} \tag{4.26}$$

It is clear that (4.26) is fulfilled within the connected OCP as long as it is fulfilled within each DOCP (see (4.23)). Due to the fact that the DOCPs on their own fulfill the KKT conditions (as they are optimal), it is clear that they automatically fulfill the connected OCP in (4.26) as well.

As a next step, the derivative of the LAGRANGE function is looked at (KKT Condition 2 in Definition 2.1) that yields the dual feasibility as follows:

$$\begin{aligned}\nabla_{\bar{\mathbf{z}}}\mathcal{L}\left(\bar{\mathbf{z}}_{\text{opt}},\bar{\boldsymbol{\lambda}}_{\text{opt}},\bar{\boldsymbol{\mu}}_{\text{opt}};\mathbf{q}_{\text{opt}}\right) &= \nabla_{\bar{\mathbf{z}}}J\left(\bar{\mathbf{z}}_{\text{opt}};\mathbf{q}_{\text{opt}}\right)\\ &\quad+\bar{\boldsymbol{\lambda}}_{\text{opt}}^{\mathsf{T}}\nabla_{\bar{\mathbf{z}}}\mathbf{c}\left(\bar{\mathbf{z}}_{\text{opt}};\mathbf{q}_{\text{opt}}\right)+\bar{\boldsymbol{\mu}}_{\text{opt}}^{\mathsf{T}}\nabla_{\bar{\mathbf{z}}}\boldsymbol{\psi}\left(\bar{\mathbf{z}}_{\text{opt}};\mathbf{q}_{\text{opt}}\right)\overset{!}{=}\mathbf{0}\end{aligned} \tag{4.27}$$

Now, the DOCPs for cost and constraints in (4.27) can be inserted yielding:

$$\begin{aligned}\nabla_{\bar{\mathbf{z}}}\sum_{j=1}^{Q}\left[J^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)}\right)+\widetilde{J}^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)};\mathbf{q}_{\text{opt}}\right)\right]&+\left[\left[\boldsymbol{\lambda}_{\text{opt}}^{(1)}\right]^{\mathsf{T}}\quad\dots\quad\left[\boldsymbol{\lambda}_{\text{opt}}^{(Q)}\right]^{\mathsf{T}}\right]\nabla_{\bar{\mathbf{z}}}\begin{bmatrix}\mathbf{c}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}\right)\\ \vdots\\ \mathbf{c}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}\right)\end{bmatrix}\\ &+\left[\left[\boldsymbol{\mu}_{\text{opt}}^{(1)}\right]^{\mathsf{T}}\quad\dots\quad\left[\boldsymbol{\mu}_{\text{opt}}^{(Q)}\right]^{\mathsf{T}}\right]\nabla_{\bar{\mathbf{z}}}\begin{bmatrix}\boldsymbol{\psi}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}\right)\\ \vdots\\ \boldsymbol{\psi}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}\right)\end{bmatrix}\end{aligned} \tag{4.28}$$

As differentiation is a linear operator, (4.28) can be rewritten as follows (this is possible in this case as the constraints are unconnected):

$$\begin{aligned}\sum_{j=1}^{Q}\nabla_{\bar{\mathbf{z}}}\left[J^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)}\right)+\widetilde{J}^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)};\mathbf{q}_{\text{opt}}\right)\right]&+\left[\left[\boldsymbol{\lambda}_{\text{opt}}^{(1)}\right]^{\mathsf{T}}\quad\dots\quad\left[\boldsymbol{\lambda}_{\text{opt}}^{(Q)}\right]^{\mathsf{T}}\right]\nabla_{\bar{\mathbf{z}}}\begin{bmatrix}\mathbf{c}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}\right)\\ \vdots\\ \mathbf{c}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}\right)\end{bmatrix}\\ &+\left[\left[\boldsymbol{\mu}_{\text{opt}}^{(1)}\right]^{\mathsf{T}}\quad\dots\quad\left[\boldsymbol{\mu}_{\text{opt}}^{(Q)}\right]^{\mathsf{T}}\right]\nabla_{\bar{\mathbf{z}}}\begin{bmatrix}\boldsymbol{\psi}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}\right)\\ \vdots\\ \boldsymbol{\psi}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}\right)\end{bmatrix}\end{aligned} \tag{4.29}$$

Finally, differentiation and summation are commutative for smooth functions [5, p. 1037f. & p. 1129] and the order can be rearranged once more. Equation (4.29) simplifies as follows:

$$\sum_{j=1}^{Q}\Bigg[\underbrace{\nabla_{\bar{\mathbf{z}}}J^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)}\right)+\nabla_{\bar{\mathbf{z}}}\widetilde{J}^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)};\mathbf{q}_{\text{opt}}\right)+\left(\boldsymbol{\lambda}_{\text{opt}}^{(j)}\right)^{\mathsf{T}}\nabla_{\bar{\mathbf{z}}}\mathbf{c}^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)}\right)+\left(\boldsymbol{\mu}_{\text{opt}}^{(j)}\right)^{\mathsf{T}}\nabla_{\bar{\mathbf{z}}}\boldsymbol{\psi}^{(j)}\left(\mathbf{z}_{\text{opt}}^{(j)}\right)}_{\equiv\mathbf{0},\quad\text{KKT Condition 2}}\Bigg]\overset{!}{=}\mathbf{0} \tag{4.30}$$

It is clear that (4.30) is fulfilled as each of the DOCPs in (4.23) fulfills the KKT conditions and the connection problem is solved, i.e., the optimal connection variables and therefore the exact trajectories are assigned.

Using the knowledge gained from (4.26) and (4.30), the Lagrange multiplier condition (KKT Condition 1 in Definition 2.1) as well as the complementary slackness condition (KKT Condition 4 in Definition 2.1) are easily shown to be fulfilled in the distributed setup. The Lagrange multiplier condition can be stated as follows:

$$\boldsymbol{\lambda}_{\text{opt}}=\begin{bmatrix}\boldsymbol{\lambda}_{\text{opt}}^{(1)}\\ \vdots\\ \boldsymbol{\lambda}_{\text{opt}}^{(Q)}\end{bmatrix}\geq\mathbf{0} \tag{4.31}$$

Similarly, the slackness condition is as follows:

$$\boldsymbol{\lambda}_{\text{opt}}^{\mathsf{T}}\mathbf{c}\left(\mathbf{z}_{\text{opt}}\right) = \begin{bmatrix} \left(\boldsymbol{\lambda}_{\text{opt}}^{(1)}\right)^{\mathsf{T}} \mathbf{c}^{(1)}\left(\mathbf{z}_{\text{opt}}^{(1)}\right) \\ \vdots \\ \left(\boldsymbol{\lambda}_{\text{opt}}^{(Q)}\right)^{\mathsf{T}} \mathbf{c}^{(Q)}\left(\mathbf{z}_{\text{opt}}^{(Q)}\right) \end{bmatrix} = \mathbf{0} \tag{4.32}$$

Once more, (4.31) and (4.32) are fulfilled because the DOCP (see (4.23)) does fulfill the KKT optimality conditions.

Thus, (4.26), (4.30), (4.31), and (4.32) show that the DOC framework fulfills the optimality conditions of the original connected OCP based on the optimality of lower level DOCPs (see (4.1)). It is therefore a solution of this original OCP and the proposed DOC algorithm can be used to calculate robust trajectories of a complex ROCP without simplifications.

It should be noted that there are also other (more general) ways to prove the convergence of distributed (bi-level) optimization frameworks [49, 55, 78]. These are not covered in this thesis, as merely the developed DOC algorithm based on gPC-SC should be proven to be optimal. Additionally, the functionality of the algorithm is further examined in application cases. Still, it is important to note that the main prerequisite for the convergence of the DOC algorithm is that the lower level DOCPs remain feasible and result in optimal trajectories as well as that the connection problem is solved. This must be assured by e.g., choosing suitable initial values as well as optimality and feasibility conditions for the lower levels and the connection problem. Additionally, the convergence behavior of the NLPs can be controlled to assure better convergence, e.g., by an appropriate adaptation of the barrier parameter in IPOPT. This also improves the convergence speed.

Chapter 5

Optimal Control Transcription Using Generalized Polynomial Chaos

This chapter introduces the developed generalized polynomial chaos (gPC) transcription for the robust open-loop direct optimal control problem (ROCP) of this thesis. Thus, Contribution 1 of the thesis is covered in this chapter. Within this framework, the standard open-loop direct optimal control (OC) nonlinear program (NLP) transcription, introduced in Subsection 2.1.2.1, is changed to a representation using the gPC expansion coefficients. This transcription procedure is also illustrated in Figure 5.1. A benefit of the method is the fact that it provides the opportunity to directly optimize statistical moments within e.g., the cost function, as the expansion coefficients are part of the optimization parameters. Additionally, the method gives the capability to evaluate chance constraints (CCs) as the gPC expansion is available for sampling within the NLP as well.

In contrast to the distributed open-loop direct optimal control (DOC) framework, which was introduced in Chapter 4 and can also be applied to solve ROCPs, the proposed gPC collocation method does not rely on a distribution and thus, multiple solutions of the open-loop direct optimal control problem (OCP). Instead it creates a large-scale OCP that can be solved in a single optimization run. To reduce the already mentioned dimensionality problem, it does directly apply the expansion coefficients rather than the physical stochastic collocation (SC) trajectories, as the expansion coefficients are evaluated on a reduced tensor grid (Definition 2.3). Therefore, fewer optimization parameters are required. Overall, the DOC in Chapter 4 and the gPC collocation, introduced in this chapter can be considered as complementary frameworks, where the DOC can e.g., be applied to large-scale OCPs that cannot be solved by the introduced gPC collocation method, while the gPC collocation method provides an embedded method without computational overhead.

To illustrate the developed method, at first, the general framework applying a collocation scheme (i.e., the trapezoidal integration) on the expansion coefficients is introduced in Section 5.1. Therefore, a differential equation for the expansion coefficients, based on the physical state derivatives, is derived. An intuition on the introduced method is then given in Section 5.2.

Take into account that initial ideas on using gPC in a large-scale OCP were already proposed in e.g., [80, 94]: Compared to the thesis, these studies use a pseudo-spectral OC transcription, do not provide the derivation of Jacobian and Hessian analytically, and, most importantly, do not propagate the gPC expansion coefficient in the collocation scheme directly. These developments are made by this thesis and thus, available methods are extended. It should further be noted that an initial framework of this kind was published by the author in [102], which did not yet calculate the collocation defect (CD) based on the expansion states, but on the physical SC state derivatives. Additionally, the control history was not yet expanded.

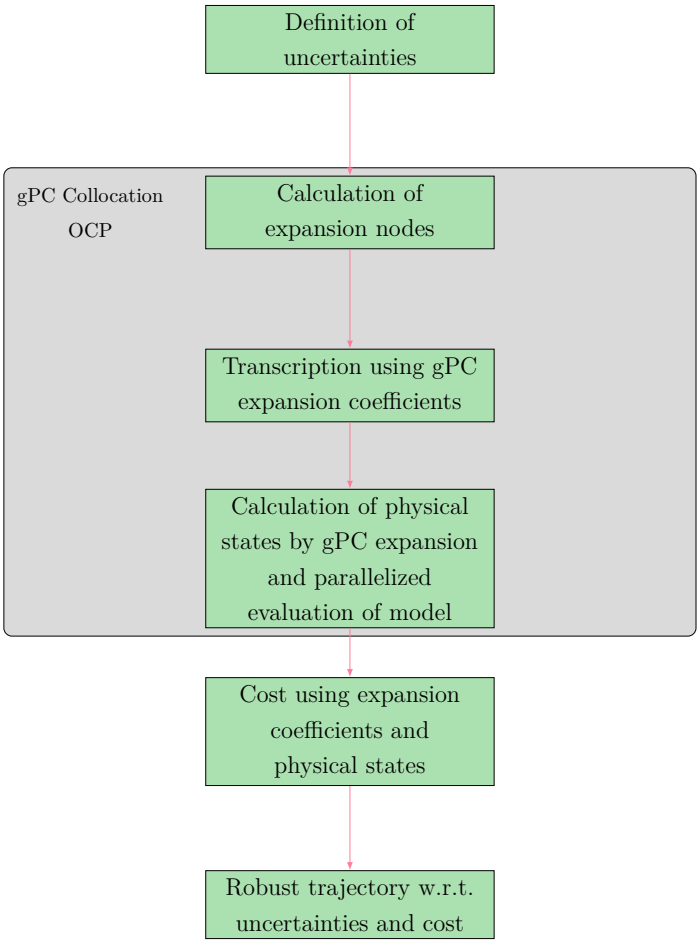

Figure 5.1: *Structure of optimal control framework with generalized polynomial chaos transcription methodology.*

5.1 Transcription Using Expansion States in Collocation Defect

At first, the gPC collocation OC parameter ("decision variable") vector is extended compared to the standard OC decision variable vector in (2.6) as follows (note that the final time is explicitly introduced to make the following derivations clearer):

$$\begin{aligned}\hat{\mathbf{z}} = \Big[\hat{t_f}^{(0)}, \dots \hat{t_f}^{(M-1)},\ \hat{\mathbf{p}}^{(0)}, \dots, \hat{\mathbf{p}}^{(M-1)},\ \hat{\mathbf{x}}_1^{(0)}, \dots, \hat{\mathbf{x}}_1^{(M-1)},\ \hat{\mathbf{u}}_1^{(0)}, \dots, \hat{\mathbf{u}}_1^{(M-1)}, \dots, \\ \hat{\mathbf{x}}_n^{(0)}, \dots, \hat{\mathbf{x}}_{n_\tau}^{(M-1)},\ \hat{\mathbf{u}}_n^{(0)}, \dots, \hat{\mathbf{u}}_{n_\tau}^{(M-1)}\Big]^\mathsf{T} \in \mathbb{R}^{n_{\hat{z}}}\end{aligned} \tag{5.1}$$

Take into account that $t_0 = 0 = \text{const.}$ can be assumed, like in Subsection 2.1.2.1, without loss of generality and is done in this chapter for the sake of brevity. Additionally, the sizes of the vector is $n_{\hat{z}} = (n_x + n_u) \cdot M \cdot n_\tau + n_p \cdot M$. It should be further noted that (5.1) is an extension of the decision variable vector already introduced by the author in [102], which did not yet expand the control history and optimizable parameters. Thus, (5.1) provides a generalization of the work in [102]. Compared to the definition in [102], the decision variable vector in (5.1) has the benefit that e.g., non-expanded as well as an expanded control history can be calculated. A non-expanded control history can e.g., be used to calculate a single, robust control command such that only a single control history must be applied in an application and the dynamic system follows the optimal trajectory under uncertainty. This is a major benefit compared to the distributed open-loop direct optimal control problem (DOCP) that always calculates a distributed, i.e., expanded, control history as the optimal control history varies for each SC node. Therefore, e.g., large deviations in the control history can occur. A drawback of the non-expanded control history is that it might be very conservative, because all uncertainty combinations must fulfill the constraints by one control command, which might even lead to infeasibilities. In these cases, and especially when an online estimation of the uncertainty (e.g., by monitoring the dynamic system or wind estimation) can be made, an expanded control history provides the best option as an online update of the command can be made by the gPC expansion. Thus, the expanded control history is generally less conservative and might improve optimality.

The benefits of a single robust control history were initially studied by the author in [102] and showed good applicability in the context of robust open-loop direct optimal control (ROC). More specifically, the single control history ensures that no updates of the robust trajectory dependent on the measurement of the random variable (RV) are required in e.g., onboard applications. This makes the usage straightforward and easy to implement. Still, some ROCP formulations might require an expanded control history due to e.g., constraints that cannot be fulfilled by only a single control history. Then, measuring the RVs might be required to apply the robust trajectory in e.g., an onboard

application, because this enables to update the expanded control history. Thus, depending on the application and the problem formulation, the proposed framework gives the user the opportunity to choose a desired representation.

Further take into account that, compared to the ROCP used for the DOC in (4.1), the decision variable vector in (5.1) does not grow as fast as the SC expansion vector with the expansion order and the number of uncertainties applied. This is based on the fact that the gPC expansion is evaluated on a reduced tensor grid (Definition 2.3), as there is no accuracy improvement by applying the full tensor grid, which is required for the SC. Thus, the proposed gPC collocation also remains applicable for larger OCPs.

It should be further noted that (5.1) is written for this tensor grid collocation. Thus, the following derivations are also made for a tensor grid (Definition 2.3). The implementation of sparse grids is also possible by a suitable adaption of the optimization parameter vector according to the used sparse grid method (e.g., Definition 2.4), i.e., considering the combination of lower order tensor grids. However, the derivation is more complex in terms of writing and skipped within this thesis for the sake of clarity and brevity. Nonetheless, a sparse grid gPC collocation methods can also be implemented following the structure of this section.

Remember that in (5.1), the incorporation of the expansion coefficients for states and controls as opposed to directly using the physical states and controls yields the possibility to directly optimize the statistical moments for the trajectory and the control effort (e.g., by (2.69) and (2.70)). Additionally, the introduction of CCs by sampling the gPC expansion becomes possible. The physical state and control history, required to evaluate the dynamic model, is calculated by using the gPC expansion (2.61). This is valid as long as the expansion order is large enough and thus an accurate analytic representation of the system by the gPC expansion is available. Using the dynamic model, the state derivatives and output values at the SC nodes can also be obtained, and thus, a minimization of their statistical moments is also possible by using (2.68) to obtain their expansion coefficients. Thus, (5.1) basically describes a change of variables used to describe the OCP. This change of variables is directly given by the gPC expansion formula in (2.61), i.e., by the projection using the orthogonal polynomials, and its inverse.

Then, the residual vector in (2.11) is expanded according to (5.1) as well, to have a CD for each expansion coefficient and a path constraint for each of the SC nodes as follows:

$$\begin{aligned}\hat{\mathbf{F}} = \Big[J, \mathbf{y}_1^{(1)}, \dots, \mathbf{y}_1^{(Q)}, \widehat{\mathbf{CD}}_{1,2}^{(0)}, \dots, \widehat{\mathbf{CD}}_{1,2}^{(M-1)}, \dots, \widehat{\mathbf{CD}}_{n_\tau-1,n_\tau}^{(0)}, \dots, \\ \widehat{\mathbf{CD}}_{n_\tau-1,n_\tau}^{(M-1)}, \dots, \mathbf{y}_{n_\tau}^{(1)}, \dots, \mathbf{y}_{n_\tau}^{(Q)}\Big]^\mathsf{T} \in \mathbb{R}^{n_{\hat{F}}}\end{aligned} \tag{5.2}$$

It should be noted that the cost function is not expanded here, i.e., a robust cost value instead of a statistical representation of the cost value is calculated. Still, the following derivations can directly be extended to a distributed cost index as well. Additionally, the size of the vector is $n_{\hat{F}} = n_x \cdot M \cdot (n_\tau - 1) + n_y \cdot n_\tau \cdot Q + 1$. It is also important to note that

the outputs are constrained at the SC nodes ("physical values"), i.e., $j = 1, \ldots, Q$, while the CD is calculated within the gPC (expansion coefficient) domain, i.e., $m = 0, \ldots, M-1$. This shows the desired mixture of the gPC and SC domain that is used to efficiently solve the ROCP and is in contrast to the optimization parameter vector in (5.1), which is only defined for $m = 0, \ldots, M-1$. This distinction is important for the ROCP and continued while deriving the gradient (Jacobian; Subsection 5.1.1) and the Hessian (Subsection 5.1.2) in the following.

Further, take into account that the path constraints in (5.2) may also contain the constraints for the physical states and controls at each SC node. This might be necessary as the optimization parameter vector contains the expansion coefficients, but does not include the physical values that are normally bounded in this vector. Now, the calculated trajectories may no longer make physical sense if it is not possible to meaningfully bound the expansion coefficients. Therefore, constraining the physical state and control history in the residual vector, instead of the expansion coefficients in the decision vector, is generally required and also used in this thesis. In this context, it should further be noted that it might not be possible, desired, or even necessary to constrain all physical trajectories at each SC node (i.e., from $j = 1, \ldots, Q$). This is due to the fact that there cannot be more constraints than decision variables, i.e., optimization parameters. In such cases, it is therefore also possible to only constrain a subset of the Q SC trajectories to achieve a physical trajectory representation by the gPC collocation scheme.

Continuing the derivation, contrary to the basic discretization in Subsection 2.1.2.1, the CD in (2.10) is redefined for the transcription method given by (5.1) and (5.2) directly using the state expansion coefficients and their derivatives as follows:

$$\widehat{\mathbf{CD}}_{i,i+1}^{(m)} = \hat{\mathbf{x}}_{i+1}^{(m)} - \hat{\mathbf{x}}_{i}^{(m)} - \frac{\bar{t_f}}{2} h_\tau \left(\dot{\hat{\mathbf{x}}}_{i+1}^{(m)} + \dot{\hat{\mathbf{x}}}_{i}^{(m)} \right) \stackrel{!}{=} 0 \in \mathbb{R}^{n_x}, \quad m = 0, \ldots, M-1 \tag{5.3}$$

Take into account that (5.3) directly allows the NLP algorithm to control the integration error of the expansion coefficients and therefore, the statistical moments. This control of the statistical moment accuracy is generally desired in ROC as it can enhance the statistical results, their significance, as well as the confidence in them. It is further important to note that the scaling using the final time is denoted here by $\bar{t_f}$, which is a yet to be determined quantity. In general, it is already known that the time transformation for the physical state derivatives is given by the physical time in (2.8). Thus, the derivation in the following also introduces the physical time at each of the SC nodes to weigh the time derivatives of the states.

To calculate the CD given in (5.3), at first a differential equation for the time development of the expansion coefficients is required. Therefore, the coefficient calculation by the SC (see (2.68)) for a general output variable must be differentiated with respect to time as follows (sensitive parameters are left out for the sake of readability):

$$\frac{\partial \hat{\mathbf{y}}^{(m)}(\mathbf{z})}{\partial t} = \dot{\hat{\mathbf{y}}}^{(m)}(\mathbf{z}) = \sum_{j=1}^{Q} \dot{\mathbf{y}}\left(\mathbf{z}; \boldsymbol{\theta}^{(j)}\right) \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}, \quad m = 0, \ldots, M-1 \tag{5.4}$$

Take into account that (5.4) is valid as long as the uncertainty is not changing over time and thus, the orthogonal polynomials as well as the GAUSSIAN QUADRATURE weights can be considered constant. If a time-varying uncertainty is used, the chain rule can be applied to extend (5.4) accordingly. It should be further noted that (5.4) contains the desired expansion state derivative by using a linear output mapping in (2.12) with the state derivatives of the model.

Thus, using (5.4) and the definition of the state dynamics, the CD in (5.3) can be reformulated as follows:

$$\begin{aligned}\widehat{\mathbf{CD}}_{i,i+1}^{(m)} = \hat{\mathbf{x}}_{i+1}^{(m)} - \hat{\mathbf{x}}_{i}^{(m)} - \frac{h_\tau}{2} \cdot \\ \cdot \left[\sum_{j=1}^{Q} \overbrace{t_f^{(j)} \underbrace{\dot{\mathbf{x}}_{i+1}^{(j)}}_{\mathbf{f}^{(j)}\left(\mathbf{x}_{i+1}^{(j)}, \mathbf{u}_{i+1}^{(j)}; \boldsymbol{\theta}^{(j)}\right)}}^{\left(\mathbf{x}_{i+1}^{(j)}\right)'} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} + \sum_{j=1}^{Q} \overbrace{t_f^{(j)} \underbrace{\dot{\mathbf{x}}_{i}^{(j)}}_{\mathbf{f}^{(j)}\left(\mathbf{x}_{i}^{(j)}, \mathbf{u}_{i}^{(j)}; \boldsymbol{\theta}^{(j)}\right)}}^{\left(\mathbf{x}_{i}^{(j)}\right)'} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right] \stackrel{!}{=} 0\end{aligned} \tag{5.5}$$

Take into account that the final time influences each of the addends within the gPC expansion individually, as each state derivative must be scaled properly with its corresponding final time at the SC nodes to achieve the desired non-dimensional representation. Additionally, it is assumed that orthonormal polynomials are used and therefore, the normalization constant is not specifically included in the formulas.

Then, the discretization by (5.1), (5.2), and (5.5) yields an NLP, extending the original continuous OCP in (2.1), as follows:

$$
\begin{aligned}
&\min_{\hat{\mathbf{z}}} \quad J \\
&\text{s.t.} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \hat{\mathbf{z}}_{\text{lb}} \leq \hat{\mathbf{z}} \leq \hat{\mathbf{z}}_{\text{ub}}, \\
&\mathbf{c}(\hat{\mathbf{z}};\boldsymbol{\theta},\mathbf{q}) = \begin{bmatrix}
\mathbf{y}_{1,\text{lb}}^{(j)} - \mathbf{y}_1^{(j)} \\
\mathbf{y}_1^{(j)} - \mathbf{y}_{1,\text{ub}}^{(j)} \\
\mathbf{x}_{1,\text{lb}} - \mathbf{x}_1^{(j)} = \mathbf{x}_{1,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{x}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{x}_1^{(j)} - \mathbf{x}_{1,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{x}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{x}_{1,\text{ub}} \\
\mathbf{u}_{1,\text{lb}} - \mathbf{u}_1^{(j)} = \mathbf{u}_{1,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{u}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{u}_1^{(j)} - \mathbf{u}_{1,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{u}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{u}_{1,\text{ub}} \\
\vdots \\
\mathbf{y}_{n_\tau,\text{lb}}^{(j)} - \mathbf{y}_{n_\tau}^{(j)} \\
\mathbf{y}_{n_\tau}^{(j)} - \mathbf{y}_{n_\tau,\text{ub}}^{(j)} \\
\mathbf{x}_{n_\tau,\text{lb}} - \mathbf{x}_{n_\tau}^{(j)} = \mathbf{x}_{n_\tau,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{x}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{x}_{n_\tau}^{(j)} - \mathbf{x}_{n_\tau,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{x}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{x}_{n_\tau,\text{ub}} \\
\mathbf{u}_{n_\tau,\text{lb}} - \mathbf{u}_{n_\tau}^{(j)} = \mathbf{u}_{n_\tau,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{u}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{u}_{n_\tau}^{(j)} - \mathbf{u}_{n_\tau,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{u}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{u}_{n_\tau,\text{ub}}
\end{bmatrix} \leq \mathbf{0}, \\
&\boldsymbol{\psi}(\hat{\mathbf{z}};\boldsymbol{\theta},\mathbf{q}) = \begin{bmatrix}
\hat{\mathbf{x}}_2^{(m)} - \hat{\mathbf{x}}_1^{(m)} - \frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_2^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right. \\
\qquad \left. + \sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_1^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \\
\vdots \\
\hat{\mathbf{x}}_{n_\tau}^{(m)} - \hat{\mathbf{x}}_{n_\tau-1}^{(m)} - \frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_{n_\tau}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right. \\
\qquad \left. + \sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_{n_\tau-1}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right]
\end{bmatrix} = \mathbf{0}
\end{aligned}
\tag{5.6}
$$

It is imminent from (5.6) that the physical trajectories $\mathbf{x}_i^{(j)}$ and $\mathbf{u}_i^{(j)}$, obtained from the gPC expansion, are, as mentioned before, constrained to achieve physical results.

It should be noted that the NLP in (5.6) is, as already mentioned, an extension of the work presented by the author in [102]. In comparison to [102], (5.6) uses the CD for the expansion state derivative rather than the physical state derivative. Thus, a structure is used that completely remains in the gPC domain and therefore, a development compared to [102] is made.

It is further important to note that the cost function in (2.14) is extended as well for (5.6) by using the expansion coefficients as follows:

$$J = e\left(\hat{\mathbf{z}}_{n_\tau}; \mathbf{q}\right) + h_\tau \frac{\hat{t_f}^{(0)}}{2} \sum_{i=1}^{n_\tau - 1} \left[L\left(\hat{\mathbf{z}}_i; \mathbf{q}\right) + L\left(\hat{\mathbf{z}}_{i+1}; \mathbf{q}\right)\right] \tag{5.7}$$

This allows e.g., the optimization of statistical moments, available from the gPC expansion coefficients, in the cost functional. Additionally, (5.6) generally allows to constrain statistical moments using the gPC expansion coefficients appropriately in the constraints.

For the efficient solution of the NLP in (5.6), the required Jacobian and Hessian are derived in the following subsections.

5.1.1 Derivation of Jacobian

For the Jacobian, the directional derivative of (5.4) for the expansion state derivatives, with respect to the physical states at the SC nodes, is given as follows:

$$\frac{\partial \dot{\hat{\mathbf{x}}}^{(m)}(\mathbf{z})}{\partial \mathbf{x}^{(j)}} = \mathbf{J}_{\mathbf{x}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \tag{5.8}$$

Take into account that the directional derivative with respect to the controls is calculated analogously.

Additionally, the derivative of the gPC expansion formula in (2.52), with respect to the expansion coefficients, is calculated as follows:

$$\frac{\partial \mathbf{y}\left(\mathbf{z}; \boldsymbol{\theta}\right)}{\partial \hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)} \approx \frac{\partial}{\partial \hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)} \left(\sum_{m=0}^{M-1} \hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right) \Phi^{(m)}\left(\boldsymbol{\theta}\right) \right) = \Phi^{(m)}\left(\boldsymbol{\theta}\right) \tag{5.9}$$

Then, in order to calculate the Jacobian of the CD with respect to the expansion states, it must be considered whether the derivative is calculated with respect to the same gPC expansion order m or a different expansion order n yielding to two different cases. In the following formulas, these different cases are directly evaluated applying (5.8) and (5.9):

$$\frac{\partial \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_i^{(m)}} = -\mathbf{I} - \frac{h_\tau}{2} \left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{J}_{\mathbf{x}_i}^{(j)} \left[\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\right]^2 \alpha^{(j)} \right] \tag{5.10a}$$

$$\frac{\partial \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_{i+1}^{(m)}} = \mathbf{I} - \frac{h_\tau}{2} \left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{J}_{\mathbf{x}_{i+1}}^{(j)} \left[\Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right)\right]^2 \alpha^{(j)} \right] \tag{5.10b}$$

$$\frac{\partial \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_i^{(n)}} = -\frac{h_\tau}{2} \left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{J}_{\mathbf{x}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right], \quad n \neq m \tag{5.10c}$$

$$\frac{\partial \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_{i+1}^{(n)}} = -\frac{h_\tau}{2} \left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{J}_{\mathbf{x}_{i+1}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right], \quad n \neq m \tag{5.10d}$$

$$\frac{\partial \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{u}}_i^{(n)}} = -\frac{h_\tau}{2} \left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{J}_{\mathbf{u}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right] \tag{5.10e}$$

$$\frac{\partial \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{u}}_{i+1}^{(n)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{J}_{\mathbf{u}_{i+1}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.10f}$$

Furthermore, the path constraint Jacobian is calculated similar to (2.27a) and (2.27b), using (5.9) and the chain rule as follows:

$$\frac{\partial \mathbf{y}_i^{(j)}}{\partial \hat{\mathbf{x}}_i^{(m)}} = \frac{\partial \mathbf{y}_i^{(j)}}{\partial \mathbf{x}_i^{(j)}} \frac{\partial \mathbf{x}_i^{(j)}}{\partial \hat{\mathbf{x}}_i^{(m)}} = \mathbf{J}_{\mathbf{y},\mathbf{x}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \tag{5.11a}$$

$$\frac{\partial \mathbf{y}_i^{(j)}}{\partial \hat{\mathbf{u}}_i^{(m)}} = \frac{\partial \mathbf{y}_i^{(j)}}{\partial \mathbf{u}_i^{(j)}} \frac{\partial \mathbf{u}_i^{(j)}}{\partial \hat{\mathbf{u}}_i^{(m)}} = \mathbf{J}_{\mathbf{y},\mathbf{u}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \tag{5.11b}$$

Additionally, it must be considered that each of the derivatives is depending on the SC trajectory final times, because the addends in the gPC expansion sum are scaled by those. This is due to the fact that each of the SC nodes must be evaluated on its specific non-dimensional time grid scaled, which is calculated by the physical times. The CD in (5.5) is thus expanded by introducing the gPC expansion for the final time at the SC nodes (see (2.61)) as follows:

$$\begin{aligned}\widehat{\mathbf{CD}}_{i,i+1}^{(m)} = \hat{\mathbf{x}}_{i+1}^{(m)} - \hat{\mathbf{x}}_i^{(m)} - \frac{h_\tau}{2} \\ \left[\sum_{j=1}^{Q}\left(\sum_{n=0}^{M-1} \hat{t_f}^{(n)} \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right)\right) \dot{\mathbf{x}}_{i+1}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right. \\ \left. + \sum_{j=1}^{Q}\left(\sum_{n=0}^{M-1} \hat{t_f}^{(n)} \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right)\right) \dot{\mathbf{x}}_i^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \stackrel{!}{=} 0\end{aligned} \tag{5.12}$$

Then, the derivative of (5.12) with respect to the final time expansion coefficient is given as follows:

$$\begin{aligned}\frac{\partial \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{t_f}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \dot{\mathbf{x}}_{i+1}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right. \\ \left. + \sum_{j=1}^{Q} \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \dot{\mathbf{x}}_i^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right]\end{aligned} \tag{5.13}$$

Overall the Jacobian sparsity structure and the corresponding equations are shown in Table 5.1. The table shows that the sparse and banded structure of the original trapezoidal collocation scheme (Table 2.1) is conserved, which is beneficial for the NLP algorithm because efficient methods to solve equation systems with sparse matrices exist [35].

Table 5.1: *Sparsity pattern of generalized polynomial chaos collocation transcription Jacobian* $\hat{\mathbf{J}} = \frac{\mathrm{d}\hat{\mathbf{F}}}{\mathrm{d}\hat{\mathbf{z}}}$ *using the expansion states to calculate the collocation defect.*

G	$\hat{t}_f^{(0)}$	…	$\hat{t}_f^{(M-1)}$	$\hat{\mathbf{x}}_1^{(0)}$	…	$\hat{\mathbf{x}}_1^{(M-1)}$	$\hat{\mathbf{u}}_1^{(0)}$	…	$\hat{\mathbf{u}}_1^{(M-1)}$	$\hat{\mathbf{x}}_2^{(0)}$	…	$\hat{\mathbf{x}}_2^{(M-1)}$	$\hat{\mathbf{u}}_2^{(0)}$	…	$\hat{\mathbf{u}}_2^{(M-1)}$	$\hat{\mathbf{x}}_3^{(0)}$	…	$\hat{\mathbf{x}}_3^{(M-1)}$	$\hat{\mathbf{u}}_3^{(0)}$	…	$\hat{\mathbf{u}}_3^{(M-1)}$
J	*	…	*	*	…	*	*	…	*	*	…	*	*	…	*	*	…	*	*	…	*
$\mathbf{y}_1^{(1)}$	0	…	0	(5.11a)	…	(5.11a)	(5.11b)	…	(5.11b)	0	…	0	0	…	0	0	…	0	0	…	0
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\mathbf{y}_1^{(Q)}$	0	…	0	(5.11a)	…	(5.11a)	(5.11b)	…	(5.11b)	0	…	0	0	…	0	0	…	0	0	…	0
$\widehat{\mathbf{CD}}_{1,2}^{(1)}$	(5.13)	…	(5.13)	(5.10a)	…	(5.10c)	(5.10e)	…	(5.10e)	(5.10b)	…	(5.10d)	(5.10f)	…	(5.10f)	0	…	0	0	…	0
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\widehat{\mathbf{CD}}_{1,2}^{(Q)}$	(5.13)	…	(5.13)	(5.10c)	…	(5.10a)	(5.10e)	…	(5.10e)	(5.10d)	…	(5.10b)	(5.10f)	…	(5.10f)	0	…	0	0	…	0
$\mathbf{y}_2^{(1)}$	0	…	0	0	…	0	0	…	0	(5.11a)	…	(5.11a)	(5.11b)	…	(5.11b)	0	…	0	0	…	0
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\mathbf{y}_2^{(Q)}$	0	…	0	0	…	0	0	…	0	(5.11a)	…	(5.11a)	(5.11b)	…	(5.11b)	0	…	0	0	…	0
$\widehat{\mathbf{CD}}_{2,3}^{(1)}$	(5.13)	…	(5.13)	0	…	0	0	…	0	(5.10a)	…	(5.10c)	(5.10e)	…	(5.10e)	(5.10b)	…	(5.10c)	(5.10e)	…	(5.10e)
⋮	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	…
$\widehat{\mathbf{CD}}_{2,3}^{(Q)}$	(5.13)	…	(5.13)	0	…	0	0	…	0	(5.10c)	…	(5.10a)	(5.10e)	…	(5.10e)	(5.10d)	…	(5.10b)	(5.10e)	…	(5.10e)
$\mathbf{y}_3^{(1)}$	0	…	0	0	…	0	0	…	0	0	…	0	0	…	0	(5.11a)	…	(5.11a)	(5.11b)	…	(5.11b)
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\mathbf{y}_3^{(Q)}$	0	…	0	0	…	0	0	…	0	0	…	0	0	…	0	(5.11a)	…	(5.11a)	(5.11b)	…	(5.11b)

5.1.2 Derivation of Hessian

This section deals with the calculation of the Hessian for the gPC collocation using the expansion coefficients within the CD. Thus, at first the second order derivative of the expansion state derivative given in (5.8) is required. It is calculated as follows (assuming that Schwarz's theorem [5, p. 800] can be applied, which is the case as smooth functions are required for the OC theory):

$$\begin{aligned}
\frac{\partial^2 \dot{\hat{\mathbf{x}}}^{(m)}(\mathbf{z})}{\partial \left(\mathbf{x}^{(j)}\right)^2} &= \mathbf{H}_{\mathbf{x}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \\
\frac{\partial^2 \dot{\hat{\mathbf{x}}}^{(m)}(\mathbf{z})}{\partial \mathbf{x}^{(j)} \partial \mathbf{u}^{(j)}} &= \mathbf{H}_{\mathbf{xu}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} = \frac{\partial^2 \dot{\hat{\mathbf{x}}}^{(m)}(\mathbf{z})}{\partial \mathbf{u}^{(j)} \partial \mathbf{x}^{(j)}} \\
\frac{\partial^2 \dot{\hat{\mathbf{x}}}^{(m)}(\mathbf{z})}{\partial \left(\mathbf{u}^{(j)}\right)^2} &= \mathbf{H}_{\mathbf{u}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}
\end{aligned} \tag{5.14}$$

Using (5.14), the second derivative of the CD in (5.5) is calculated as follows (take into account that zero entries, e.g., cross-time index coupling, are directly omitted for simplicity of writing):

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_i^{(n)} \partial \hat{\mathbf{x}}_i^{(k)}} = -\frac{h_\tau}{2} \left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{H}_{\mathbf{x}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right] \tag{5.15a}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_{i+1}^{(n)} \partial \hat{\mathbf{x}}_{i+1}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{H}_{\mathbf{x}_{i+1}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.15b}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{u}}_{i}^{(n)} \partial \hat{\mathbf{u}}_{i}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{H}_{\mathbf{u}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.15c}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{u}}_{i+1}^{(n)} \partial \hat{\mathbf{u}}_{i+1}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{H}_{\mathbf{u}_{i+1}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.15d}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_{i}^{(n)} \partial \hat{\mathbf{u}}_{i}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{H}_{\mathbf{x}_i,\mathbf{u}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.15e}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{\mathbf{x}}_{i+1}^{(n)} \partial \hat{\mathbf{u}}_{i+1}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} {t_f}^{(j)} \mathbf{H}_{\mathbf{x}_{i+1},\mathbf{u}_{i+1}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.15f}$$

Then, the Hessian entries regarding the final time can also be calculated by using the structure given in (5.12), (5.13), and (5.14) as follows (zero derivatives are directly omitted):

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{t_f}^{(n)} \partial \hat{\mathbf{x}}_{i}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \mathbf{J}_{\mathbf{x}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.16a}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{t_f}^{(n)} \partial \hat{\mathbf{u}}_{i}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \mathbf{J}_{\mathbf{u}_i}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.16b}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{t_f}^{(n)} \partial \hat{\mathbf{x}}_{i+1}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \mathbf{J}_{\mathbf{x}_{i+1}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.16c}$$

$$\frac{\partial^2 \widehat{\mathbf{CD}}_{i,i+1}^{(m)}}{\partial \hat{t_f}^{(n)} \partial \hat{\mathbf{u}}_{i+1}^{(k)}} = -\frac{h_\tau}{2}\left[\sum_{j=1}^{Q} \Phi^{(n)}\left(\boldsymbol{\theta}^{(j)}\right) \mathbf{J}_{\mathbf{u}_{i+1}}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \Phi^{(k)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right] \tag{5.16d}$$

The sparsity structure of the Hessian is depicted in Table 5.2. It should be noted that Schwarz's theorem [5, p. 800] can again be assumed to hold. Similar to the Jacobian, the main changes in the OCP definition and solution come with the different methods of calculating the entries of the matrix (here by the CD), but likewise the structure is similar to the standard trapezoidal collocation.

Table 5.2: *Sparsity pattern of generalized polynomial chaos collocation transcription Hessian* $\widehat{\mathbf{H}} = \frac{\mathrm{d}^2\hat{\mathbf{F}}}{\mathrm{d}\hat{\mathbf{z}}^2}$ *using the expansion states to calculate the collocation defect with no specific cost influence.*

$\widehat{\mathbf{H}}$	$t_f^{(0)}$	…	$t_f^{(M-1)}$	$\hat{\mathbf{x}}_1^{(0)}$	…	$\hat{\mathbf{x}}_1^{(M-1)}$	$\hat{\mathbf{u}}_1^{(0)}$	…	$\hat{\mathbf{u}}_1^{(M-1)}$	$\hat{\mathbf{x}}_2^{(0)}$	…	$\hat{\mathbf{x}}_2^{(M-1)}$	$\hat{\mathbf{u}}_2^{(0)}$	…	$\hat{\mathbf{u}}_2^{(M-1)}$	$\hat{\mathbf{x}}_3^{(0)}$	…	$\hat{\mathbf{x}}_3^{(M-1)}$	$\hat{\mathbf{u}}_3^{(0)}$	…	$\hat{\mathbf{u}}_3^{(M-1)}$
$t_f^{(0)}$	0	…	0	(5.16a)	…	(5.16a)	(5.16b)	…	(5.16b)	(5.16a)	…	(5.16a)	(5.16b)	…	(5.16b)	(5.16a)	…	(5.16a)	(5.16b)	…	(5.16b)
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$t_f^{(M-1)}$	0	…	0	(5.16a)	…	(5.16a)	(5.16b)	…	(5.16b)	(5.16a)	…	(5.16a)	(5.16b)	…	(5.16b)	(5.16a)	…	(5.16a)	(5.16b)	…	(5.16b)
$\hat{\mathbf{x}}_1^{(0)}$	(5.16a)	…	(5.16a)	(5.15a)	…	(5.15a)	(5.15e)	…	(5.15e)	0	…	0	0	…	0	0	…	0	0	…	0
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\hat{\mathbf{x}}_1^{(M-1)}$	(5.16a)	…	(5.16a)	(5.15a)	…	(5.15a)	(5.15e)	…	(5.15e)	0	…	0	0	…	0	0	…	0	0	…	0
$\hat{\mathbf{u}}_1^{(0)}$	(5.16b)	…	(5.16b)	(5.15e)	…	(5.15e)	(5.15c)	…	(5.15c)	0	…	0	0	…	0	0	…	0	0	…	0
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\hat{\mathbf{u}}_1^{(M-1)}$	(5.16b)	…	(5.16b)	(5.15e)	…	(5.15e)	(5.15c)	…	(5.15c)	0	…	0	0	…	0	0	…	0	0	…	0
$\hat{\mathbf{x}}_2^{(0)}$	(5.16a)	…	(5.16a)	0	…	0	0	…	0	(5.15a)	…	(5.15a)	(5.15e)	…	(5.15e)	0	…	0	0	…	0
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\hat{\mathbf{x}}_2^{(M-1)}$	(5.16a)	…	(5.16a)	0	…	0	0	…	0	(5.15a)	…	(5.15a)	(5.15e)	…	(5.15e)	0	…	0	0	…	0
$\hat{\mathbf{u}}_2^{(0)}$	(5.16b)	…	(5.16b)	0	…	0	0	…	0	(5.15e)	…	(5.15e)	(5.15c)	…	(5.15c)	0	…	0	0	…	0
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\hat{\mathbf{u}}_2^{(M-1)}$	(5.16b)	…	(5.16b)	0	…	0	0	…	0	(5.15e)	…	(5.15e)	(5.15c)	…	(5.15c)	0	…	0	0	…	0
$\hat{\mathbf{x}}_3^{(0)}$	(5.16a)	…	(5.16a)	0	…	0	0	…	0	0	…	0	0	…	0	(5.15a)	…	(5.15a)	(5.15e)	…	(5.15e)
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\hat{\mathbf{x}}_3^{(M-1)}$	(5.16a)	…	(5.16a)	0	…	0	0	…	0	0	…	0	0	…	0	(5.15a)	…	(5.15a)	(5.15e)	…	(5.15e)
$\hat{\mathbf{u}}_3^{(0)}$	(5.16b)	…	(5.16b)	0	…	0	0	…	0	0	…	0	0	…	0	(5.15e)	…	(5.15e)	(5.15c)	…	(5.15c)
⋮	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…	…	⋱	…
$\hat{\mathbf{u}}_3^{(M-1)}$	(5.16b)	…	(5.16b)	0	…	0	0	…	0	0	…	0	0	…	0	(5.15e)	…	(5.15e)	(5.15c)	…	(5.15c)

5.2 Intuition on Generalized Polynomial Chaos Collocation Methods

An intuition on the viability of the proposed transcription method of Section 5.1 is given in the following. Additionally, an initial validation of the results by simulation is made. Furthermore, this section clarifies why an exchange of the physical states in the transcription from Subsection 2.1.2.1 is possible to the gPC expansion coefficients in Section 5.1. Therefore, a second order lag, defined as follows, is considered [21, p. 302ff]:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2 \cdot \zeta \cdot \omega_0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_0^2 \end{bmatrix} u \tag{5.17}$$

In (5.17), the two states are denoted by x_1 and x_2 respectively, while the control is given by u and in this case is a unit step at the initial time. The natural frequency is given by $\omega_0 = 10\frac{rad}{s}$ and the relative damping is ζ. The relative damping is the uncertain parameter in this case. It is defined by a GAUSSIAN probability density function (PDF) as follows:

$$\theta_1 = \zeta \sim \mathcal{N}(\mu = 0.5, \sigma = 0.1) \tag{5.18}$$

Then, there exist two options on how to obtain an uncertain representation of the system with a unit step in the control by simulation: The first one is the standard approach using (5.17) and simulating the system at the SC nodes, as introduced in Subsection 2.3.3. With the results of these simulations, the expansion coefficients are calculated according to (2.68) and a statistical analysis is made similar to Subsection 2.3.4. This is the standard approach often used within the research community [30, 58, 109, 129, 149]. In this case, the system in (5.17) is simulated within the physical state domain (see (5.18)).

A further, not that intuitive method, is the transformation of the problem to another set of state variables $\hat{x}_1$ and $\hat{x}_2$, i.e., the expansion coefficients. In this case, the transformation is given by the inverse of the gPC expansion in (2.52) and the new set of state variables are the expansion coefficients. In general, the following transformation relation, based on (2.52), holds:

$$\hat{x} \underset{\Phi^{-1}}{\overset{\Phi}{\rightleftharpoons}} x \tag{5.19}$$

This is possible as the gPC expansion is indeed, for a sufficiently large expansion order, an analytic representation of the system and therefore, converges to the true system response [144]. It should be noted that (5.19) also holds for the state derivatives as the uncertainty PDF is not time-varying:

$$\dot{\hat{x}} \underset{\Phi^{-1}}{\overset{\Phi}{\rightleftharpoons}} \dot{x} \tag{5.20}$$

The proposed approach is generally known as “intrusive” gPC method. Analogous to the first option, the dynamic system in (5.17) can be evaluated at the physical states. These physical states are consequently used to calculate the expansion states by the gPC expansion (see (2.52)). It should be noted that this change of variables for evaluating the system model distinguishes the proposed method from the intrusive gPC method, which would also directly evaluate the dynamic system in the expansion coefficient domain.

Thus, the usage of the gPC expansion coefficients is generally just a wrapper around the physical, deterministic system. Take into account that the integration algorithm indeed integrates the expansion coefficients. This is possible e.g., using the transformation given by the time derivative of (2.68). Here, a direct connection between the expansion coefficients and the state derivatives at the SC nodes is given using (5.4):

$$\hat{\dot{x}}_i^{(m)}(u) \approx \sum_{j=1}^{Q} \dot{x}_i\left(u;\zeta^{(j)}\right)\Phi^{(m)}\left(\zeta^{(j)}\right)\alpha^{(j)}, \qquad i=1,2 \tag{5.21}$$

Remember that this is possible as only the expansion coefficient history is time variable (because of the states and controls), while both the orthogonal polynomials as well as the integration weights are time-invariant. If a time-variable uncertainty is considered, (5.21) must be adapted by the chain rule accordingly.

Consequently, the transformed system can be written as follows, starting with the extended dynamic model in (5.17):

$$\begin{bmatrix} \dot{x}_1^{(1)} \\ \dot{x}_2^{(1)} \\ \vdots \\ \dot{x}_1^{(Q)} \\ \dot{x}_2^{(Q)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dots & 0 & 0 \\ -\omega_0^2 & -2\cdot\zeta^{(1)}\cdot\omega_0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & -\omega_0^2 & -2\cdot\zeta^{(Q)}\cdot\omega_0 \end{bmatrix} \cdot \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_1^{(Q)} \\ x_2^{(Q)} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_0^2 \\ \vdots \\ 0 \\ \omega_0^2 \end{bmatrix} u \tag{5.22}$$

Then, the state derivatives of the expansion coefficients can be directly calculated using (5.21) as follows:

$$\begin{bmatrix} \hat{\dot{x}}_1^{(0)} \\ \hat{\dot{x}}_2^{(0)} \\ \vdots \\ \hat{\dot{x}}_1^{(M-1)} \\ \hat{\dot{x}}_2^{(M-1)} \end{bmatrix} = \begin{bmatrix} \Phi^{(0)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \Phi^{(0)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \dots & \Phi^{(0)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} & \Phi^{(0)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} \\ \Phi^{(0)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \Phi^{(0)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \dots & \Phi^{(0)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} & \Phi^{(0)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Phi^{(M-1)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \Phi^{(M-1)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \dots & \Phi^{(M-1)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} & \Phi^{(M-1)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} \\ \Phi^{(M-1)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \Phi^{(M-1)}\left(\zeta^{(1)}\right)\alpha^{(1)} & \dots & \Phi^{(M-1)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} & \Phi^{(M-1)}\left(\zeta^{(Q)}\right)\alpha^{(Q)} \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_1^{(1)} \\ \dot{x}_2^{(1)} \\ \vdots \\ \dot{x}_1^{(Q)} \\ \dot{x}_2^{(Q)} \end{bmatrix} \tag{5.23}$$

Thus, the transformed dynamics equation, i.e., the equations in the expansion coefficient domain, can directly be used in the simulation by combining (5.22)–(5.23).

Following the argumentation of the previous paragraphs, it is clear why a transformation of the OC transcription from the physical states to the expansion coefficients is merely a change in the set of variables describing the problem. Although this transformation is at first more expensive in calculation, because the transformation in (5.21) needs to be made in e.g., each OC iteration, it opens the possibility to use the statistical moments within the optimization as an uncertain representation of the system is available. This is the overall goal of ROC.

Figure 5.2 and Figure 5.3 show the results for the standard simulation (solid blue; “Std”) and the simulation with the gPC expansion coefficients (dashed green; “gPC”) for a unit step in the control input. The gPC expansion is calculated using a third order expansion. Here Figure 5.2 depicts the mean values, while Figure 5.3 illustrates the standard deviation. It is seen that the results lie on top of each other. This shows the viability of the approach. Minor errors are related to the third order expansion and should reduce using higher

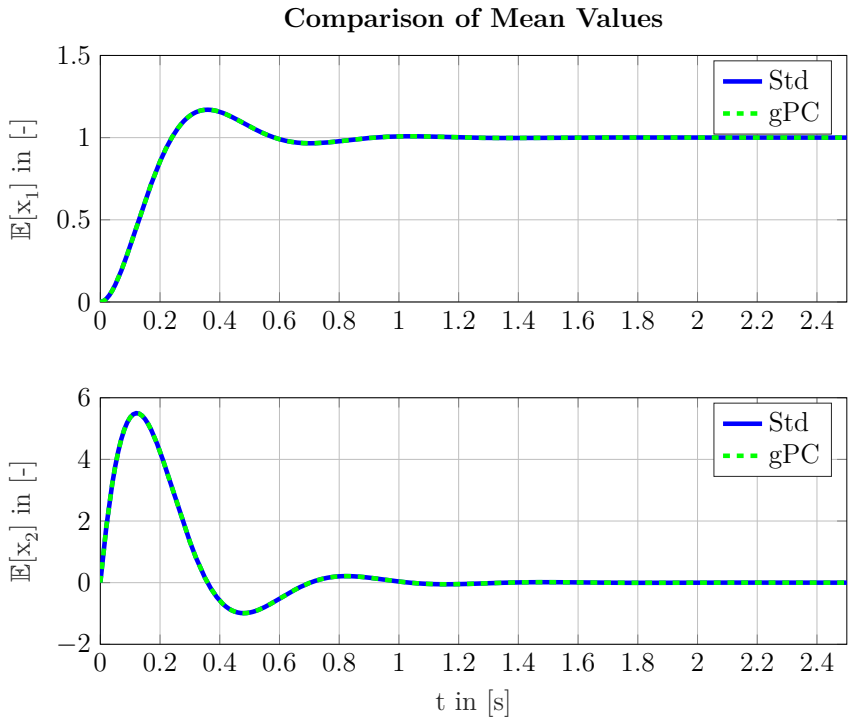

Figure 5.2: *Mean value comparison for PT2 element with standard ("physical" states) and generalized polynomial chaos ("expansion" states) simulation for a unit step.*

order expansions. Additionally, the errors are related to the different dynamics being integrated by the integration algorithm: Here, the Matrix Laboratory® (MATLAB®) ODE45 solver[1] is used as integration algorithm.

These minor errors are also visualized in Figure 5.4 that depicts the relative errors obtained for the mean values (Figure 5.2). The relative error is calculated as the difference error divided by the mean value of the standard simulation. The figure shows that the relative error in the transient and convergence phase (up until approximately $3s$) is negligibly small for the first state. The second state exhibits a larger error but mainly in the areas where the standard simulation is close to zero (Figure 5.2). Here, the definition of the relative error is problematic due to numerical instabilities encountered when dividing by a small value. Additionally, the error gets larger at sharp curvatures because of the fact that here errors introduced due to the smooth approximation by the gPC expansion become dominant. Nonetheless, the results show that the approximation is fairly good for most of the trajectory. This again strengthens the previously made statement that the gPC expansion coefficients are merely a change of the state variables.

Overall, the use of the gPC expansion shows their applicability for simulation purposes and thus, the introduction of the gPC expansion in OC applications is also valid. Therefore, the next chapter uses the introduced method to incorporate CCs in the ROCP.

[1]MATLAB ODE documentation (Retrieved April 23, 2019)

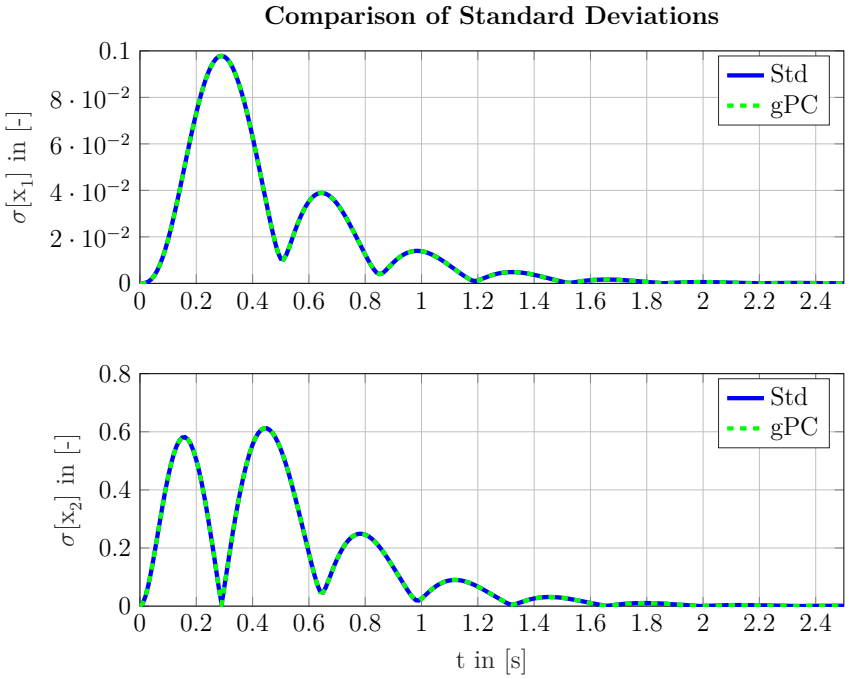

Figure 5.3: *Standard deviation comparison for PT2 element with standard ("physical" states) and generalized polynomial chaos ("expansion" states) simulation for a unit step.*

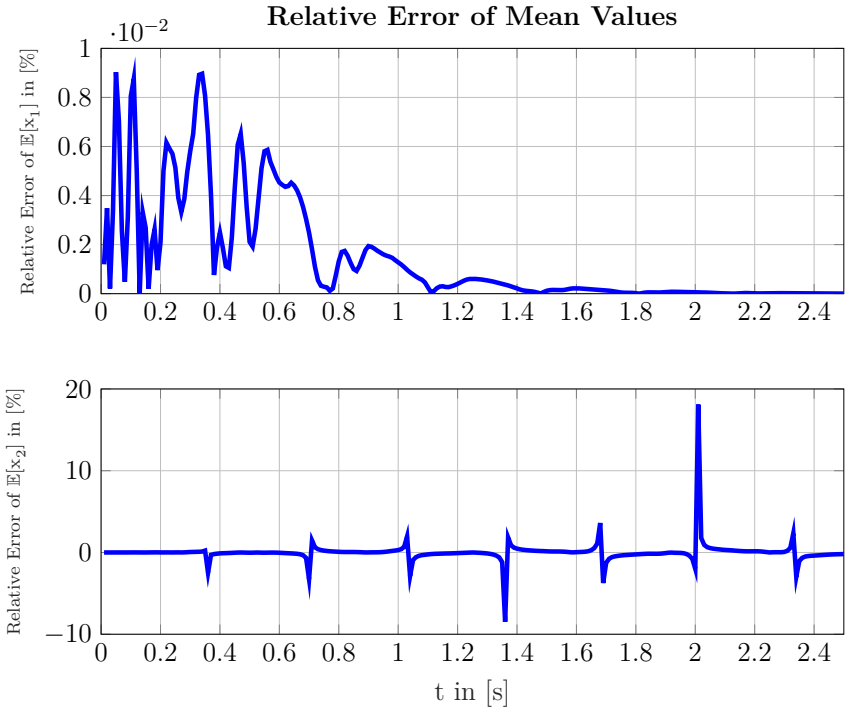

Figure 5.4: *Relative error of mean values for PT2 element between standard ("physical" states) and generalized polynomial chaos ("expansion" states) simulation for a unit step in percent (calculated as difference error related to the standard simulation result).*

Chapter 6

Chance-constrained Optimal Control with Polynomial Chaos

This chapter introduces the chance-constrained open-loop direct optimal control (CC-OC) framework developed in the thesis: Within this framework, methods to add probabilistic constraints, i.e., chance constraints (CCs), to the open-loop direct optimal control problem (OCP) are discussed. Thus, Contribution 4 and Contribution 5 of the thesis are covered by this chapter. The methods have the common goal to maintain a deterministic baseline OCP formulation with probabilistic robustifications provided by the CCs. Thus, the discretization method of the framework is based on e.g., the direct transcription methods using the generalized polynomial chaos (gPC) expansion coefficients introduced in Chapter 5 or the distributed open-loop direct optimal control (DOC) formulation in Chapter 4.

To show the incorporation of CCs in open-loop direct optimal control (OC), Section 6.1 introduces a CC-OC framework that is based on a Monte Carlo analysis (MCA) of the gPC expansion inside a constraint function, the CC, of the OCP. The framework can mainly be used for "frequent" events, but is very efficient to use in robust open-loop direct optimal control (ROC), because the sampling is based on a matrix-vector operation. A major part of this section is also dedicated to the introduction of a differentiable approximation of the exact CC by a smooth function, such that it can be used within the NEWTON-type nonlinear program (NLP) scheme (see Contribution 4).

Afterward, Section 6.2 incorporates the subset simulation (SubSim) method within the CC-OC framework, which makes it possible to calculate the failure probability of rare-events within the CC (Contribution 5). These rare-events are e.g., encountered in reliability and safety engineering. Specifically, this framework extends currently available methods in chance-constrained open-loop direct optimal control problem (CC-OCP).

Generally, an original work of this framework was already published by the author in [102]. This thesis operates on the basis of this work and extends the principles to cover more general cases by e.g., deriving the analytic Jacobian and Hessian of the multivariate sigmoid CC approximation (Subsection 6.1.2). Furthermore, this thesis generalizes

the incorporation and mathematical description of CCs within the robust open-loop direct optimal control problem (ROCP), e.g., by introducing confidence intervals (CIs) (Subsection 6.1.1). Finally, the capability of approximating and robustifying the rare-event failure probabilities by SubSim is extended by using the probability density function (PDF) estimation of the failure probability introduced in (2.46) (Section 6.2). Here, once more the derivation of the Jacobian and Hessian for the NEWTON-type NLP solver is made.

6.1 Sampling-based Chance-constrained Optimal Control

Within this section, the thesis looks at the introduction of the CC-OC framework based on a gPC approximation of the response surface (Contribution 4). It is assumed in this section and chapter that the expansion coefficients are directly used as decision variables ("gPC collocation"; Chapter 5) or can be calculated because all required stochastic collocation (SC) trajectories are available ("DOC framework"; Chapter 4). Thus, the expansion coefficients are available for the CC probability calculation by e.g., sampling the gPC expansion within each NLP iteration.

Applying this idea, the generic CC-OCP in (2.87) can be extended by e.g., applying the gPC collocation scheme in (5.6) as follows (it should be noted that the following derivation is made for the gPC collocation only for the sake of compactness; nonetheless the DOC formulation can be treated similarly):

$$
\begin{aligned}
\min_{\hat{\mathbf{z}}} \quad & J(\hat{\mathbf{z}};\mathbf{q}) \\
\text{s.t.} \quad & \hat{\mathbf{z}}_{\text{lb}} \leq \hat{\mathbf{z}} \leq \hat{\mathbf{z}}_{\text{ub}}, \\
& \mathbf{c}(\hat{\mathbf{z}};\mathbf{q}) = \begin{bmatrix}
\mathbf{y}_{1,\text{lb}}^{(j)} - \mathbf{y}_1^{(j)} \\
\mathbf{y}_1^{(j)} - \mathbf{y}_{1,\text{ub}}^{(j)} \\
\mathbf{x}_{1,\text{lb}} - \mathbf{x}_1^{(j)} = \mathbf{x}_{1,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{x}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{x}_1^{(j)} - \mathbf{x}_{1,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{x}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{x}_{1,\text{ub}} \\
\mathbf{u}_{1,\text{lb}} - \mathbf{u}_1^{(j)} = \mathbf{u}_{1,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{u}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{u}_1^{(j)} - \mathbf{u}_{1,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{u}}_1^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{u}_{1,\text{ub}} \\
\vdots \\
\mathbf{y}_{n_\tau,\text{lb}}^{(j)} - \mathbf{y}_{n_\tau}^{(j)} \\
\mathbf{y}_{n_\tau}^{(j)} - \mathbf{y}_{n_\tau,\text{ub}}^{(j)} \\
\mathbf{x}_{n_\tau,\text{lb}} - \mathbf{x}_{n_\tau}^{(j)} = \mathbf{x}_{n_\tau,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{x}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{x}_{n_\tau}^{(j)} - \mathbf{x}_{n_\tau,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{x}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{x}_{n_\tau,\text{ub}} \\
\mathbf{u}_{n_\tau,\text{lb}} - \mathbf{u}_{n_\tau}^{(j)} = \mathbf{u}_{n_\tau,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{u}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\
\mathbf{u}_{n_\tau}^{(j)} - \mathbf{u}_{n_\tau,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{u}}_{n_\tau}^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{u}_{n_\tau,\text{ub}}
\end{bmatrix} \leq \mathbf{0}, \\
& \boldsymbol{\psi}(\hat{\mathbf{z}};\boldsymbol{\theta},\mathbf{q}) = \begin{bmatrix}
\hat{\mathbf{x}}_2^{(m)} - \hat{\mathbf{x}}_1^{(m)} - \frac{h_\tau}{2}\left(\sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_2^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right. \\
\qquad \left. + \sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_1^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right) \\
\vdots \\
\hat{\mathbf{x}}_{n_\tau}^{(m)} - \hat{\mathbf{x}}_{n_\tau-1}^{(m)} - \frac{h_\tau}{2}\left(\sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_{n_\tau}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right. \\
\qquad \left. + \sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_{n_\tau-1}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)}\right)
\end{bmatrix} = \mathbf{0}, \\
& \begin{bmatrix}
\mathbb{P}\left[\left(\mathbf{y}_1(\hat{\mathbf{z}};\boldsymbol{\theta}) = \sum_{m=0}^{M-1} \hat{\mathbf{y}}_1^{(m)}(\hat{\mathbf{z}}) \Phi^{(m)}(\boldsymbol{\theta})\right) \notin \mathcal{F}\right] \\
\vdots \\
\mathbb{P}\left[\left(\mathbf{y}_{n_\tau}(\hat{\mathbf{z}};\boldsymbol{\theta}) = \sum_{m=0}^{M-1} \hat{\mathbf{y}}_{n_\tau}^{(m)}(\hat{\mathbf{z}}) \Phi^{(m)}(\boldsymbol{\theta})\right) \notin \mathcal{F}\right]
\end{bmatrix} \geq \boldsymbol{\xi}
\end{aligned}
\tag{6.1}
$$

Thus, the CC-OCP in (6.1) adds a probabilistic constraint $\mathbb{P}[\cdot]$, i.e., a CC, to the deterministic OC formulation that is used to robustify the optimal trajectory. The probability is calculated to not be in the failure set/domain $\mathcal{F}$ and must fulfill a minimum probability of ξ. These CCs are generally calculated using the gPC expansion and the efficient sampling that is possible by using it. The following sections elaborate on the procedure: Here, the general formulation of the CC in the deterministic OCP is introduced in Sub-

section 6.1.1. Afterward, Subsection 6.1.2 introduces a differentiable approximation of the exact CC and Subsection 6.1.3 concludes with a homotopy strategy to iteratively sharpen the developed differentiable CC representation.

6.1.1 Derivation of Chance Constraint Formulation

As stated in Subsection 2.2.1, MCA is a powerful, but computationally expensive tool for stochastic analysis purposes. Generally, it can be used to estimate the non-failure probability in (6.1) as the samples that are not within the failure domain can be counted and compared to the total number of samples. As already mentioned, this way of sampling is generally very expensive (in terms of computational time) because thousands of samples might be required for the central limit theorem (CLT) to converge and thus, to have a reasonably small CI (Subsection 2.2.3) for the MCA.

When using direct methods to solve ROCPs with NEWTON-type NLP solvers, a further issue is to provide sufficiently smooth cost and constraints with their Jacobians and Hessians to the NEWTON-type NLP solver. Especially, with some iterative MCA techniques, e.g., Metropolis-Hastings algorithm (MHA) or modified Metropolis-Hastings algorithm (MMHA) (Algorithm 2.2), which improve the convergence speed, the derivatives might not be trivial to calculate or approximate. This is due to the fact that these algorithms select samples adaptively, depending on whether or not they are e.g., in or closer to the failure set, to have an efficient approximation. As this selection process would be required in each iteration of the NLP, the samples and thus, the depending derivatives are adapted constantly as well, which would most likely yield to bad convergence. In general, this would therefore require the introduction of stochastic gradient updates within the NLP solver [4, 24], which would require major updates in the off-the-shelf solvers, such as Interior Point Optimizer (IPOPT) or Sparse Nonlinear Optimizer (SNOPT), used in this thesis.

Thus, this thesis uses a deterministic MCA sampling alternative that is differentiable and uses the same samples throughout the NLP solution process. Looking at the gPC expansion in (2.52), an approximation for any output quantity of the OCP with respect to a stochastic disturbance is directly available (this idea of sampling the gPC expansion was e.g., already used in [102, 139]). Thus, in cases where the expansion coefficients are available within the NLP, as e.g., in the ROCP of (6.1), the gPC expansion can be sampled for thousands of samples via a matrix-vector operation in a MCA-type way, but with improved efficiency due to the simple evaluation. The fact that thousands of samples can be used also removes the task to update the samples in each NLP step as "enough" samples to cover the CC/random variable (RV) evaluation domain can be chosen (both with respect to the CLT as well as CIs).

To illustrate the MCA character of the CC evaluation by sampling the gPC expansion, consider n_s random samples obtained from the PDFs of $\boldsymbol{\theta} \sim \rho_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$, labeled $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(n_s)}$. These samples yield corresponding samples, $\mathbf{y}^{(1)} = \mathbf{y}\left(\mathbf{z}; \boldsymbol{\theta}^{(1)}\right), \ldots, \mathbf{y}^{(n_s)} = \mathbf{y}\left(\mathbf{z}; \boldsymbol{\theta}^{(n_s)}\right)$, for the output $\mathbf{y}$, applying the gPC expansion in (2.61), as follows [102]:

$$\underbrace{\begin{bmatrix}\mathbf{y}^{(1)} & \ldots & \mathbf{y}^{(n_s)}\end{bmatrix}}_{\mathbb{R}^{n_y \times n_s}} = \underbrace{\begin{bmatrix}\hat{\mathbf{y}}^{(0)}(\mathbf{z}) & \ldots & \hat{\mathbf{y}}^{(M-1)}(\mathbf{z})\end{bmatrix}}_{\mathbb{R}^{n_y \times M}} \underbrace{\begin{bmatrix}\Phi^{(0)}\left(\boldsymbol{\theta}^{(1)}\right) & \ldots & \Phi^{(0)}\left(\boldsymbol{\theta}^{(n_s)}\right) \\ \vdots & \ddots & \vdots \\ \Phi^{(M-1)}\left(\boldsymbol{\theta}^{(1)}\right) & \ldots & \Phi^{(M-1)}\left(\boldsymbol{\theta}^{(n_s)}\right)\end{bmatrix}}_{\mathbb{R}^{M \times n_s}} \tag{6.2}$$

Thus, the output samples are provided from a simple matrix-vector operation on the expansion coefficients $\hat{\mathbf{y}}$. As mentioned, these are part of the ROCP formulation in (6.1) (either via the state and control expansion coefficients in the decision variable vector or by the SC trajectories in the DOC framework). Thus, thousands of random trajectories can be created and the failure probability can be easily calculated by checking the random trajectories for failures. It should be noted that (6.2) is the natural extension of (2.62).

Take into account that the derivative of (6.2) with respect to the expansion coefficients is generally required for the NEWTON-type NLP scheme and is given as follows:

$$\begin{bmatrix}\frac{\partial \mathbf{y}^{(1)}}{\partial \hat{\mathbf{y}}^{(m)}(\mathbf{z})} & \ldots & \frac{\partial \mathbf{y}^{(n_s)}}{\partial \hat{\mathbf{y}}^{(m)}(\mathbf{z})}\end{bmatrix} = \begin{bmatrix}\Phi^{(m)}\left(\boldsymbol{\theta}^{(1)}\right) & \ldots & \Phi^{(m)}\left(\boldsymbol{\theta}^{(n_s)}\right)\end{bmatrix} \tag{6.3}$$

Thus, the derivative of the output samples with respect to the gPC expansion is the orthogonal polynomial evaluated at this random parameter sample. This was also already introduced and used in (5.9), when deriving the gPC collocation scheme. With the output samples available from (6.2), the general equation for fulfilling the CC, i.e., not being in the failure domain, is as follows [102]:

$$\mathbb{P}\left[\mathbf{y}(\mathbf{z}; \boldsymbol{\theta}) \notin \mathcal{F}\right] = \int_{\boldsymbol{\Omega}} \mathcal{I}\left[\mathbf{y}(\mathbf{z}; \boldsymbol{\theta})\right] \rho_{\Theta}(\theta) \, \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{n_s} \sum_{i=1}^{n_s} \mathcal{I}\left[\mathbf{y}^{(i)}\right] \tag{6.4}$$

Here, (6.4) converges in the sense of the CLT (Section B.7) and $\mathcal{I}\left[\mathbf{y}(\mathbf{z}; \boldsymbol{\theta})\right]$ is the indicator function defined by [102]:

$$\mathcal{I}\left[\mathbf{y}(\mathbf{z}; \boldsymbol{\theta})\right] = \begin{cases} 1, & \text{if} \quad \mathbf{y}(\mathbf{z}; \boldsymbol{\theta}) \notin \mathcal{F} \\ 0, & \text{else} \end{cases} \tag{6.5}$$

Take into account that the probability is defined to not be in the failure domain, as e.g., opposed to other probabilistic methods like the SubSim approach in Subsection 2.2.4. These definitions are interchangeable, but the formulation to not be in the failure domain is preferred in this work as it generally yields a better conditioned NLP (mainly because the numeric probability values do not get too small, i.e., converge to zero, but remain close to one).

It should be further noted here that (6.4) provides one realization of the exact probability (i.e., a value that should be close to the exact probability, but is generally not exactly the same) due to e.g., the fact that only a finite amount of samples is used (this is due the CLT in Section B.7 and results in the CIs from Subsection 2.2.3). Additionally, the sampling of the gPC expansion in (6.2) only gives an approximation of the real, exact system response. Summarizing, the fact that (6.4) does not calculate the exact probability is based on the fact there are only limited information available to calculate it, i.e., it is consequently only approximated. Thus, as already mentioned for the SubSim in Subsection 2.2.4.4, the calculated probability can be treated as a RV, the relative plausibility/accuracy can be assessed, and statistical moments can be calculated accordingly (for SubSim e.g., according to [148, p. 292ff.]).

Consequently, for safety critical applications, i.e., applications where it must be assured to fulfill the CC, it can be meaningful to optimize the CI bounds rather than only the mean value in (6.4), even though the optimality might be reduced by this. Here, the CI for the mean value can be calculated as e.g., introduced in (2.38) or by a one-sided representation [81, p. 20f.] because in a CC-OCP formulation like in (6.1) the lower bound is generally of main interest.

For this CI, the standard deviation of the probability is required, which can be calculated as given in [81, p. 16]:

$$\sigma\left[\mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\notin\mathcal{F}\right]\right]=\frac{\mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\notin\mathcal{F}\right]\cdot\left(1-\mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\notin\mathcal{F}\right]\right)}{n_s-1} \tag{6.6}$$

With this estimation, the bounds of the CI can be used within the optimization to assure a threshold to the desired minimum probability level. Generally, this CI-based CC-OC procedure gives more certainty that the optimal trajectory is indeed fulfilling the desired CC, while it normally comes at the cost of getting a less optimal trajectory. This is the already mentioned trade-off between optimality and robustness in ROC.

It is important to note that the indicator function $\mathcal{I}$ in (6.5) is trivial to evaluate but non-differentiable. Thus, it can create difficulties when being used in the context of a NEWTON-type NLP algorithm. Therefore, a smooth approximation function $s\left[\cdot\right]$ of the indicator function in (6.5) is introduced to solve the ROCP. This smooth approximator should have the following properties [102]:

$$\begin{aligned}
&s\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\in\left[0;1\right]\\
&s\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\approx 1 \qquad &&\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\notin\mathcal{F}\\
&s\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\approx 0 \qquad &&\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\in\mathcal{F}
\end{aligned} \tag{6.7}$$

The functional choice and the properties of the smooth approximator are introduced in the next section.

6.1.2 Smooth Approximator for Chance Constraints

A group of smooth functions that can be used to approximate the indicator function in (6.5), which also fulfill the conditions given in (6.7), are the logistic functions. Using logistic functions as CC approximator is similar to e.g., the Bernstein polynomial approach proposed in [147], but logistic functions generally have easier to calculate derivatives, which is beneficial for OC. This approach was already introduced by the author in [102].

An example of the logistic functions is the sigmoid, which is defined as follows for a scalar system response $y^{(i)} = y(\mathbf{z}; \boldsymbol{\theta}^{(i)})$ (done here for the simplicity of writing, but extended to the multi-dimensional case in the following) obtained from (6.2) [102]:

$$s\left[y^{(i)}; a_s, b_s\right] = \frac{1}{\exp\left[-a_s \cdot (y^{(i)} - b_s)\right] + 1} \tag{6.8}$$

The parameters a_s and b_s are the scaling and offset parameter of the sigmoid respectively. These are used to shape the sigmoid in order to suitably approximate the desired CC domain. It should be noted that the use of a sigmoid is generally a conservative way of approximating CCs as the sigmoid does not reach an exact value of 1. Thus, the exact CC probability is generally underestimated if the sigmoid is sharp enough, i.e., has a large enough scaling. This is a natural behavior of most available approximator functions and might even be a desired feature to e.g., assure safety. It is also useful when applying the gPC expansion to approximate the system response, as introduced errors are reduced.

Naturally, the scalar sigmoid in (6.8) can be combined by e.g., multiplication, in order to approximate a desired (multi-dimensional) uncertainty interval. This is shown in Figure 6.1, which depicts the combination of two sigmoids by multiplication (solid blue) with one gradual descend (dashed green; number 1) and one steeper ascend (dashed red; number 2). Take into account that the following derivations are, once more, done for a scalar output variable at first for the sake of simplicity and afterward, extended to multiple dimension.

In order to calculate the non-failure probability by the sigmoid, the function values of the sigmoid must only be summed and divided by the number of samples taken:

$$\mathbb{P}\left[y \notin \mathcal{F}\right] \approx \frac{\sum_{i=1}^{n_s} s\left[y^{(i)}; a_s, b_s\right]}{n_s} \tag{6.9}$$

Furthermore, the sigmoid has a simple derivative that can be used to efficiently calculate the gradient necessary for the NLP optimizer as follows [65, p. 107]:

$$\begin{aligned}
\frac{\mathrm{d}s\left[y^{(i)}; a_s, b_s\right]}{\mathrm{d}y^{(i)}} &= \frac{-(-a_s)\exp\left[-a_s \cdot \left(y^{(i)} - b_s\right)\right]}{\left(\exp\left[-a_s \cdot (y^{(i)} - b_s)\right] + 1\right)^2} \\
&= \frac{a_s}{\exp\left[-a_s \cdot (y^{(i)} - b_s)\right] + 1} \cdot \frac{\left(\exp\left[-a_s \cdot \left(y^{(i)} - b_s\right)\right] + 1\right) - 1}{\exp\left[-a_s \cdot (y^{(i)} - b_s)\right] + 1} \\
&= a_s \cdot s\left[y^{(i)}; a_s, b_s\right] \cdot \left[1 - s\left[y^{(i)}; a_s, b_s\right]\right]
\end{aligned} \tag{6.10}$$

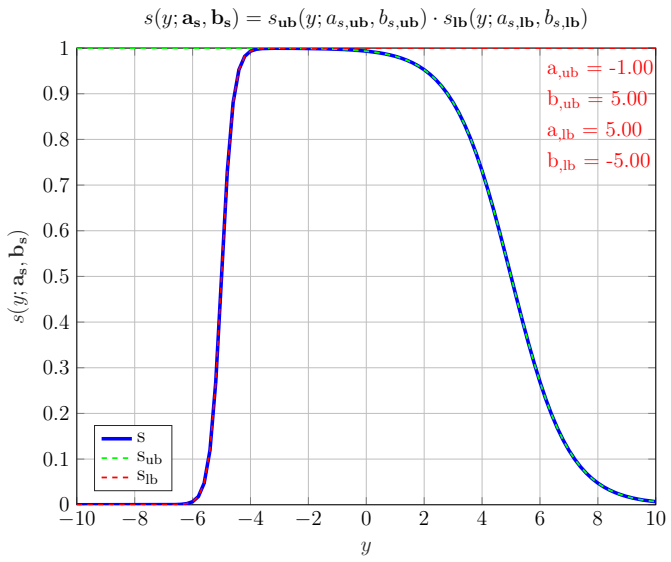

Figure 6.1: *Multiplication of two scalar sigmoids with different scaling and offset parameters to approximate a box bound chance constraint domain (after [102]).*

Additionally, the second derivative of the sigmoid, required for the Hessian, can also be calculated directly by applying the rule of the first derivative in (6.10) as follows:

$$\begin{aligned}\frac{\mathrm{d}^2 s\left[y^{(i)}; a_s, b_s\right]}{\mathrm{d}\left(y^{(i)}\right)^2} &= a_s \cdot \frac{\mathrm{d}s\left[y^{(i)}; a_s, b_s\right]}{\mathrm{d}y^{(i)}} \cdot \left[1 - s\left[y^{(i)}; a_s, b_s\right]\right] \\ &\quad + a_s \cdot s\left[y^{(i)}; a_s, b_s\right] \cdot \left[1 - \frac{\mathrm{d}s\left[y^{(i)}; a_s, b_s\right]}{\mathrm{d}y^{(i)}}\right] \\ &= {a_s}^2 \cdot s\left[y^{(i)}; a_s, b_s\right] \cdot \left[1 - s\left[y^{(i)}; a_s, b_s\right]\right] \cdot \left[1 - 2 \cdot s\left[y^{(i)}; a_s, b_s\right]\right]\end{aligned} \tag{6.11}$$

Thus, (6.10) and (6.11) show that the first and second derivative of the sigmoid only depend on the sigmoid value in (6.8) itself. This makes the evaluation very efficient in the context of numerical optimization as only the anyway necessary sigmoid function evaluation is required. It should be noted that (6.10) and (6.11) can be connected directly with (6.3) using the chain rule to calculate the derivative with respect to the expansion coefficients, which is required in the NLP optimization.

The concept from (6.8)–(6.11) can then be extended to multiple dimensions, i.e., a multidimensional failure domain depending on multiple output variables by e.g., multiplication of scalar sigmoids (leaving out the random sample index for the sake of readability):

$$s\left[\mathbf{y}; \mathbf{a_s}, \mathbf{b_s}\right] = \prod_{j=1}^{n} s\left[y_j; a_{s,j}, b_{s,j}\right] \tag{6.12}$$

This is possible because the sigmoid mainly returns function values of one and zero, except for the small transition area (see Figure 6.1). Therefore, a combination of multiple sigmoids is possible by multiplication, which also conserves the desired smoothness properties in (6.7).

In (6.12), the different sigmoids, denoted by the index j, can e.g., be used to design the box constraint in Figure 6.1. Generally, there might be n sigmoids required to recreate the shape of the failure/non-failure domain. It should be noted that the multiplication in (6.12) can be used to create box constraints defined by hyper-rectangles. By combining different hyper-rectangles (e.g., by weighting and rotating) also more complex shapes can be created. This is omitted here for the sake of brevity and the fact that box constraints, as defined by hyper-rectangles, are the main constraint type in OC.

Then, the derivative of (6.12) with respect to each output variable, required in the NLP, can be calculated as follows:

$$\frac{\mathrm{d}s\left[\mathbf{y};\mathbf{a_s},\mathbf{b_s}\right]}{\mathrm{d}y_k} = \frac{\mathrm{d}s\left[y_k;a_{s,k},b_{s,k}\right]}{\mathrm{d}y_k} \cdot \prod_{\substack{j=1\\j\neq k}}^{n} s\left[y_j;a_{s,j},b_{s,j}\right], \quad \forall k \tag{6.13}$$

This is a combination of the direct derivative with respect to the currently considered output k as well as the product of the remaining original sigmoid values and a direct consequence of the product rule.

The Hessian is then consequently given by applying the chain rule once more:

$$\frac{\mathrm{d}s\left[\mathbf{y};\mathbf{a_s},\mathbf{b_s}\right]^2}{\mathrm{d}y_k\mathrm{d}y_l} = \frac{\mathrm{d}s\left[y_k;a_{s,k},b_{s,k}\right]}{\mathrm{d}y_k} \cdot \frac{\mathrm{d}s\left[y_l;a_{s,l},b_{s,l}\right]}{\mathrm{d}y_l} \cdot \prod_{\substack{j=1\\j\neq k\\j\neq l}}^{n} s\left[y_j;a_{s,j},b_{s,j}\right], \quad \forall k,l \tag{6.14}$$

Thus, also in the multi-dimensional case, here for hyper-rectangular domains, the Jacobian and Hessian can be provided to the NLP (see (6.13) and (6.14)). Again, this shows the viability of the proposed approach using sigmoid functions to approximate the probabilistic constraint in CC-OC.

6.1.3 Homotopy Strategy for Smooth Approximator

As given in (6.7), a smooth approximator is required for the NEWTON-type NLP scheme (especially for the calculation of the Jacobian and Hessian). This can be achieved by using sigmoids as defined in (6.8), which are shaped using the two parameters a_s (scaling) and b_s (offset). Specifically, the scaling factor must be large enough to approximate the exact CC bound accurately (see Figure 6.1).

To enable convergence of the NLP algorithm, these parameters must be adapted suitably: Here, a smooth approximation (based on the problem characteristics) is normally used in the beginning, e.g., $|a_s| = 1$, and afterward continuously increased by a homotopy factor $a_{s,\mathrm{hom}} > 1$. Then, the value of b_s can be calculated appropriately. Therefore, the value is split into two parts:

$$b_{s,\mathrm{lb/ub}} = y_{\mathrm{lb/ub}} - c_{s,\mathrm{lb/ub}} \tag{6.15}$$

Here, the known bound value $y_{\mathrm{lb/ub}}$ (i.e., the lower or the upper bound of the CC domain) is used and a shift c_s is introduced on that bound. This is required because only using the bound value results in $s\left[y_{\mathrm{lb/ub}}; a_{s,\mathrm{lb/ub}}, b_{s,\mathrm{lb/ub}}\right] = 0.5$ by the definition of the sigmoid. This is clearly not desired, as the bound value should be 1 (or at least sufficiently close to 1).

Inserting (6.15) in (6.8), an equation that can be solved for the bound shift value is resulting:

$$s\left[y_{\mathrm{lb/ub}}; a_{s,\mathrm{lb/ub}}, b_{s,\mathrm{lb/ub}}\right] = \frac{1}{\exp\left[-a_{s,\mathrm{lb/ub}} \cdot \left(y - y_{\mathrm{lb/ub}} + c_{s,\mathrm{lb/ub}}\right)\right] + 1} \equiv \mathrm{BL}_s \tag{6.16}$$

Here, the value BL_s ("bound level") is introduced that defines the value the sigmoid should have at the boundary value of the constraint (i.e., the failure domain bound). This is required due to the fact that the sigmoid, as mentioned, never exactly reaches the value of 1. It should be noted that this bound level should be much larger than the desired probability the CC must fulfill (from the author's experience at least three orders of magnitude, i.e., if $\xi = 0.99$, then $\mathrm{BL}_s = 0.99999$, should be chosen). Then, the value of b_s can be calculated by solving (6.16) using (6.15). By this, the sigmoid is fully defined and the ROCP can be solved.

Generally, the described procedure in the previous paragraph, also called "homotopy" and as described in Algorithm 6.1, requires the repetitive solution of multiple ROCPs (Step 5 in Algorithm 6.1) until a sigmoid shape is reached that approximates the exact CC bound by the indicator function in (6.5) reasonably accurate. Take into account that this procedure introduces computational overhead. Still, it is reasonable to use such a procedure, because it might enable convergence in the first place. Here, the smooth approximation (by a small scaling) in the beginning normally ensures that a good initial optimal trajectory for the CC-OCP is found. When gradually increasing the sharpness of the sigmoid, the optimization can always be started with the previous optimal solution (Step 6 in Algorithm 6.1). By this procedure, the optimizer is generally converging very fast, as the previous solution should already fulfill most of the constraints and only the CC might require minor adaptations (due to the sharper bound). Thus, the homotopy procedure for the CC domain is generally reasonable to apply, especially if no suitable initial guess is available.

An important aspect of the sigmoid homotopy procedure is the determination of a suitable final/desired value for the sigmoid scaling $a_{s,\mathrm{desired}}$ (see Step 0 Algorithm 6.1). This factor is crucial as also already seen in Figure 6.1: A too small value yields a too smooth approximation of the CC and thus, failure events might still have a large sigmoid fulfillment value although they should be zero in reality. On the other hand, a too large value might introduce significant computational overhead, as the problem is solved with a higher accuracy than required. Clearly, the terms "small" and "large" must be seen in this context with respect to the magnitude of the considered quantity in the CC. Here,

Algorithm 6.1 Implemented homotopy strategy for sigmoid scaling and offset parameter in chance-constrained optimal control framework (after [102]).

Require:

Define the homotopy factor $a_{s,\text{hom}} > 1$.

Define the desired final sigmoid scaling $a_{s,\text{desired}}$.

Define the initial sigmoid scalings $a_{s,\text{init}}$.

Define the bound level BL_s.

Define the bound value of the sigmoid $y_{\text{lb/ub}}$ (i.e., the bound value of the CC).

1: Initialize the sigmoid scalings $a_{s,\text{lb/ub}} = a_{s,\text{init}}$.

2: **while** $a_{s,\text{lb/ub}} < a_{s,\text{desired}}$ **do**

3: Calculate the sigmoid values to use in the optimization:
Sigmoid bound shift: $c_{s,\text{lb/ub}} = -\frac{\ln\left(\frac{1}{\text{BL}_s}-1\right)}{a_{s,\text{lb/ub}}}$
Sigmoid offset: $b_{s,\text{lb/ub}} = y_{\text{lb/ub}} - c_{s,\text{lb/ub}}$

4: Assign the sigmoid values to the CC-OCP.

5: Solve the CC-OCP in (6.1):
Use initial guess from last homotopy step (if available).

6: Increase $a_{s,\text{lb/ub}}$ by homotopy factor: $a_{s,\text{lb/ub}} = a_{s,\text{hom}} \cdot a_{s,\text{lb/ub}}$.

7: Save the optimal trajectory as an initial guess for the next homotopy step.

8: **end while**

9: **return** Robust optimal trajectory with desired CC approximation accuracy.

a straightforward way to check if the homotopy parameter is already sufficient is given by evaluating the gPC expansion response, after solving the CC-OCP with the current homotopy parameter (Step 5 in Algorithm 6.1) and checking the CC bounds by the indicator function approach in (6.4) directly (i.e., without the sigmoid approximation error). If the calculated probability is fulfilling the desired probability level, the sigmoid scaling is sufficient to create feasible trajectories for the desired CC-OCP with the indicator function. If not, the scaling factor should be increased. Take into account that this procedure does not significantly increase the computational effort as the sampling of the gPC expansion outside the NLP iteration is, as already mentioned, very efficient (see (6.2)).

Another strategy in the context of choosing an appropriate final scaling parameter for the sigmoid, is by looking at the sigmoid offset parameter calculated in the homotopy step (Step 3) of Algorithm 6.1. As noted in Subsection 6.1.3, and also seen in Figure 6.1, this offset defines the point at which the sigmoid reaches the value 0.5. As the bound value of the CC is known, the difference between it and the offset gives an indicator on how smooth or sharp the sigmoid is. Especially, when looking at the relative error $(|c_s - y_{\text{lb/ub}}|)/(y_{\text{lb/ub}})$, it has proven to be sufficient that this ratio is at least smaller than 10^{-3}. More generally, it can also be aimed that the ratio is smaller than the feasibility tolerance of the ROCP.

Then, the sigmoid is sharp enough such that even small violations of the CC bound yield to a notable decrease in the CC fulfillment probability, which is detected by the optimizer. Thus, solutions outside the CC domain become unlikely and the scaling parameter can be considered sufficiently large.

Overall, both mentioned approaches have proven to be efficient in the application cases of this work and can also be combined to improve the confidence in the ROCP result. Nonetheless, there might still be more suited strategies/methodologies for specific problem formulations. Thus, further investigations can be done in this direction.

6.2 Subset Simulation based Chance-constrained Optimal Control

Although the MCA-like CC approximation based on the gPC expansion in Section 6.1 can already be applied to solve CC-OCPs, it has limitations with regard to calculating rare-event CC probabilities, i.e., probabilities of events that occur with a very low frequency (e.g., "one in one million"). Thus, this section introduces the extension of the sampling method presented in Section 6.1 to cover rare-event CC probabilities, which is Contribution 5 of this thesis. Therefore, the method of subset simulation (SubSim) (Subsection 2.2.4; a Markov chain Monte Carlo (MCMC) algorithm), which is specifically tailored to calculate rare-event probabilities, is incorporated within the CC-OC framework.

Summarizing, the necessity for introducing SubSim comes with the fact that the approximation of the probability by (6.4) can normally only be used if rather loose bounds on the probability (e.g., domain of $\xi = 99\%$) are considered. For tighter bounds (i.e., rare-events; e.g., $\xi = 99.9999\%$), as often used in e.g., reliability of safety engineering, better suited algorithms to calculate and sample the probability are required. Indeed, a reliable estimation of the probability of rare-events normally requires a very large number of samples.

This approach has also already been used by the author in [102]. Building on this, the thesis offers extensions to the work in [102] by e.g., applying a PDF approximation of the failure probability domain. This PDF can then be used to improve the robustness of the calculated results as e.g., the CI of the CC can be optimized. It should be noted that this section also uses the approximations of the CC domain by sigmoids and thus, mitigates the same benefits and limitations within OC (Section 6.1).

Before giving the technical details of the developed algorithm, it is important to note that the feasibility tolerance of the ROCP (see (2.20)) must be adapted accordingly when solving a rare-event CC-OCP. This means that the feasibility tolerance must be chosen much smaller than the failure probability such that the desired high probability and thus, confidence in the ROC solution is achieved.

6.2.1 Combination of Subset Simulation and Optimal Control

As already introduced in Subsection 2.2.4, a classical approach to circumvent the issue of requiring a large number of samples to get a reliable estimate of the rare-event probability is the use of the SubSim method. This method is tailored to evaluate such rare-events [7–9, 81]. The basic SubSim algorithm (see Subsection 2.2.4) is adapted to the needs of the CC-OC framework in the following.

In this thesis, it is proposed to use the SubSim algorithm (Algorithm 2.3), which is based on a MMHA (Algorithm 2.2), in a homotopy step of the CC-OC algorithm. The basic algorithm that results from this connection in a homotopy step is as shown in Algorithm 6.2 (the steps in the algorithm are detailed within the next paragraphs). Take into account that this homotopy step does not necessarily increase the computational burden of the CC-OCP solution process, as it can be combined with the anyway required homotopy step for the sigmoid sharpening (Algorithm 6.1). The idea of the homotopy update is to calculate the evaluation samples, created during the MMHA in Algorithm 2.3 (Step 9), outside the NLP iteration (Step 19 in Algorithm 6.2). This removes the task to run the time-consuming and stochastic sample creation in each iteration of the NLP, i.e., while solving the ROCP. Additionally, the MMHA would require a stochastic Newton-type update of the NLP iteration because of the stochasticity of the sample creation. Thus, it is not desired to have the MMHA in a deterministic NLP scheme and rather create the samples independent of the NLP.

Summarizing, SubSim is considered in this thesis as an add-on to the CC-OCP that mainly provides the evaluation samples for the CC evaluation within the NLP iteration. The core idea here is the separation of the stochastic sample creation and the deterministic NLP. Thus, SubSim runs “outside” of the solution process for the CC-OCP. In the following, the calculation of the CC probability is introduced as well as the connection to the calculated SubSim probability is cleared. Specifically, the relationship character of SubSim and NLP is detailed.

Algorithm 6.2 Basic strategy of homotopy for the subset simulation algorithm within the chance-constrained optimal control framework (after [102]).

Require:

ROCP as e.g., in (6.1) with initial guess for decision variables $\hat{\mathbf{z}}$.
Define the desired bound level that the rare-event CC should be fulfilled to ("CC probability level"): $\boldsymbol{\xi}$.
Number of samples per level n_s.
Conditional probability p_0.
Critical threshold b.
Maximal MCMC level $n_{\text{SubSim,max}}$.
Maximal number of homotopy steps: $l_{\max}$.
Homotopy factor for CC probability: $0 < \boldsymbol{\xi}_{\text{hom}} < 1$

1: Calculate an optimal solution for a likely failure (e.g., $\boldsymbol{\xi} = 99\%$) using the MCA-type ROC (Section 6.1) and use it as initial guess.
2: Obtain the subset probability $\mathbb{P}_{\text{SubSim}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$ for the MCA-type ROC by applying Algorithm 2.3 (e.g., by simulation or sampling the gPC expansion).
3: Save the generated samples $\{\boldsymbol{\theta}_{jk}^{(m-1)} : j = 1, \ldots, n_c,\, k = 1, \ldots, n_{sc}\}$ used for the calculation of the previous system response set, i.e., the set that was used to calculate $\mathbb{P}_{\text{SubSim}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$ (Algorithm 2.3 Step 10). These are used as the constant samples to evaluate the CC in the NLP.
4: Save the number of SubSim levels n_{ss}, which is also used as a constant within the NLP CC evaluation.
5: Set counter: $l = 1$
6: Set the CC probability that the CC-OCP is currently solved by: $\boldsymbol{\xi}_{\text{OCP}} = \boldsymbol{\xi}$
7: **while** $\mathbb{P}_{\text{SubSim}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right] > \boldsymbol{\xi}$ & $\mathbb{P}_{\text{OCP}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right] > \boldsymbol{\xi}$ & $l \leq l_{max}$ **do**
8: Assign the current CC probability $\boldsymbol{\xi}_{\text{OCP}}$ to the NLP CC.
9: Assign the SubSim samples $\{\boldsymbol{\theta}_{jk}^{(n_{ss})} : j = 1, \ldots, n_c,\, k = 1, \ldots, n_{sc}\}$ and the number of SubSim levels n_{ss} from the last iteration to the evaluation routine of the CC within the OCP.
10: Solve the CC-OCP (see (6.1)) to get the updated probability $\mathbb{P}_{\text{OCP}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$.
11: **if** Optimization not successful **then**
12: Relax the CC-OCP: $\boldsymbol{\xi}_{\text{OCP}} = \boldsymbol{\xi}_{\text{hom}} \cdot \boldsymbol{\xi}_{\text{OCP}}$
13: **else**
14: **if** $\boldsymbol{\xi}_{\text{OCP}} \neq \boldsymbol{\xi}$ **then**
15: Increase the CC probability for the CC-OCP: $\boldsymbol{\xi}_{\text{OCP}} = \frac{\boldsymbol{\xi}_{\text{OCP}}}{\boldsymbol{\xi}_{\text{hom}}}$
16: **end if**
17: **end if**

Algorithm 6.2 Basic strategy of homotopy for the subset simulation algorithm within the chance-constrained optimal control framework (continued; after [102]).

18: Obtain the new subset probability $\mathbb{P}_{\text{SubSim}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$ by applying Algorithm 2.3 (e.g., by simulation or sampling the gPC expansion).

19: Save the generated samples $\{\boldsymbol{\theta}_{jk}^{(n_{ss})} : j = 1, \ldots, n_c,\, k = 1, \ldots, n_{sc}\}$ used for the calculation of the previous system response set, i.e., the set that was used to calculate $\mathbb{P}_{\text{SubSim}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$ (Algorithm 2.3 Step 10). These are used as the constant samples to evaluate the CC in the NLP.
Store the number of SubSim levels n_{ss}, which is also used as a constant within the NLP CC evaluation.

20: Increase counter: $l = l + 1$

21: **end while**

22: **return** Optimal decision variables $\mathbf{z}_{\text{opt}}$ and CC probabilities $\mathbb{P}_{\text{SubSim}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$ as well as $\mathbb{P}_{\text{OCP}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$.

6.2.2 Probability Calculation in Subset Simulation based Chance-constrained Optimal Control

First of all, it should be noted that there are two different implemented strategies to calculate the probability of the rare-event failure in the ROCP of (6.1) by SubSim (desired fulfillment probability: $\boldsymbol{\xi}$): Within the NLP iteration the sampling-based approach (Subsection 6.1.1 using Algorithm 6.2) using the sigmoid (Subsection 6.1.2) is applied. Here, the sigmoid is used to evaluate the failure probability by (2.43) or (2.47). Take into account that the fulfillment probability of this CC, $\boldsymbol{\xi}_{\text{OCP}}$, can be updated while solving the CC-OCP with SubSim (Steps 11–17 in Algorithm 6.2). This might be necessary if the initial guess does not allow the NLP iteration to converge and a relaxed NLP should therefore be solved initially. It is further reminded here that the formulation in (2.47) is generally preferred because it gives a continuous representation of the failure probability (due to the Beta PDF approximation), while it also provides the possibility to calculate the standard deviation, which can be used to robustify the solution by CIs and calculate the coefficient of variation (CoV). Thus, (2.47) is applied in the following derivations and the examples of this thesis.

For the probability calculation, e.g., in the ROCP, it is further important to note that the failure probability in both (2.43) and (2.47) is calculated for being inside the failure domain $\mathcal{F}$. As also already stated, it is generally preferred to calculate the probability to not be in the failure domain, due to the numeric scaling of the NLP, which has the simple relation $\mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right] = 1 - \mathbb{P}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \in \mathcal{F}\right]$. Thus, it is easy to change between the two descriptions. In Algorithm 6.2, the calculated probability at the optimal point of the NLP is further on denoted by $\mathbb{P}_{\text{OCP}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$.

Furthermore, the SubSim in the homotopy step (Step 18 in Algorithm 6.2) is based on a full simulation of the model (e.g., by a standard RUNGE-KUTTA scheme [16, p. 145f.] or a hybrid Matrix Laboratory® (MATLAB®) differential equation integrator[1]). Here, the robust control history from the previous optimization result is used for the simulation. The results can then be evaluated using the indicator function to have an independent formulation of the ROCP, which is based on the sigmoid. By this, it is secured that the calculated control history does also fulfill the rare-event CC probability without the errors introduced by e.g., using the gPC expansion for the sampling, the tolerances of the NLP algorithm (i.e., the optimality and feasibility tolerances), as well as the sigmoid approximation. Additionally, the errors occurring by taking the random samples from the previous SubSim run are also mitigated. The corresponding probability calculated by the SubSim in the homotopy is further on denoted by $\mathbb{P}_{\text{SubSim}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right]$ (see Step 18 of Algorithm 6.2).

It should be further noted that the random samples, created in Step 18 of Algorithm 6.2, as well as the corresponding number of SubSim levels n_{ss} for these samples, are directly used in the NLP algorithm to evaluate the CC and remain constant during the NLP solution process. As mentioned, this moves the stochasticity of the SubSim to the homotopy step.

As also already stated, the basic procedure of the SubSim within CC-OC, and especially the two step SubSim evaluation procedure, once within the NLP using the gPC expansion with constant level samples and the other time within the homotopy step using a simulation with the calculated robust control history, is illustrated in Algorithm 6.2 as well: Here, it is clear that the CC-OCP is only considered to be solved if both the CC within the NLP as well as the SubSim homotopy step fulfill the desired CC probability level $\mathbb{P}_{\text{des}}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) \notin \mathcal{F}\right] = \boldsymbol{\xi}$ (Step 7 in Algorithm 6.2).

It is important to note that the procedure of assigning constant samples as well as using the number of SubSim from the homotopy step (Steps 9 and 19 in Algorithm 6.2), makes it straightforward to calculate the Jacobian and Hessian of the failure probability estimate from the BETA PDF in (2.46) and (2.47). This is based on the fact that the derivative must only be taken with respect to the expansion coefficients. To illustrate this, the parameters c_1 and c_2, which are used for the calculation of the BETA PDF shape and rate parameter (see (2.46)), are looked at:

$$\begin{aligned} c_1 &= \underbrace{\frac{\left(p_0 \cdot n_s + 1\right)^{n_{ss}}}{\left(n_s + 2\right)^{n_{ss}}}}_{\text{const.}} \cdot \left[\sum_{i=1}^{n_s} \widetilde{\mathcal{I}}\left[\mathbf{y}_{n_{ss}}^{(i)}\left(\mathbf{z};\boldsymbol{\theta}_{n_{ss}}^{(i)}\right)\right] + 1\right] \\ c_2 &= \underbrace{\frac{\left(p_0 \cdot n_s + 2\right)^{n_{ss}}}{\left(n_s + 3\right)^{n_{ss}}}}_{\text{const.}} \cdot \left[\sum_{i=1}^{n_s} \widetilde{\mathcal{I}}\left[\mathbf{y}_{n_{ss}}^{(i)}\left(\mathbf{z};\boldsymbol{\theta}_{n_{ss}}^{(i)}\right)\right] + 2\right] \end{aligned} \tag{6.17}$$

[1] `https://mathworks.com/help/matlab/math/choose-an-ode-solver.html` (Retrieved April 23, 2019)

Here, the indicator function is approximated by the sigmoid representation within the NLP for smoothness reasons (it should be reminded that the sigmoid in (6.9) is defined for the non-failure domain and thus, the sigmoid must be subtracted from one to get the failure domain probability):

$$\begin{aligned} c_1 &= \underbrace{\frac{(p_0 \cdot n_s + 1)^{n_{ss}}}{(n_s + 2)^{n_{ss}}}}_{\text{const.}} \cdot \left[1 - \frac{\sum_{i=1}^{n_s} s\left[\mathbf{y}_{n_{ss}}^{(i)}; \mathbf{a_s}, \mathbf{b_s}\right]}{n_s} + 1 \right] \\ c_2 &= \underbrace{\frac{(p_0 \cdot n_s + 2)^{n_{ss}}}{(n_s + 3)^{n_{ss}}}}_{\text{const.}} \cdot \left[1 - \frac{\sum_{i=1}^{n_s} s\left[\mathbf{y}_{n_{ss}}^{(i)}; \mathbf{a_s}, \mathbf{b_s}\right]}{n_s} + 2 \right] \end{aligned} \tag{6.18}$$

Then, the derivative with respect to the expansion coefficients is given by the chain rule combining (6.3), (6.10), and (6.13) as follows:

$$\begin{aligned} \frac{\partial c_1}{\partial \hat{\mathbf{y}}^{(m)}} &= \underbrace{\frac{(p_0 \cdot n_s + 1)^{n_{ss}}}{(n_s + 2)^{n_{ss}}}}_{\text{const.}} \cdot \left[-\frac{1}{n_s} \sum_{i=1}^{n_s} \left(\underbrace{\frac{\partial s\left[\mathbf{y}_{n_{ss}}^{(i)}; \mathbf{a_s}, \mathbf{b_s}\right]}{\partial \mathbf{y}_{n_{ss}}^{(i)}}}_{(6.13)} \cdot \underbrace{\frac{\partial \mathbf{y}_{n_{ss}}^{(i)}}{\partial \hat{\mathbf{y}}^{(m)}}}_{(6.3)} \right) \right] \\ \frac{\partial c_2}{\partial \hat{\mathbf{y}}^{(m)}} &= \underbrace{\frac{(p_0 \cdot n_s + 2)^{n_{ss}}}{(n_s + 3)^{n_{ss}}}}_{\text{const.}} \cdot \left[-\frac{1}{n_s} \sum_{i=1}^{n_s} \left(\underbrace{\frac{\partial s\left[\mathbf{y}_{n_{ss}}^{(i)}; \mathbf{a_s}, \mathbf{b_s}\right]}{\partial \mathbf{y}_{n_{ss}}^{(i)}}}_{(6.13)} \cdot \underbrace{\frac{\partial \mathbf{y}_{n_{ss}}^{(i)}}{\partial \hat{\mathbf{y}}^{(m)}}}_{(6.3)} \right) \right] \end{aligned} \tag{6.19}$$

Thus, the necessary Jacobian of the probability for the NLP algorithm using (6.19) can be calculated.

Take into account that the Hessian can be derived in the same manner as (6.19) using (6.14). Additionally, the chain rule for the product of derivatives in (6.19) as well as the fact that the derivative of (6.3) with respect to the expansion coefficients is zero must be considered. Then, the Hessian is given as follows:

$$\begin{aligned} \frac{\partial^2 c_1}{\partial \hat{\mathbf{y}}^{(m)} \partial \hat{\mathbf{y}}^{(n)}} &= \underbrace{\frac{(p_0 \cdot n_s + 1)^{n_{ss}}}{(n_s + 2)^{n_{ss}}}}_{\text{const.}} \cdot \left[-\frac{1}{n_s} \sum_{i=1}^{n_s} \left(\underbrace{\frac{\partial^2 s\left[\mathbf{y}_{n_{ss}}^{(i)}; \mathbf{a_s}, \mathbf{b_s}\right]}{\partial \left(\mathbf{y}_{n_{ss}}^{(i)}\right)^2}}_{(6.14)} \cdot \underbrace{\frac{\partial \mathbf{y}_{n_{ss}}^{(i)}}{\partial \hat{\mathbf{y}}^{(n)}}}_{(6.3)} \cdot \underbrace{\frac{\partial \mathbf{y}_{n_{ss}}^{(i)}}{\partial \hat{\mathbf{y}}^{(m)}}}_{(6.3)} \right) \right] \\ \frac{\partial^2 c_2}{\partial \hat{\mathbf{y}}^{(m)} \partial \hat{\mathbf{y}}^{(n)}} &= \underbrace{\frac{(p_0 \cdot n_s + 2)^{n_{ss}}}{(n_s + 3)^{n_{ss}}}}_{\text{const.}} \cdot \left[-\frac{1}{n_s} \sum_{i=1}^{n_s} \left(\underbrace{\frac{\partial^2 s\left[\mathbf{y}_{n_{ss}}^{(i)}; \mathbf{a_s}, \mathbf{b_s}\right]}{\partial \left(\mathbf{y}_{n_{ss}}^{(i)}\right)^2}}_{(6.14)} \cdot \underbrace{\frac{\partial \mathbf{y}_{n_{ss}}^{(i)}}{\partial \hat{\mathbf{y}}^{(n)}}}_{(6.3)} \cdot \underbrace{\frac{\partial \mathbf{y}_{n_{ss}}^{(i)}}{\partial \hat{\mathbf{y}}^{(m)}}}_{(6.3)} \right) \right] \end{aligned} \tag{6.20}$$

Then, (6.19) and (6.20) provide the Jacobian and Hessian for the coefficients used to calculate the shape and rate parameter of the BETA PDF respectively, which is used to estimate the failure probability. Thus, the Jacobian and Hessian calculation must be extended for the NLP to the e.g., mean and standard deviation value in (2.47) applying (2.46)

(using the coefficients in (6.19) and (6.20)). These results can directly be calculated using e.g., symbolic derivations and the chain rule and are thus, left out here for the sake of compactness.

To conclude the SubSim-based CC-OC framework, the following two paragraphs discuss the convergence as well the choice of SubSim parameters for the developed algorithm:

Convergence Discussion of Proposed Algorithm

As a general statement, it can be said that the proposed SubSim homotopy strategy is viable for the use within NLPs as especially the Jacobian and Hessian can be provided efficiently. On the other hand, it should be noted that by using the SubSim samples calculated from the previous optimal solution within the new optimization, a bias might be introduced as the samples drawn from the Markov chain are based on the optimal result created by the previous NLP solution. Generally, the samples would have to adapted in each iteration of the NLP, as the system response changes and therefore, the MMHA might create different results. As this is not done within the NLP, but within the homotopy step after a new optimal solution has been calculated, the CC and its rare-event probability is solved using "non-ideal" samples compared to the ones that would be calculated within the SubSim (still, the used samples could be a possible outcome set of the random MMHA within the SubSim nonetheless). This issue is coped with in this thesis by checking the fulfillment of the CC probability both in the CC-OCP as well as after the CC-OCP is solved by the SubSim in the homotopy step, i.e., with the new response surface. Thus, the CC-OCP is only considered to be solved if both results show that the CC is fulfilled to the desired probability level. As stated, within this thesis, the CC-OCP converges fine, but further studies can explore the effects and influences of this bias and how to reduce it (e.g., by introducing importance sampling techniques [81, p. 23ff.]). Additionally, sensitivity analysis techniques could be used to update the samples based on the difference to the previous optimal solution, i.e., the change in the response surface (this would require "inverse" sensitivities, i.e., parameter updates based on the decision variable changes rather than decision variable updates based on the sensitive parameters changes). Still, the application cases of this thesis show that the proposed algorithm converges well even omitting these issues. As mentioned, this is due to the two-step procedure and independent SubSim evaluation in the NLP and the homotopy to assure a feasible solution.

Choice of Subset Simulation Proposal Distribution

A further important aspect of the SubSim, and especially the used MMHA within it, is the choice of the proposal PDF around the current samples in Step 0 of Algorithm 2.2. Here, generally the choice of a proposal PDF is difficult, because the required "spread" is largely depending on the original PDF of the samples and the weighting of acceptance rate and spatial dependence. Using the gPC method, this choice of a proposal PDF becomes

more straightforward: This is due to the fact that the samples are created in a way that they can be evaluated in the orthogonal polynomial domain. This consequently means that the samples are always defined such that they are in their "standard" domains of gPC expansion as defined in Table 2.2. Thus, it is not necessary to adapt the proposal PDF choice for different parameter settings, but instead it is possible to define a standard relation between the parameter's PDF and the proposal PDF. As stated in [9, p. 123], a safe strategy to choose the standard deviation is the use of the standard deviation of the target distribution. Thus, for an uncertain parameter with a GAUSSIAN PDF, the proposal PDF can also be chosen as the standard GAUSSIAN PDF. By this, a good exploration of the failure region can be assured. Furthermore, for a UNIFORM PDF, the exploration can be done with the standard UNIFORM spread around the current sample and be clipped at the standard domain bounds accordingly. This has proven to be efficient as well. Therefore, the gPC algorithm actually has some benefits for SubSim, due to its standard definition in terms of the PDFs.

Concluding, both introduced CC-OC framework, with MCA-based sampling for "frequent" events (Section 6.1) as well as the SubSim-based sampling for rare-events, are tested in Chapter 9 to show their applicability in the OC context.

Chapter 7

Design of Robust Gains for Control Loop

This chapter covers the first application example of the developed robust open-loop direct optimal control (ROC) frameworks of this thesis. Within this chapter, the bi-level framework, introduced in Chapter 3, is applied to calculate robust controller gains and thus, results for Contribution 2 are shown. An adaptive control loop is considered to show the applicability of the methodology to calculate robust, optimal adaptation gains.

Generally, the calculation is done by at first applying the framework in Section 3.1, i.e., with the differential evolution algorithm (DEA) in the upper level, to restrict the domain of the (globally) optimal gains. Afterward, the framework from Section 3.2 is used, in the domain around the optimal gains from the DEA result, to find the numerically exact optimum. This procedure is applied to be more certain that the calculated optimal gains are globally optimal.

To show the procedure and the results, the chapter is organized as follows: In Section 7.1, the dynamic model for the control loop is introduced. Then, Section 7.2 depicts the application of the bi-level framework to the calculation of the adaptation gain in an adaptive control loop with Section 7.3 illustrating the results. Here, especially probabilistic constraints, i.e., chance constraints (CCs), are applied to achieve a robustification.

7.1 Dynamic Model for Gain Design

In this example, a short period approximation of an F-16 is used to show the robust gain design based on the bi-level generalized polynomial chaos (gPC) method (Chapter 3; after [122, p. 259] and [59, p. 29f.]):

$$\underbrace{\begin{bmatrix}\dot{\alpha_K}\\ \dot{q_K}\end{bmatrix}}_{\dot{\mathbf{x}}\in\mathbb{R}^{2\times1}} = \underbrace{\begin{bmatrix}Z_{\alpha_K} & 1+Z_{q_K}\\ \theta_{\alpha_K}\cdot M_{\alpha_K} & \theta_{q_K}\cdot M_{q_K}\end{bmatrix}}_{\mathbf{A}\in\mathbb{R}^{2\times2}}\cdot\underbrace{\begin{bmatrix}\alpha_K\\ q_K\end{bmatrix}}_{\mathbf{x}\in\mathbb{R}^{2\times1}} + \underbrace{\begin{bmatrix}0\\ \theta_\eta\cdot M_\eta\end{bmatrix}}_{\mathbf{B}\in\mathbb{R}^{2\times1}}\underbrace{\eta}_{\mathbf{u}\in\mathbb{R}^{1\times1}} \tag{7.1}$$

Table 7.1: *Description of constants in the state and control matrix of the F-16 short period approximation dynamic model.*

Meaning	Symbol	Value
Lift force due to angle of attack	Z_{α_K}	-1.8406
Lift force due to pitch rate	Z_{q_K}	-0.0908
Pitch moment due to angle of attack	M_{α_K}	-30.7874
Pitch moment due to pitch rate	M_{q_K}	-3.7534
Pitch moment due to elevator	M_η	-12.7528

Here, the states are given by the angle of attack α_K and the pitch rate q_K. The elevator deflection η is the control. The multiplicative uncertainties for the angle of attack, pitch rate, and elevator deflection influence on the pitch acceleration $\dot{q_K}$ are denoted by θ_{α_K}, θ_{q_K}, and θ_η respectively. The constant entries in the state and control matrix with their meaning are introduced in Table 7.1. These values were calculated by a linearization of the nonlinear rigid-body F-16 model [122, p. 714ff.], using a steady-state horizontal flight condition with a velocity of $135\frac{\mathrm{m}}{\mathrm{s}}$. This model is used in the following for the gain design.

7.2 Gain Design for Adaptive Control Loop

In this section, the calculation of the adaptation/learning gains for an adaptive controller structure is looked at. Generally, adaptive controllers have a different philosophy compared to standard controllers: Instead of using a fixed control gain matrix, they are continuously adapting their control gain matrix to recover a desired system performance. This system performance is often defined by a reference model and the procedure is known as model reference adaptive control (MRAC). A structural breakdown of the MRAC scheme, used in this thesis, is also given in Figure 7.1. A detailed description of all parts in this figure is given in the following paragraphs.

Generally, the goal for the gain calculation of the adaptive controller structure is not the calculation of the control gain matrix but the adaptation gain matrix. These adaptation gains must allow a reasonable fast adaptation of the system, while securing a "good" transient behavior. A transient behavior is considered to be "good", if oscillations and overshoots of the system are minimal.

At first, the state predictor/reference model is introduced: This reference model uses the plant model in (7.1) for the nominal case (i.e., no uncertainties). For this nominal model, a desired dynamic behavior by means of pole placement is specified [122, p. 381ff.]. The general idea of this procedure is to use a control gain matrix such that the closed-loop system (i.e., when inserting the control law in the state dynamics) fulfills predefined dynamics defined by the eigenvalues of the state dynamic matrix. Generally, the following equation is looked at in the context of pole placement:

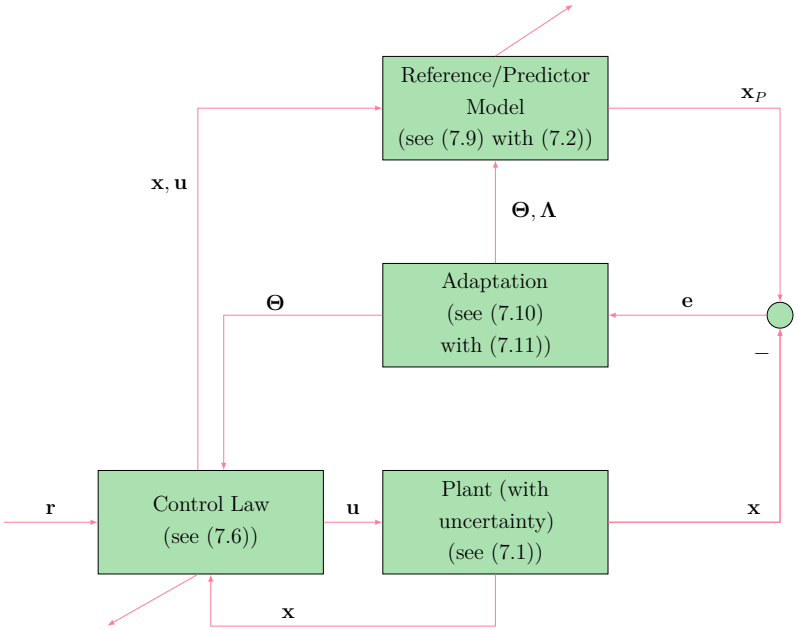

Figure 7.1: *Structure of implemented model reference adaptive control loop for robust adaptation/learning rate design.*

$$\mathbf{A}_P = \mathbf{A} - \mathbf{B}\mathbf{K} \tag{7.2}$$

For (7.2), the eigenvalues λ can be calculated as follows [5, p. 594ff.]:

$$\det\left(\mathbf{A}_P - \lambda\mathbf{I}\right) \equiv 0 \tag{7.3}$$

Take into account that the operator $\det(\cdot)$ in (7.3) is the determinant operator [5, p. 547f.] and $\mathbf{I}$ is the identity matrix of appropriate size. Equation (7.3) results in a polynomial of the order of the number of states in the eigenvalues. By assigning eigenvalues to the system, the entries to the, yet unknown, control gain matrix $\mathbf{K}$ can be calculated. These values are then chosen such that the closed-loop system fulfills desired dynamics.

For the model of this thesis, the eigenvalues are assigned as follows:

$$\lambda_{1,2} = -3 \pm 3\mathtt{i} \tag{7.4}$$

Here, $\mathtt{i}$ is the imaginary unit [5, p. 124f.] with the property $\mathtt{i}^2 = -1$ and thus, a stable oscillatory pole is assigned to the system, which yields the control gain matrix as follows:

$$\mathbf{K} \approx \begin{bmatrix} 1.5220 & -0.0318 \end{bmatrix} \tag{7.5}$$

Then, (7.2) provides the state dynamics of the reference model for the MRAC.

After defining the reference model state dynamic matrix, the control law for the adaptive controller is formulated as follows:

$$\mathbf{u} = \boldsymbol{\Lambda}^{-1}\left(\mathbf{K}_\mathbf{r}\mathbf{r} - \boldsymbol{\Theta}\mathbf{x}\right) \tag{7.6}$$

Here, $\mathbf{\Lambda}$ is the matching matrix for the control uncertainty ("control matching matrix"), while $\mathbf{\Theta}$ is the matching matrix for the state uncertainties ("state matching matrix"). Both of these matrices are adapted in the MRAC scheme to cancel out uncertainties. The reference command, i.e., the desired trajectory, is defined by $\mathbf{r}$ and $\mathbf{K_r}$ is the corresponding control gain matrix of the reference command to achieve a desired closed-loop performance (e.g., steady-state accuracy). It should be noted that $\mathbf{K_r}$ is designed to be the inverse of the low-frequency gain for the first state of the closed-loop dynamic model (this assures steady-state accuracy) as follows [84, p. 167ff.]:

$$\mathbf{K_r} = \left[-\begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{A}_P^{-1} \mathbf{B} \right]^{-1} \approx -1.5524 \tag{7.7}$$

The goal of the adaptive control loop is to control the error between the reference model and the real plant behavior to zero. This error is defined as follows:

$$\mathbf{e} = \mathbf{x}_P - \mathbf{x} \tag{7.8}$$

Here, the state vector $\mathbf{x}$ includes the uncertainties from the real plant model in (7.1), while $\mathbf{x}_P$ is the state vector obtained from the reference/predicted plant, defined as follows:

$$\dot{\mathbf{x}}_P = \begin{bmatrix} \dot{\alpha}_{K,P} & \dot{q}_{K,P} \end{bmatrix}^\mathsf{T} = \mathbf{A}_P \cdot \mathbf{x}_P + \mathbf{B}\mathbf{\Lambda}\mathbf{u} + \mathbf{B}\mathbf{\Theta}\mathbf{x} \tag{7.9}$$

It should be noted that inserting (7.6) in (7.9) yields the desired plant dynamics with reference command tracking.

In addition to the plant and the predictor dynamics model, an adaptive controller additionally comprises the dynamic equations for the matching conditions. These are dynamically updated by the error between prediction and plant and thus, try to reduce this error. The dynamic equations are given as follows [63, ch. 3c, p. 24]:

$$\begin{aligned} \dot{\mathbf{\Theta}} &= -\mathbf{\Gamma}_{\mathbf{\Theta}} \mathbf{x} \mathbf{e}^\mathsf{T} \mathbf{P}_L \mathbf{B} \\ \dot{\mathbf{\Lambda}} &= -\mathbf{\Gamma}_{\mathbf{\Lambda}} \mathbf{u} \mathbf{e}^\mathsf{T} \mathbf{P}_L \mathbf{B} \end{aligned} \tag{7.10}$$

Thus, the matching condition matrices are updated based on the error in (7.8) until it approaches zero. Here, the adaptation gain matrices $\mathbf{\Gamma}_{\mathbf{\Lambda}}$ and $\mathbf{\Gamma}_{\mathbf{\Theta}}$ for the control and state matching matrix respectively play an important role as they define the adaptation "speed". Generally, a fast adaptation (i.e., a large gain is desired), while there should not be any excessive oscillation (i.e., a smaller gain is required). This trade-off is approached within this thesis using the bi-level ROC approach. It should be noted that $\mathbf{\Gamma}_{\mathbf{\Lambda}}$ and $\mathbf{\Gamma}_{\mathbf{\Theta}}$ are positive definite and, generally, diagonal matrices (Table 7.2). Further take into account that (7.10) is derived in such a way that the LYAPUNOV function of the closed-loop remains stable [63, ch. 3c, p. 24].

In (7.10), the matrix $\mathbf{P}_L$ refers to the LYAPUNOV matrix that is the solution of the following LYAPUNOV equation [85, p. 300]:

Table 7.2: *Control and feed-forward gain values used for the adaptive closed-loop F-16 model optimization.*

Meaning	Symbol	(Rounded) Value
Lyapunov scaling matrix	$\mathbf{Q}_L$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Lyapunov matrix	$\mathbf{P}_L$	$\begin{bmatrix} 0.7627 & -0.0794 \\ -0.0794 & 0.1028 \end{bmatrix}$
Reference adaptation gain matrix	$\mathbf{\Gamma_\Theta}$	$\begin{bmatrix} \mathbf{\Gamma}_{\alpha_K} & 0 \\ 0 & \mathbf{\Gamma}_{q_K} \end{bmatrix} = \begin{bmatrix} 20 & 0 \\ 0 & 1 \end{bmatrix}$
Control gain matrix	$\mathbf{\Gamma_\Lambda}$	1
Feedforward gain	$\mathbf{K_r}$	-1.5524
Feedback gain matrix	$\mathbf{K}$	$\begin{bmatrix} 1.5220 & -0.0318 \end{bmatrix}$

$$\mathbf{A}_P^\top \mathbf{P}_L + \mathbf{P}_L \mathbf{A}_P = -\mathbf{Q}_L \tag{7.11}$$

Here, $\mathbf{Q}_L$ is a positive definite scaling matrix.

In Table 7.2 the numeric values of the matrices used in the bi-level open-loop direct optimal control problem (OCP) are introduced. The diagonal entries of the adaptation gain matrix for the states $\mathbf{\Gamma_\Theta}$ are the optimization parameters for the bi-level algorithms and thus, Table 7.2 shows the reference values. All other matrices and values in Table 7.2 are constants in the bi-level procedure and chosen such that a reasonable closed-loop behavior of the adaptive control loop is achieved.

7.3 Gain Design Results for Adaptive Control Loop

This section covers the results of the bi-level ROC gain design for the adaptive controller. At first, Subsection 7.3.1 introduces the problem definition of the bi-level ROC framework applied to the adaptive control loop. Then, Subsection 7.3.2 introduces the minimization of the mean and standard deviation of the control command area, while Subsection 7.3.3 introduces CCs in the previous bi-level ROC to robustify the results of the gain design.

7.3.1 Problem Definition

Generally, the adaptive control loop is subject to the following uncertainties in the state dynamics (see (7.1)), defined by their probability density functions (PDFs) as follows:

$$\begin{aligned} \theta_{\alpha_K} &\sim \mathcal{U}\,(a = 0.5, b = 1.5) \\ \theta_{q_K} &\sim \mathcal{U}\,(a = 0.5, b = 1.5) \\ \theta_\eta &\sim \mathcal{U}\,(a = 0.5, b = 1.5) \end{aligned} \tag{7.12}$$

This means that the mean values, defined in Table 7.1, can vary up to $\pm 50\,\%$. Take into account that this is a quite large uncertainty, but it can illustrate the capabilities of the ROC algorithm very well. A UNIFORM PDF is used as it can be assumed to be a worst-case PDF.

The reference command is defined to be a step as follows:

$$\mathbf{r} = \begin{cases} 0, & \text{for } 0\text{s} \leq t < 1\text{s} \\ 10\,^\circ \cdot \frac{\pi}{180\,^\circ}, & \text{for } 1\text{s} \leq t \leq 10\text{s} \end{cases} \tag{7.13}$$

It should be noted that a step excitation is used in this thesis due to the fact that it is a very common test excitation in controller design.

Take into account that the step height in (7.13) is deliberately large to show the potential of the ROC approach, although in a real application the assumption of having the same linear system for such a step might be violated. Further take into account that the final time is set to 10s, which is a long enough time horizon for the system to be in a steady-state again.

Take into account that the bi-level robust open-loop direct optimal control problem (ROCP) is as follows, using the adaptive controller dynamic equations and the control law at each stochastic collocation (SC) node as follows:

$$\begin{aligned}
&\min_{\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}} \quad J_{\text{UL}} \\
&\text{s.t.} \quad \boldsymbol{\Gamma}_{\boldsymbol{\Theta},\text{lb}} \leq \boldsymbol{\Gamma}_{\boldsymbol{\Theta}} \leq \boldsymbol{\Gamma}_{\boldsymbol{\Theta},\text{ub}} \\
&\qquad \text{s.t.} \quad \begin{aligned}
&\mathbf{x}^{(j)} = \mathbf{x}_0^{(j)} + \int_{t=t_0}^{t_f} \dot{\mathbf{x}}^{(j)}\,\mathrm{d}t \\
&\mathbf{x}_P^{(j)} = \mathbf{x}_{0,P}^{(j)} + \int_{t=t_0}^{t_f} \dot{\mathbf{x}}_P^{(j)}\,\mathrm{d}t \\
&\boldsymbol{\Theta}^{(j)} = \boldsymbol{\Theta}_0^{(j)} + \int_{t=t_0}^{t_f} \dot{\boldsymbol{\Theta}}^{(j)}\,\mathrm{d}t \\
&\boldsymbol{\Lambda}^{(j)} = \boldsymbol{\Lambda}_0^{(j)} + \int_{t=t_0}^{t_f} \dot{\boldsymbol{\Lambda}}^{(j)}\,\mathrm{d}t \\
&\dot{\mathbf{x}}^{(j)} = \mathbf{A}^{(j)}\mathbf{x}^{(j)} + \mathbf{B}^{(j)}\mathbf{u}^{(j)} \\
&\dot{\mathbf{x}}_P^{(j)} = \mathbf{A}_P^{(j)}\mathbf{x}_P^{(j)} + \mathbf{B}\boldsymbol{\Lambda}^{(j)}\mathbf{u}^{(j)} + \mathbf{B}^{(j)}\boldsymbol{\Theta}^{(j)}\mathbf{x}^{(j)} \\
&\dot{\boldsymbol{\Theta}}^{(j)} = -\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}\mathbf{x}^{(j)}\left(\mathbf{e}^{(j)}\right)^{\mathsf{T}}\mathbf{P}_L\mathbf{B}^{(j)} \\
&\dot{\boldsymbol{\Lambda}}^{(j)} = -\boldsymbol{\Gamma}_{\boldsymbol{\Lambda}}\mathbf{u}^{(j)}\left(\mathbf{e}^{(j)}\right)^{\mathsf{T}}\mathbf{P}_L\mathbf{B}^{(j)} \\
&\mathbf{u}^{(j)} = \left(\boldsymbol{\Lambda}^{(j)}\right)^{-1}\left(\mathbf{K}_{\mathbf{r}}\mathbf{r} - \boldsymbol{\Theta}^{(j)}\mathbf{x}^{(j)}\right)
\end{aligned}
\end{aligned} \tag{7.14}$$

The initial conditions used for the simulations in (7.14) are as follows:

$$\begin{aligned}
\mathbf{x}_0^{(j)} &= \mathbf{x}^{(j)}\left(t = t_0\right) = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathsf{T}} \\
\mathbf{x}_{0,P}^{(j)} &= \mathbf{x}_P^{(j)}\left(t = t_0\right) = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathsf{T}} \\
\boldsymbol{\Theta}_0^{(j)} &= \boldsymbol{\Theta}^{(j)}\left(t = t_0\right) = \begin{bmatrix} 1.5220 & -0.0318 \end{bmatrix}^{\mathsf{T}} \\
\boldsymbol{\Lambda}_0^{(j)} &= \boldsymbol{\Lambda}^{(j)}\left(t = t_0\right) = 1
\end{aligned} \tag{7.15}$$

It should be noted that the initial values for the matching matrix of the states are the ideal matching condition values, which can be calculated for this simple system analytically [63, ch. 3c, p. 23] and thus, provide a very good initial guess for the simulation. In other applications, appropriate values must be defined by the user.

It is reminded here that the initial solution of the problem is calculated by a DEA and afterward, the sensitivity based approach is applied. Here, the bounds for the DEA are chosen as $\Gamma_{\boldsymbol{\Theta},\mathrm{lb,DE}} = 0.1$ and $\Gamma_{\boldsymbol{\Theta},\mathrm{ub,DE}} = 150$ for each diagonal entry of the gain matrix to cover the main range of normally used adaptation gain values. For the bounds of sensitivity-based optimization conducted after the DEA, the adaptation gains bounds are calculated based on the optimal gains from the DEA:

$$\Gamma_{\boldsymbol{\Theta},\mathrm{lb/ub,sen}} = \Gamma_{\boldsymbol{\Theta},\mathrm{DE,opt}} \pm 0.25 \cdot |\Gamma_{\boldsymbol{\Theta},\mathrm{DE,opt}}| \tag{7.16}$$

Furthermore, the following setup for the DEA is used (Algorithm 3.1):

- Population size: $n_{\mathrm{pop}} = 25$
- Backtrack steps: $n_{\mathrm{bt}} = 15$
- Maximum number of iterations: $k_{\max} = 250$
- Optimality tolerance: $\epsilon_{\mathrm{opt}} = 10^{-3}$
- Feasibility tolerance: $\epsilon_{\mathrm{feas}} = 10^{-3}$
- gPC polynomial order: $d_i = 3$
- SC expansion nodes (tensor grid): $Q = 27$

For the sensitivity/NEWTON-type optimization step, Interior Point Optimizer (IPOPT) as version 3.12.12 is used. Here, the optimality and feasibility tolerances are set to $\epsilon_{\mathrm{opt}} = \epsilon_{\mathrm{feas}} = 10^{-5}$ respectively and the linear solver `ma97` is used.

7.3.2 Minimization of Control Area Mean and Standard Deviation

Within this section, the minimization of the mean control area and its standard deviation is looked at. Specifically the mean control area is a standard cost for control loops as a small control area, and thus, control effort, is generally desired. Additionally, the spread, i.e., the standard deviation, in the control effort should not be too large, which provides a robustification. Thus, the upper level cost function in (7.14) is defined by:

$$J_{\mathrm{UL}} = \int_{t=0}^{10} \left(\mathbb{E}\left[\mathbf{u}^{\mathsf{T}}\right] \mathbb{E}\left[\mathbf{u}\right] \right) \mathrm{d}t + \int_{t=0}^{10} \left(\sigma\left[\mathbf{u}^{\mathsf{T}}\right] \sigma\left[\mathbf{u}\right] \right) \mathrm{d}t \tag{7.17}$$

Take into account that due to the nonlinearity in the adaptive control loop introduced in e.g., (7.9) and (7.10), the mean value of the control command $\mathbf{u}$ and the result when simulating the system in (7.1) for the nominal parameter uncertainty value (i.e., no uncertainty influence) is generally not the same. Thus, the calculation of a gPC expansion to even calculate the mean value is required in this application.

The solution of the bi-level OCP in (7.14) with the cost function in (7.17) yields the following optimal gain matrix:

$$\boldsymbol{\Gamma}_{\boldsymbol{\Theta},\text{sen,opt}} \approx \begin{bmatrix} 9.294 & 0 \\ 0 & 0.075 \end{bmatrix} \tag{7.18}$$

It is imminent that the calculated adaptation gains are smaller than the reference/initial gains defined in Table 7.2. This is mainly based on the fact that with a smaller adaptation gain, the adaptive controller is not trying to reduce the errors created by the uncertainty fast. This yields smaller required control effort, but also worse tracking. Take into account that, as only the statistics of the control effort are considered as the cost function, fast tracking is not a requirement for the controller.

To examine the optimal gains for the adaptive control loop, the time history of the plant angle of attack in Figure 7.2 is looked at. Here, the optimal gains are depicted by a solid, blue line, while the reference gains from Table 7.2 is depicted by a dashed red line. Additionally, a worst-case (i.e., the combination of uncertainties that yields the maximal angle of attack) is depicted for the reference gains in solid magenta and for the optimal gains in solid cyan. It can be seen that the convergence to the desired $10\,^\circ$ step takes around 6 s for both gains, which is rather slow but a consequence of the chosen cost function that only tries to minimize the control area. Here, the steady-state step value should be achieved with a small control effort which in turn yields a slow convergence. From an overall perspective, both gains behave similar with the smaller optimal gains lagging slightly behind in transient phases. This is also seen in the standard deviation, where the optimal gain convergences to zero at a later time instance and has generally a larger value. It should be noted here that the uncertainties and the slow adaptation yield standard deviations of more than $4\,^\circ$, which is generally an undesired behavior. This is also seen in the worst-case approximations, which show that an angle of attack of more than twice the step size is reached with the optimal gains for the worst-case parameter combination. Thus, further robustifications are required in this context.

A similar behavior is also seen for the control history in Figure 7.3: The mean control command is similar for both reference and optimal gains, while the standard deviation is smaller for the optimal gains. This consequently creates less adaptation to the uncertainty influences and worse tracking. It should be noted that the magnitude of the values is related to the magnitude of the uncertainties and the fact that the control command is not limited. Thus, it might not be fully feasible for real applications but merely shows the general applicability.

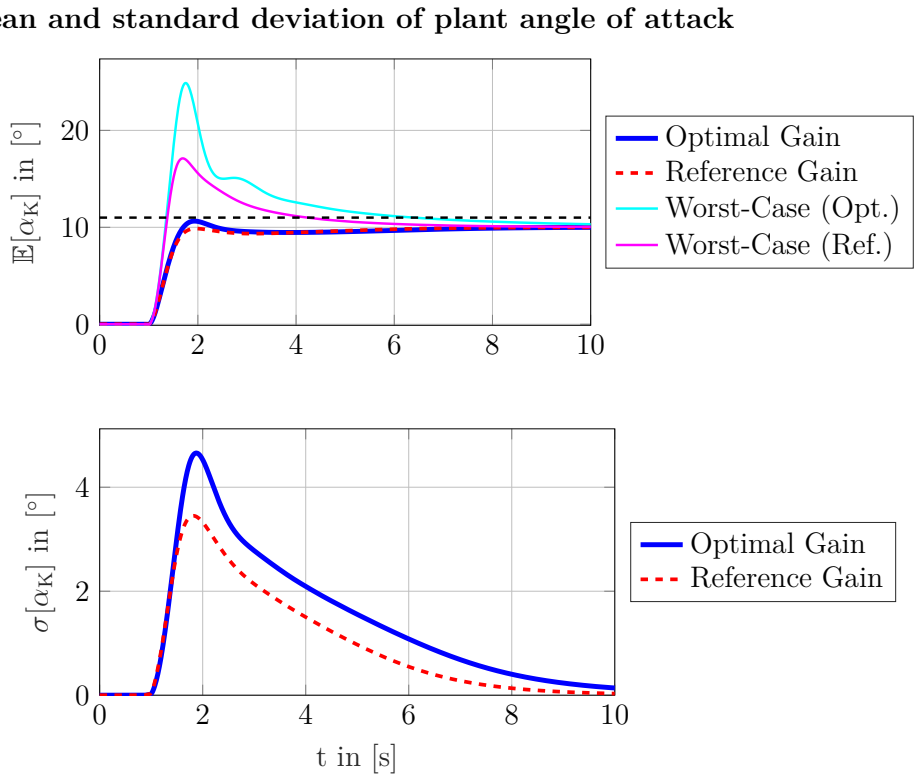

Figure 7.2: *Mean and standard deviation of plant angle of attack reaction to step input including worst-case approximations for reference and optimal adaptation gains.*

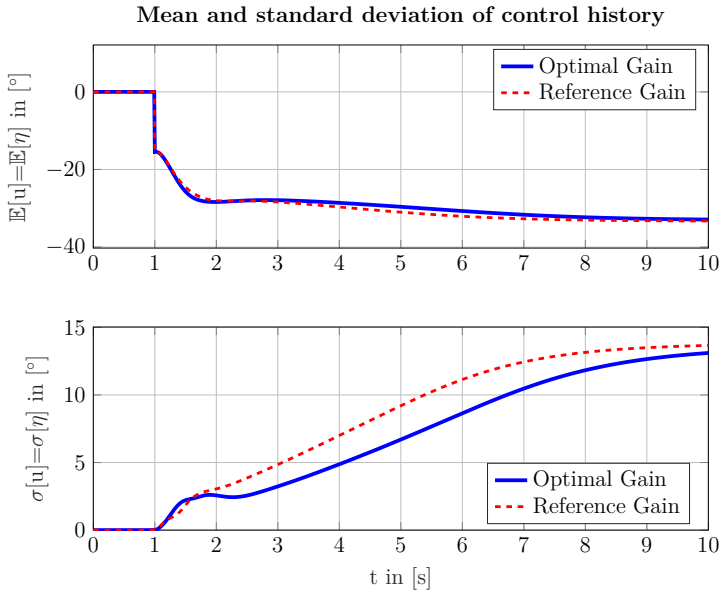

Figure 7.3: *Mean and standard deviation of control history during step input for reference and optimal adaptation gains.*

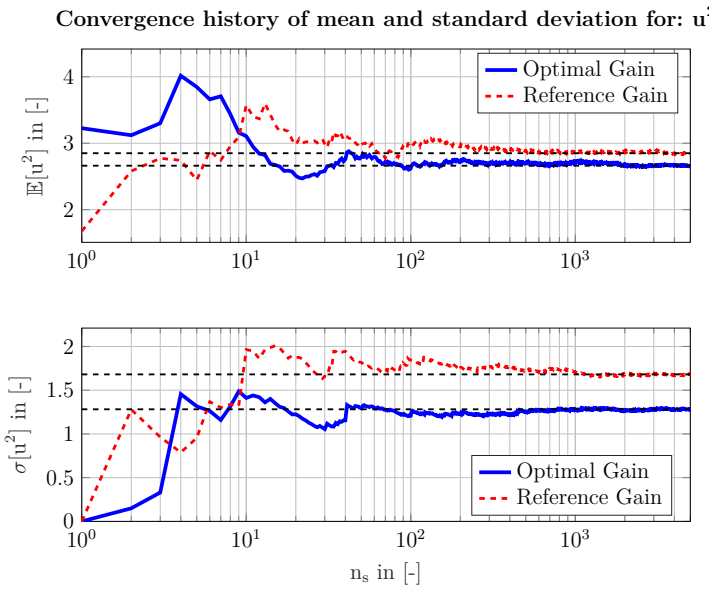

Figure 7.4: *Convergence history of mean value and standard deviation of control area for optimal and reference adaptation gains from minimization of control area mean and standard deviation in adaptive control loop.*

Furthermore, Figure 7.4 looks at the convergence history of the control area over time, i.e., the bi-level cost function. As already seen in Figures 7.2 and 7.3, the mean control area is very similar, with the optimal gain reaching a slightly smaller value. The standard deviation also shows that the optimal gain has a smaller variation in the control effort and thus, control area. Thus, the bi-level OCP achieves the desired cost goal.

Overall, the previous results show that the minimization of the control area statistics yields rather slow adaptation results, i.e., small adaptation gains, in order to keep the control area small (which is the goal of the cost function). Thus, the results cannot be considered safely robust (i.e., they are robust with respect to the uncertainty but lack a high safety, which is in this case e.g., not exceeding a maximal angle of attack). This is also imminent from Figure 7.5: Here, the probability of having an angle of attack smaller than 11 ° over the simulation time is depicted. It should be noted that this probability is calculated from sampling the simulation results using the optimal and reference gains with $n_s = 5000$ in a Monte Carlo analysis (MCA). From an application point of view, this bound can be reasoned by the fact that a too large angle of attack yields to a stall. Additionally, as the uncertainty domain is rather large (see (7.12)), a 10 % overshoot should generally be allowed for a step command. In Figure 7.5, it can be observed that the desired CC is actually violated by up to 40 % of the trajectories at approximately 2 s for both optimal as well as reference gain respectively. Additionally, the integrated area (see legend text box in Figure 7.5) of fulfillment is reduced by around 5.5 % using the robust optimal gains (see values in Figure 7.5).

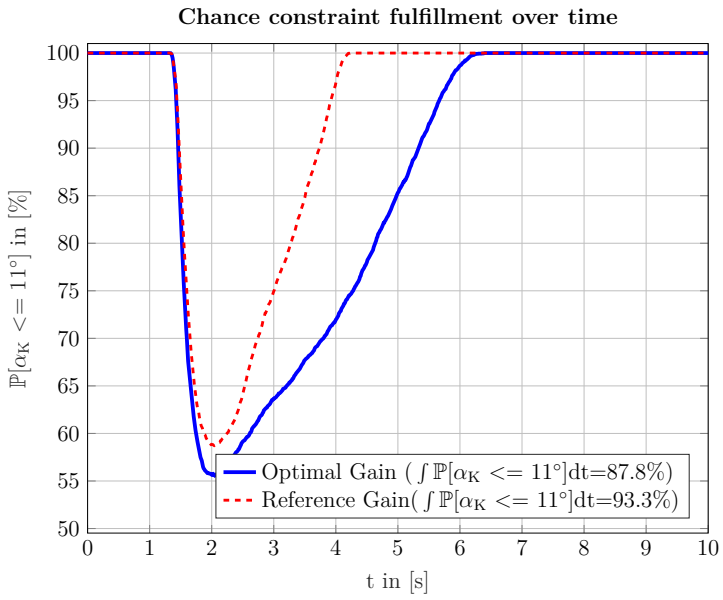

Figure 7.5: *Probability of not exceeding angle of attack limitation of eleven degrees over simulation time for optimal and reference adaptation gains.*

Thus, it must be secured in the following that the controller and resulting control commands create a safe and robust trajectory. This trajectory should only have a small (or at best no) probability of, in this case, violating the maximal allowable angle of attack. Concluding, the calculated adaptation gain cannot be considered as safely robust to the uncertainties with the standard cost function for control area statistics minimization and robustness modifications in the gain optimization process are required. This is examined in the next subsection by introducing CCs.

7.3.3 Minimization of Control Area Statistics with Chance Constraint Fulfillment

Although the previous section already gave an idea of the robustness improvements that are possible using the gPC bi-level approach, the improvements were not yet considered safely robust. This is dealt with in this section, which introduces CCs in the formulation. This is a distinctive feature of the gPC bi-level approach and can normally not be achieved using standard robust gain design techniques.

Generally, robustness modifications could be introduced in the cost function (see standard deviation in (7.17)) naturally requiring the solution of a Pareto optimization problem (which is omitted here for the sake of simplicity). Because Pareto problems are generally cumbersome to solve, this section directly uses the probability of fulfilling the desired CC bound, in this case for the angle of attack, as a constraint in the upper level of the bi-level problem. Thus, the bi-level problem formulation in (7.14) is adapted as follows:

$$
\begin{aligned}
\min_{\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}} \quad & J_{\mathrm{UL}} \\
\text{s.t.} \quad & \boldsymbol{\Gamma}_{\boldsymbol{\Theta},\mathrm{lb}} \leq \boldsymbol{\Gamma}_{\boldsymbol{\Theta}} \leq \boldsymbol{\Gamma}_{\boldsymbol{\Theta},\mathrm{ub}} \\
& \mathbb{P}\left[\mathbf{x} \in \mathcal{P}\right] \geq \xi \\
& \quad \text{s.t.} \quad \begin{aligned}
&\mathbf{x}^{(j)} = \mathbf{x}_0^{(j)} + \int_{t=t_0}^{t_f} \dot{\mathbf{x}}^{(j)} \,\mathrm{d}t \\
&\mathbf{x}_P^{(j)} = \mathbf{x}_{0,P}^{(j)} + \int_{t=t_0}^{t_f} \dot{\mathbf{x}}_P^{(j)} \,\mathrm{d}t \\
&\boldsymbol{\Theta}^{(j)} = \boldsymbol{\Theta}_0^{(j)} + \int_{t=t_0}^{t_f} \dot{\boldsymbol{\Theta}}^{(j)} \,\mathrm{d}t \\
&\boldsymbol{\Lambda}^{(j)} = \boldsymbol{\Lambda}_0^{(j)} + \int_{t=t_0}^{t_f} \dot{\boldsymbol{\Lambda}}^{(j)} \,\mathrm{d}t \\
&\dot{\mathbf{x}}^{(j)} = \mathbf{A}^{(j)}\mathbf{x}^{(j)} + \mathbf{B}^{(j)}\mathbf{u}^{(j)} \\
&\dot{\mathbf{x}}_P^{(j)} = \mathbf{A}_P^{(j)}\mathbf{x}_P^{(j)} + \mathbf{B}^{(j)}\boldsymbol{\Lambda}^{(j)}\mathbf{u}^{(j)} + \mathbf{B}^{(j)}\boldsymbol{\Theta}^{(j)}\mathbf{x}^{(j)} \\
&\dot{\boldsymbol{\Theta}}^{(j)} = -\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}\mathbf{x}^{(j)} \left(\mathbf{e}^{(j)}\right)^{\mathsf{T}} \mathbf{P}_L\mathbf{B}^{(j)} \\
&\dot{\boldsymbol{\Lambda}}^{(j)} = -\boldsymbol{\Gamma}_{\boldsymbol{\Lambda}}\mathbf{u}^{(j)} \left(\mathbf{e}^{(j)}\right)^{\mathsf{T}} \mathbf{P}_L\mathbf{B}^{(j)} \\
&\mathbf{u}^{(j)} = \left(\boldsymbol{\Lambda}^{(j)}\right)^{-1} \left(\mathbf{K}_{\mathbf{r}}\mathbf{r} - \boldsymbol{\Theta}^{(j)}\mathbf{x}^{(j)}\right)
\end{aligned}
\end{aligned}
\tag{7.19}
$$

Here, ξ is the desired probability level achieved by the probabilistic/chance constraint (Subsection 2.4.4) on the states. It should be noted that the upper level cost function in (7.19) remains as defined in (7.17).

The evaluation of the CC for the bi-level formulation in (7.19) can be directly done using the methods in Subsection 6.1.1 to estimate the probability and Subsection 6.1.2 to create a differentiable approximation for the nonlinear program (NLP) solver.

Take into account that the CC is evaluated in an integral form, i.e., over the complete time interval. This must be done here as the DEA is generally bad performing with constraints, specifically a lot of constraints which would be present when enforcing the CC at each time point. Thus, the integral representation is a suitable choice for this purpose and defined as follows (already using the sigmoids for the smooth approximation of the indicator function as introduced in Subsection 6.1.2):

$$
\mathbb{P}\left[\mathbf{x} \in \mathcal{P}\right] = \frac{1}{n_s} \sum_{i=1}^{n_s} \left[\frac{1}{t_f - t_0} \int_{t=t_0}^{t_f} s\left[\mathbf{x}^{(i)}; \mathbf{a}_{\mathbf{s}}, \mathbf{b}_{\mathbf{s}}\right] \,\mathrm{d}t \right] \geq \xi \tag{7.20}
$$

It should be noted that the time integration is carried out by the trapezoidal rule and the integral result is appropriately normalized using the time horizon. Furthermore, the number of samples is given by n_s. Thus, the probability is calculated by integrating all n_s random trajectories over time to get the probability of fulfillment for each individual trajectory. Afterward, the probability value of fulfilling the CC is calculated by division through the number of samples.

In this application, the angle of attack α_K is considered as the state that should fulfill the probabilistic constraint $\mathbb{P}\left[\alpha_K \leq 11\,^\circ\right]$ in (7.20) with a desired minimal probability level of $\xi = 99\,\%$. Take into account that the shape and offset parameter of the sigmoid in (7.20) are chosen as $a_s = 250$ and $b_s \approx -0.05\,^\circ$. By this, a very good, i.e., steep

(Subsection 6.1.2), approximation of the original CC is achieved. It should be further noted that this approximation quality could be achieved without a homotopy, as the DEA already provides a feasible initial solution to the sensitivity optimization.

Then, solving the bi-level OCP in (7.19) results in the following optimal adaptation gains for the robust case:

$$\boldsymbol{\Gamma}_{\boldsymbol{\Theta},\text{sen,opt}} \approx \begin{bmatrix} 0.077 & 0 \\ 0 & 52.059 \end{bmatrix} \tag{7.21}$$

It is imminent from comparing the gains in (7.21) with the gains for the bi-level OCP without CC ("not safely robust") in (7.18) that the gain corresponding to the angle of attack is reduced, while the gain related to the pitch rate is significantly increased. The increase in the pitch rate adaptation gain can directly be explained from the fact that the uncertain parameters are mainly, and significantly, influencing the pitch rate dynamics (see (7.1) and (7.12)). Thus, a fast adaptation is required here to reduce the uncertainty influence. On the other hand, the reduction of the adaptation gain for the angle of attack is a consequence of the cost function that still tries to reduce the control area statistics. Due to the increase in control effort caused by the pitch rate adaptation gain, the angle of attack gain must be reduced consequently. This yields worse tracking of the reference command. Thus, for different applications a tracking cost or constraint could also be considered as supplement in (7.19).

Looking at the results, Figure 7.6 depicts the mean value and standard deviation time development of the angle of attack (the dashed black line is the maximal allowed angle of attack). Here, the robust gain result (solid blue line) is the result with the gains as given in (7.21), while the non-robust reference gain case (dashed red line) is simulated with the reference gains in Table 7.2. Additionally, the worst-case approximation, i.e., the trajectory that yields the maximal angle of attack, is depicted (solid magenta: reference gain; solid cyan: optimal gain). It is seen that the angle of attack has slightly worse tracking as already expected from the smaller angle of attack adaptation gain. But, more importantly, the maximal standard deviation is reduced by approximately 33 % with the robust gains in (7.21). Additionally, the maximal standard deviation for the robust gain case does not occur at the first overshoot of the mean value (at approximately 2 s) as in the non-robust case, but at the first undershoot (at approximately 2.8 s). As a consequence, this also directly reduces the maximal angle of attack of the worst-case trajectory. Still, there remains a non-zero standard deviation with the optimal gains after 10 s, which is the already mentioned deteriorated tracking.

The reduction of the maximal standard deviation and the shift to the first undershoot also consequently increases the CC fulfillment probability as seen in Figure 7.7: Here, it is imminent that the maximum failure probability of around 40 % is reduced to a level of around 5 %. Thus, it is very certain that the CC bound for the angle of attack of 11 ° is not

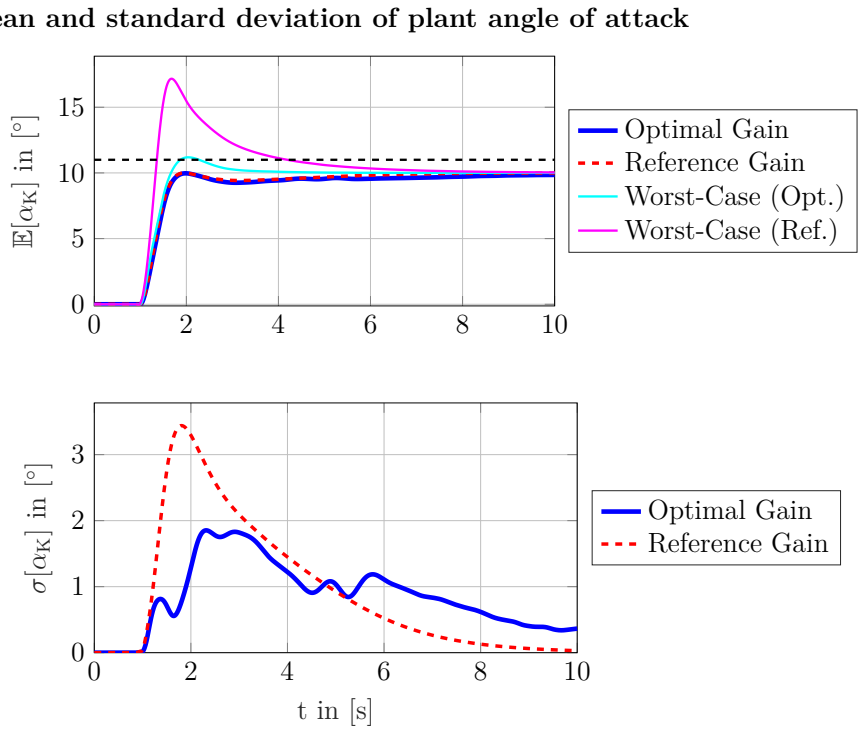

Figure 7.6: *Mean and standard deviation of plant angle of attack reaction to step input including worst-case approximations for reference and optimal adaptation gains.*

violated, even considering the large uncertainty domain in (7.12). Specifically, Figure 7.7 shows that the integrated failure probability is reduced to 0.2 %, which now combines robustness and safety.

Take into account that the increase of the fulfillment probability for the CC comes at the cost of an increased control effort and, especially, a non-smoother control command. This behavior can be examined in Figure 7.8. Here, it is clear that the control effort largely varies for different uncertainties after the step initiation, because of the large standard deviation. This is mainly based on the large adaptation gain for the pitch rate, which tries to directly cancel out the uncertainties in the transient phase. Additionally, the large control effort after the step initiation is required because the CC must be fulfilled.

Concluding the CC-based gain design, Table 7.3 compares the control area statistics and the fulfillment probabilities for the two robust gain and the reference gain results. It should be noted that the control areas are calculated with the control command in radiant rather than in degrees and thus, the values are comparably small. It can be seen that the robust gain results without CC from Subsection 7.3.2 yield the best (i.e., smallest) control area statistics. On the other hand, the robust gains with the CC (Subsection 7.3.3) have the worst (i.e., largest) control area statistics. In specific, the optimality of the mean control area is reduced by around 7 % for the CC-based robust gains, while the optimality of the standard deviation of the control area is reduced around 33 %. By reducing the optimality, an improvement of the robustness in the fulfillment probability of 12 % can be

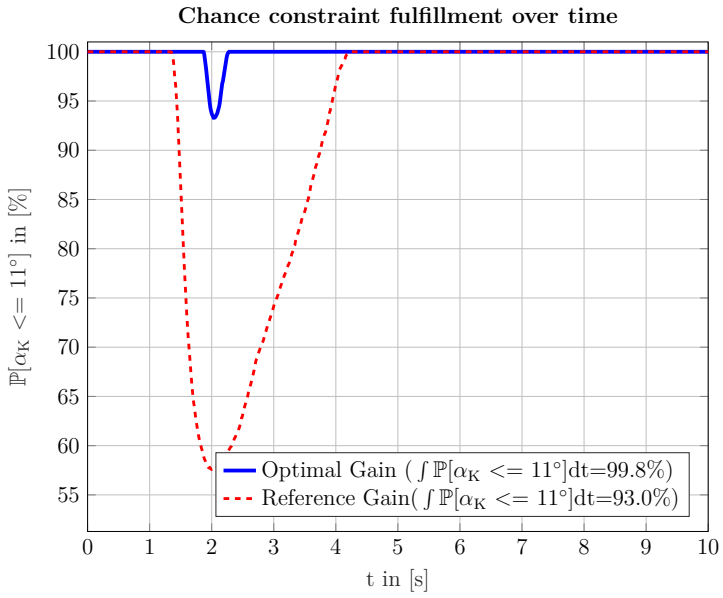

Figure 7.7: *Probability of not exceeding angle of attack limitation of eleven degrees over simulation time for optimal and reference adaptation gains.*

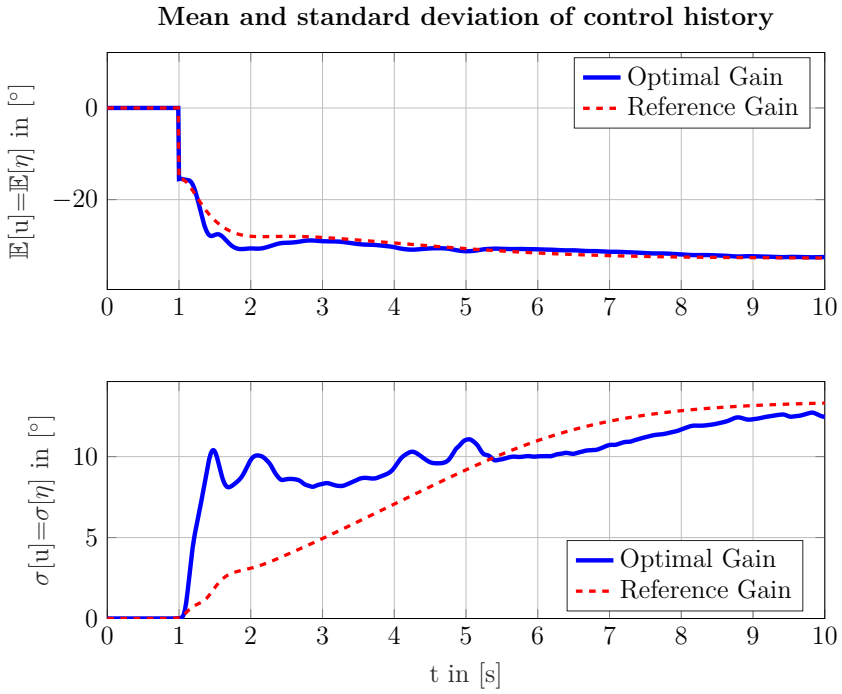

Figure 7.8: *Mean and standard deviation of control history during step input for optimal and reference adaptation gains.*

Table 7.3: *Comparison of adaptation gains for different robust and reference test cases.*

Case	Mean Control Area	Standard Deviation Control Area	Integrated Fulfillment Probability
Reference Gains (Table 7.2)	≈ 2.7792	≈ 1.6313	$\approx 93.2\,\%$
Optimal Gains (see (7.18))	≈ 2.6572	≈ 1.2799	$\approx 87.8\,\%$
Optimal Gains with CC (see (7.21))	≈ 2.8448	≈ 1.7090	$\approx 99.8\,\%$

achieved. This illustrates the well-known trade-off of robustness and optimality in ROC. As the reference case shows, there are multiple gain combination that yield intermediate results. Thus, gains specific to requirements can be calculated as well.

Overall, the example in this chapter has shown that the proposed robust bi-level approach (Contribution 2) to calculate controller gains (or also generally parameters in a dynamic system) is viable and can indeed sufficiently improve the robustness and safety. Especially, if design parameters and constraints in the form of probabilities should be considered, the proposed approach provides a generic framework applicable to multiple design tasks.

Chapter 8

Distribution of Polynomial Chaos Problems for Air Race Application

This chapter introduces an application for the distributed open-loop direct optimal control (DOC) framework introduced in Chapter 4 and thus, shows results for Contribution 3 of this thesis. As the application example, an air race is chosen. Here, a chicane-like maneuver with wind is used to show the robustness improvement that is possible by applying the DOC framework. Take into account that the chicane-like maneuver is taken as the example, because the wind influence, which is used as the uncertainty, is crucial in these high-maneuvering passages as e.g., exceeding the load factor limits as well as hitting a pylon leads to time penalties in the air race, which must be avoided.

The used dynamic model (Section 8.1) as well as initial studies on optimal air race trajectories were originally published in [17, 46]. Based on these initial studies, it can be stated that the air race application of this chapter has proven to be a challenging open-loop direct optimal control problem (OCP) [17, 46]. This is due to the fact that the resulting OCP get very large and multiple local optima can be encountered by the NEWTON-type nonlinear program (NLP) solver. Thus, the application is suited for the DOC approach introduced in Chapter 4, as the methodology splits up a large and complicated OCP into smaller OCPs that should be easier to solve.

To show the applicability of DOC, the chapter is organized as follows: In Section 8.1, the distributed open-loop direct optimal control problem (DOCP) is formulated and an equation of motion (EoM) for each state of the air race aircraft model is introduced. Then, Section 8.2 shows a comparison of the generalized polynomial chaos (gPC) method and Latin hypercube sampling (LHS) to determine the required expansion order for gPC. Afterward, a comparison between the standard generalized polynomial chaos-stochastic collocation framework (gPC-SC) of Subsection 2.3.5 and a DOCP with only a mean value cost function is made in Section 8.3. This is done to show the viability of the DOC approach and verifies it. Finally, Section 8.4 shows how DOC can be used to calculated robust trajectories by introducing robustifications in the cost function.

Table 8.1: *States with bounds, scaling, and offset as well as their dynamic influence used in air race optimization model.*

Description	Symbol	$\mathbf{x}_{\text{lb}}$	$\mathbf{x}_{\text{ub}}$	$\mathbf{x}_{\text{sc}}$	$\mathbf{x}_{\text{off}}$	Unit	Dynamics
x_N-Position in a local frame	x_N	$-\infty$	∞	10^{-2}	0	[m]	Position
y_N-Position in a local frame	y_N	-100	150	10^{-2}	0	[m]	Position
z_N-Position in a local frame	$z_N = -h$	$-\infty$	10	10^{-2}	0	[m]	Position
Kinematic course angle	χ_K	$-\infty$	∞	10^{0}	0	[rad]	Kinematics
Kinematic climb angle	γ_K	$-\frac{\pi}{3}$	$\frac{\pi}{3}$	10^{0}	0	[rad]	Kinematics
Kinematic bank angle	μ_K	$-\infty$	∞	10^{0}	0	[rad]	Kinematics
Kinematic velocity	V_K	25	102.9	10^{-1}	0	$\left[\frac{\text{m}}{\text{s}}\right]$	Translation
Kinematic angle of attack	α_K	-0.17	0.35	10^{0}	0	[rad]	Translation
Kinematic angle of sideslip	β_K	-0.17	0.17	10^{0}	0	[rad]	Translation
Kinematic roll rate	p_K	-7.33	7.33	10^{0}	0	$\left[\frac{\text{rad}}{\text{s}}\right]$	Rotation
Kinematic pitch rate	q_K	$-\pi$	π	10^{0}	0	$\left[\frac{\text{rad}}{\text{s}}\right]$	Rotation
Kinematic yaw rate	r_K	$-\pi$	π	10^{0}	0	$\left[\frac{\text{rad}}{\text{s}}\right]$	Rotation

Table 8.2: *Controls with bounds, scaling, and offset as well as their dynamic influence used in air race optimization model.*

Description	Symbol	$\mathbf{u}_{\text{lb}}$	$\mathbf{u}_{\text{ub}}$	$\mathbf{u}_{\text{sc}}$	$\mathbf{u}_{\text{off}}$	Unit	Influence
Aileron deflection	ξ	$-\frac{\pi}{8}$	$\frac{\pi}{7}$	10^{0}	0	[rad]	Roll Rate
Elevator deflection	η	$-\frac{\pi}{7}$	$\frac{\pi}{7}$	10^{0}	0	[rad]	Pitch Rate
Rudder deflection	ζ	$-\frac{\pi}{6}$	$\frac{\pi}{6}$	10^{0}	0	[rad]	Yaw Rate

8.1 Air Race Problem Formulation and Dynamic Model

The following section summarizes the dynamic equations of a rigid-body aerobatic aircraft. For the definition of coordinate frames and the transformation matrices, Appendix A can be consulted. The following description relies on the work of [46, chs. 3 & 6] as well as [17, chs. 2 & 11] and starts with the problem formulation.

8.1.1 Air Race Problem Formulation

The air race model uses the states with the respective lower and upper bounds $\mathbf{x}_{\text{lb}}$, $\mathbf{x}_{\text{ub}}$, scaling $\mathbf{x}_{\text{sc}}$ as well as offset $\mathbf{x}_{\text{off}}$ defined in Table 8.1. Furthermore, the table introduces the dynamics the state belongs to, which is detailed within the derivation of the EoMs. Additionally, the controls with the respective lower and upper bounds $\mathbf{u}_{\text{lb}}, \mathbf{u}_{\text{ub}}$, scaling $\mathbf{u}_{\text{sc}}$ as well as offset $\mathbf{u}_{\text{off}}$ are given in Table 8.2. Here, also their main influence on the EoMs is stated. It should be noted that the thrust command is not used as a control because previous studies suggested that a time-optional solution of the air race OCP results in a full thrust command along the trajectory [17, p. 236ff.].

Table 8.3: *Gate positions for the air race course and kinematic fly-through conditions used as initial and final boundary conditions for the optimal control problem [17, p. 215f.].*

Phase	IBC_{lb}	IBC_{ub}
1	$[0 \quad 0 \quad 0 \quad -0.174 \quad 0 \quad -0.1745]^\mathsf{T}$	$[0 \quad 0 \quad 0 \quad -0.174 \quad 0 \quad 0.1745]^\mathsf{T}$
2	$[237.63 \quad -58.55 \quad 0 \quad -0.174 \quad 0 \quad -0.1745]^\mathsf{T}$	$[237.63 \quad -58.55 \quad 0 \quad -0.174 \quad 0 \quad 0.1745]^\mathsf{T}$
3	$[565.00 \quad 54.45 \quad 0 \quad -0.172 \quad 0 \quad -0.1745]^\mathsf{T}$	$[565.00 \quad 54.45 \quad 0 \quad -0.172 \quad 0 \quad 0.1745]^\mathsf{T}$
4	$[880.01 \quad -29.25 \quad 0 \quad 0 \quad 0 \quad -0.1745]^\mathsf{T}$	$[880.01 \quad -29.25 \quad 0 \quad 0 \quad 0 \quad 0.1745]^\mathsf{T}$
Phase	FBC_{lb}	FBC_{ub}
4	$[1127.08 \quad -79.46 \quad 0 \quad 0 \quad 0 \quad -0.1745]^\mathsf{T}$	$[1127.08 \quad -79.46 \quad 0 \quad 0 \quad 0 \quad 0.1745]^\mathsf{T}$

Table 8.4: *Final time optimization parameter definition for different phases.*

Phase i	$t_{f,\text{init},i}$	$t_{f,\text{lb},i}$	$t_{f,\text{ub},i}$	$t_{f,\text{sc},i}$	$t_{f,\text{off},i}$
1	2.5 s	0 s	40 s	10^0	10^0
2	6.0 s	0 s	40 s	10^0	10^0
3	9.0 s	0 s	40 s	10^0	10^0
4	12.0 s	0 s	40 s	10^0	10^0

Furthermore, the OCP consists of multiple phases that model the different race gate positions, which are to be passed in wings-level position ($\mu_K = [-10; 10]$ °) as well as in horizontal flight (safety aspect; $\gamma_K = 0$) and a defined direction χ_K. The initial boundary condition (IBC) for the states $\begin{bmatrix} x_N & y_N & z_N & \chi_K & \gamma_K & \mu_K \end{bmatrix}^\mathsf{T}$ define these gate conditions in the beginning of each OCP phase (see Subsection 2.1.2.2) and is defined in Table 8.3. Here, also the final boundary condition (FBC), i.e., the condition at the final gate of the last phase, is defined in the same manner. Take into account that all positions are defined relative to the first gate, where also the local coordinate frame N is fixed.

It should be further noted that the velocity at the first gate, i.e., when entering the track, is constrained to be smaller than 102.9 $\frac{\text{m}}{\text{s}}$ and larger than 100 $\frac{\text{m}}{\text{s}}$. In addition, the OCP constrains the kinematic angle of attack as well as kinematic angle of sideslip at the first gate to be within $-5\,° \leq \alpha_K \leq 5\,°$ and $-0\,° \leq \beta_K \leq 0\,°$ respectively. Additionally, here the rotational rates are enforced to be zero. This is to have a close to steady-state flight condition at the first gate, which is reasonable.

The track definition of Table 8.3 is also visualized in Figure 8.1: Here, the gates are denoted by circles, while a trajectory based on a Dubins path [17, p. 222f.] is used for the visualization. This result is also further on used as an initial guess. It can already be observed from Figure 8.1 that a high maneuverability, especially in the passage from Gate 2 to Gate 3, is required.

As a further part of the OCP, the final times (optimization parameters) of the phases with the initial guess $t_{f,\text{init}}$, respective lower and upper bounds $t_{f,\text{lb}}, t_{f,\text{ub}}$, scaling $t_{f,\text{sc}}$ as well as offset $t_{f,\text{off}}$ are defined in Table 8.4.

In addition to the state, control, and time limitation, the load factor $(n_{T,z})_B$ in negative z_B-direction of the B frame is constrained for all phases (due to air race regulations):

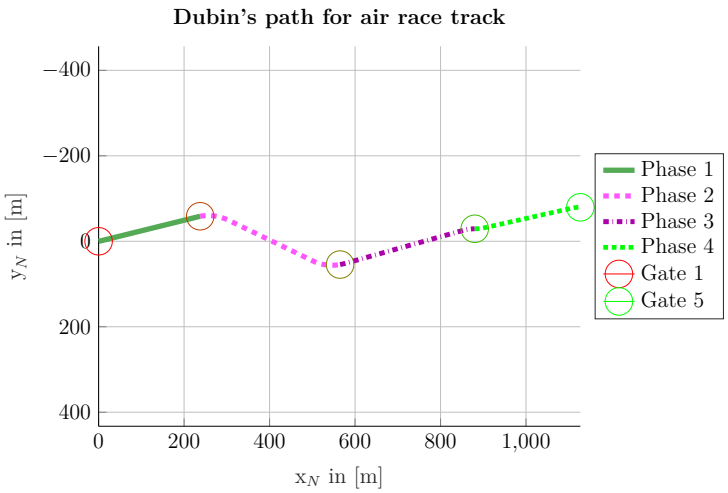

Figure 8.1: *Top view on Dubins path through defined air race track used as initial guess for optimization.*

$$-2 \leq -(n_{T,z})_B = -n_{T,z} \leq 11 \tag{8.1}$$

Furthermore, the aerodynamic angle of attack is limited as follows:

$$-10^\circ \leq \alpha_A \leq 15^\circ \tag{8.2}$$

This limitation is based on the aerodynamic profile of the aircraft's wing and prevents complete stall, i.e., loss of control, of the aircraft [17, p. 217ff.].

The objective of the OCP is to finish the track in minimum time, i.e.,

$$J = t_{f,4}. \tag{8.3}$$

It should be noted that this cost function generally yields a large control effort (especially with steps in the control command channel). Thus, when the optimal solution should be applied to the real aircraft adding a control cost should be considered. This is omitted in this thesis as the influences of the different cost parts (especially the later on used robustness modification) would get mixed, which complicates the analysis of the influences and showing the benefits of the robustness modifications.

For the open-loop direct optimal control (OC) method, there are $n_\tau = 101$ time steps chosen in each phase. This corresponds to $6,064$ optimization parameters and $5,245$ constraints for the standard OCP (i.e., without uncertainties), which can be considered large-scale [17] and is therefore, suitable to show the benefits of the DOC algorithm. The air race OCP is solved to an optimality tolerance of $\epsilon_{\text{opt}} = 10^{-8}$ and a feasibility tolerance of $\epsilon_{\text{feas}} = 10^{-8}$. The NLP solver Interior Point Optimizer (IPOPT) is used for the optimization, applying the linear solver `ma97`. Furthermore, the analytic Hessian, provided by FSD optimal control tool for MATLAB® (FALCON.m), is used.

Table 8.5: *Configuration and references values as well as environment parameters of the air race optimization model [17, p. 218].*

Description	Symbol	Value	Unit
wing reference are	S	8.928	$[\mathrm{m}^2]$
half wing span	s	3.75	[m]
mean aerodynamic chord	$\bar{c}$	1.44	[m]
aircraft mass	m_B	693	[kg]
inertia tensor	$\mathbf{I}$	$\begin{bmatrix} 420.3035 & 0 & 0 \\ 0 & 726.7184 & 0 \\ 0 & 0 & 827.3201 \end{bmatrix}$	$[\mathrm{kg} \cdot \mathrm{m}^2]$
reference thrust	T_{ref}	5500	[N]
reference velocity	$V_{A,ref}$	30	$\left[\frac{\mathrm{m}}{\mathrm{s}}\right]$
thrust command	δ_T	1	$[-]$
air density	ρ	1.225	$\left[\frac{\mathrm{kg}}{\mathrm{m}^3}\right]$
gravitational constant	g	9.80665	$\left[\frac{\mathrm{m}}{\mathrm{s}^2}\right]$

Finally, the uncertain parameters in this example are the northward and eastward wind velocities. Their probability density functions (PDFs) are defined as follows:

$$\begin{aligned} (u_W)_N &\sim \mathcal{U}\left(a = -10\,\frac{\mathrm{m}}{\mathrm{s}},\, b = 10\,\frac{\mathrm{m}}{\mathrm{s}}\right) \\ (v_W)_N &\sim \mathcal{U}\left(a = -10\,\frac{\mathrm{m}}{\mathrm{s}},\, b = 10\,\frac{\mathrm{m}}{\mathrm{s}}\right) \end{aligned} \tag{8.4}$$

These uncertainties influence e.g., the aerodynamic angle of attack as well as the aerodynamic velocity. Thus, they have a significant influence on the dynamic model. It should be noted that the interval of the PDF is chosen such that it is already difficult to control for an aerobatic aircraft (as seen the bounds are approximately 10 % of the maximal flight velocity) and represent wind velocities up until "medium wind" on the Beaufort scale. The Uniform PDF is also meaningful as there is no specific race (i.e., a location) chosen, where a more exact wind distribution could be applied. Thus, a Uniform PDF can be regarded as a worst-case.

8.1.2 Equations of Motion for Aerobatic Aircraft

This section introduces the EoMs for the rigid-body dynamic model on a fixed-flat earth (FFE) (i.e., the earth is non-rotating and a plane). This is a standard assumption in aviation, which only takes place in a restricted air space [122, p. 41f.] as the air race, and thus, is also viable for the example of this thesis. Here, the states (Table 8.1), controls (Table 8.2), and configuration parameters and reference values, as defined in Table 8.5, are used. A detailed introduction to e.g., the coordinate frames can be found in Appendix A.

Position The propagation of the x_N, y_N, and z_N position for a local coordinate system N (FFE), fixed at the initial gate, is defined as follows:

$$\begin{bmatrix} \dot{x}_N \\ \dot{y}_N \\ \dot{z}_N \end{bmatrix} = \begin{bmatrix} V_K \cdot \cos(\chi_K) \cdot \cos(\gamma_K) \\ V_K \cdot \sin(\chi_K) \cdot \cos(\gamma_K) \\ -V_K \cdot \sin(\gamma_K) \end{bmatrix} \tag{8.5}$$

Thus, the position propagation is based solely on the translational states with the dynamics introduced in the following.

Translation The translational EoMs are described by using the kinematic velocity V_K, the kinematic course angle χ_K, and the kinematic climb angle γ_K. From NEWTON's second law, the translational dynamics on a FFE are given as follows:

$$\dot{V}_K = \frac{1}{m_B} \cdot F_{T,x} \tag{8.6a}$$

$$\dot{\chi}_K = \frac{1}{m_B \cdot V_K \cdot \cos(\gamma_K)} \cdot F_{T,y} \tag{8.6b}$$

$$\dot{\gamma}_K = -\frac{1}{m_B \cdot V_K} \cdot F_{T,z} \tag{8.6c}$$

In (8.6a)–(8.6c), the components of the total force vector $\left(\vec{\mathbf{F}}_T\right)_K$, denoted in the kinematic frame K, are computed from the component load factors as follows (see Section 8.1.3):

$$\left(\vec{\mathbf{F}}_T\right)_K = \begin{bmatrix} F_{T,x} & F_{T,y} & F_{T,z} \end{bmatrix}_K^{\mathsf{T}} = m_B \cdot g \cdot \left[(\mathbf{n}_{A+P})_K + (\mathbf{n}_G)_K\right] \tag{8.7}$$

Here, $\mathbf{n}_{A+P}$ combines the aerodynamic and propulsive load factor, while $\mathbf{n}_G$ denotes the gravitational load factor. These are defined in more detail in Subsection 8.1.3.

Attitude The attitude EoMs are described by the kinematic angle of attack α_K, the kinematic bank angle μ_K, and the kinematic sideslip angle β_K as follows:

$$\dot{\mu}_K = \left(\omega_{K,x}^{KB}\right)_K - \tan(\beta_K)\left[\left(\omega_{K,y}^{KB}\right)_K \cos(\mu_K) + \left(\omega_{K,z}^{KB}\right)_K \sin(\mu_K)\right] \tag{8.8a}$$

$$\dot{\alpha}_K = \frac{1}{\cos(\beta_K)}\left[\left(\omega_{K,y}^{KB}\right)_K \cos(\mu_K) + \left(\omega_{K,z}^{KB}\right)_K \sin(\mu_K)\right] \tag{8.8b}$$

$$\dot{\beta}_K = \left(\omega_{K,y}^{KB}\right)_K \sin(\mu_K) - \left(\omega_{K,z}^{KB}\right)_K \cos(\mu_K) \tag{8.8c}$$

In (8.8a)–(8.8c), the components of $\left(\boldsymbol{\omega}_K{}^{KB}\right)_K$ denote the kinematic rotational velocities between the K frame and the B frame (here B rotates with respect to K), denoted in the K frame. These are calculated as follows:

$$\left(\boldsymbol{\omega}_K{}^{KB}\right)_K = \begin{bmatrix} \omega_{K,x}^{KB} \\ \omega_{K,y}^{KB} \\ \omega_{K,z}^{KB} \end{bmatrix}_K = \begin{bmatrix} \dot{\chi}_K \cdot \sin(\gamma_K) \\ -\dot{\gamma}_K \\ -\dot{\chi}_K \cdot \cos(\gamma_K) \end{bmatrix}_K + \mathbf{M}_{KB} \cdot \begin{bmatrix} p_K \\ q_K \\ r_K \end{bmatrix}_B \tag{8.9}$$

The first addend in this equation is known from (8.6b) and (8.6c), while the second addend can be calculated using the rotational dynamics as follows.

Rotation The final set of EoMs for the rigid-body dynamics are the rotational dynamics, obtained from the momentum conservation law (again on a FFE). These describe the evolution of the kinematic roll rate p_K, pitch rate q_K, and yaw rate r_K. Using the aerodynamic moments $\left(\vec{\mathbf{M}}_A\right)_B$ (note that neither propulsion nor gravity create a moment in this example) and the inertia tensor $\mathbf{I}_{BB}$ with respect to the center of gravity in the body-fixed frame, the angular accelerations are computed as follows:

$$\left(\dot{\boldsymbol{\omega}}_K{}^{OB}\right)_B = \begin{bmatrix}\dot{p}_K & \dot{q}_K & \dot{r}_K\end{bmatrix}^\mathsf{T} = \mathbf{I}_{BB}^{-1}\left[\left(\vec{\mathbf{M}}_A\right)_B - \left(\boldsymbol{\omega}_K{}^{OB}\right)_B \times \mathbf{I}_{BB} \cdot \left(\boldsymbol{\omega}_K{}^{OB}\right)_B\right] \tag{8.10}$$

Thus, in order to solve the EoMs, the forces, i.e., load factors, and moments are required. These are introduced in the following.

8.1.3 Forces/Load Factors and Moments for Aerobatic Aircraft

The aerodynamic load factor $(\mathbf{n}_A)_A$ in the aerodynamic frame A and the propulsive load factor $(\mathbf{n}_P)_B$ in the B frame are computed as follows:

$$(\mathbf{n}_A)_A = \frac{\bar{q}\cdot S}{m_B \cdot g}\begin{bmatrix}-C_D & C_Q & -C_L\end{bmatrix}^\mathsf{T}, \qquad \bar{q} = \frac{\rho}{2}{V_A}^2 \tag{8.11a}$$

$$C_D = C_{D0} + (C_{Q\beta_A}\cdot\beta_A)^2 \cdot k_{\beta_A} + (C_{L\alpha_A}\cdot\alpha_A)^2 \cdot k \tag{8.11b}$$

$$C_Q = C_{Q\beta_A}\cdot\beta_A + \frac{S}{V_A}\cdot C_{Qp_A}\cdot p_A + \frac{S}{V_A}\cdot C_{Qr_A}\cdot r_A + C_{Q\zeta}\cdot\zeta \tag{8.11c}$$

$$C_L = C_{L0} + C_{L\alpha_A}\cdot\alpha_A + \frac{\bar{c}}{2\cdot V_A}\cdot C_{Lq_A}\cdot q_A + \eta\cdot C_{L\eta} \tag{8.11d}$$

and

$$(\mathbf{n}_P)_B = \begin{bmatrix}\delta_T \cdot \frac{V_{A,ref}}{V_A}\cdot\frac{T_{ref}}{m_B\cdot g} & 0 & 0\end{bmatrix}^\mathsf{T} \tag{8.12}$$

Here, $\bar{q}$ is the dynamic pressure calculated from the density ρ and the aerodynamic velocity V_A. The mass of the aircraft is denoted by m_B, while the gravitational constant is given by g. The required aerodynamic force coefficients are introduced in Table 8.6, while the constants and parameters (especially of the thrust) are as defined in Table 8.5.

It is important to note that the aerodynamic velocity is calculated using the kinematic and the wind velocity as follows:

$$V_A = V_K - V_W, \qquad V_W = \sqrt{(u_W)_N^2 + (v_W)_N^2} \tag{8.13}$$

Thus, the major influence of the wind on the dynamics of the aircraft becomes clear.

It is further important to note that the aerodynamic angles can be calculated as well using the kinematic and the wind influence. This is necessary when calculating the forces and moments and the aerodynamic angles are defined as follows:

$$\begin{aligned} \alpha_A &= \operatorname{atan}\left(\frac{(w_A)_B}{(u_A)_B}\right), \quad \beta_A = \operatorname{atan}\left(\frac{(v_A)_B}{\sqrt{(u_A)_B^2 + (w_A)_B^2}}\right) \\ (\mathbf{u}_A)_B &= \begin{bmatrix} u_A & v_A & w_A \end{bmatrix}_B^T = \begin{bmatrix} u_K & v_K & w_K \end{bmatrix}_B^T - \begin{bmatrix} u_W & v_W & w_W \end{bmatrix}_B^T \end{aligned} \tag{8.14}$$

Here, the aerodynamic velocity components are calculated from the wind and kinematic velocity and are defined in the body-fixed frame. Therefore, the transformation matrices from Appendix A must be applied appropriately.

Aerodynamic and Propulsive Load Factors in the Kinematic Frame K The sum of the aerodynamic load factor $(\mathbf{n}_A)_K$ and the propulsive load factor $(\mathbf{n}_P)_K$, required in (8.7), is calculated as follows:

$$(\mathbf{n}_{A+P})_K = \mathbf{M}_{KA}\,(\mathbf{n}_A)_A + \mathbf{M}_{KB} \cdot (\mathbf{n}_P)_B \tag{8.15}$$

It should be noted that the propulsive load factor must be transformed using the transformation matrix $\mathbf{M}_{KB}$ (see Appendix A) from the B frame to the K frame. Similarly, the aerodynamic load factor must be transformed using the transformation matrix from the A frame to the K frame (see Appendix A).

Gravitational Load Factors The gravitational load factor $(\mathbf{n}_G)_K$ in the K frame, required in (8.7), is computed as follows:

$$(\mathbf{n}_G)_K = \begin{bmatrix} -\sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix}^\mathsf{T} \tag{8.16}$$

Thus, the gravitational load factor depends on the climb/descent angle of the trajectory.

Load Factor in the B frame Finally, the total load factor $(\mathbf{n}_T)_B$ in the B frame is computed by transforming the total load factor in the K frame $(\mathbf{n}_T)_K$ to the B frame using the transformation matrix $\mathbf{M}_{BK}$ (Appendix A):

$$(\mathbf{n}_T)_B = \mathbf{M}_{BA} \cdot (\mathbf{n}_T)_A \tag{8.17}$$

This load factor is required to impose the path constraint in (8.1) within the DOCP.

Table 8.6: *Aerodynamic force parameters used in the rigid-body air race optimization model [17, p. 218].*

Description	Symbol	Value	Unit
zero drag coefficient	C_{D0}	0.0295	$[-]$
induced drag coefficient	k	0.05134	$[-]$
induced drag coefficient (slip)	k_{β_A}	1.696770	$\left[\frac{1}{\text{rad}}\right]$
drag due to aileron	$C_{D,\xi}$	0.05	$\left[\frac{1}{\text{rad}}\right]$
drag due to elevator	$C_{D,\eta}$	0.05	$\left[\frac{1}{\text{rad}}\right]$
drag due to rudder	$C_{D,\zeta}$	0.05	$\left[\frac{1}{\text{rad}}\right]$
side force coefficient due to slip	C_{Q,β_A}	-0.589355	$\left[\frac{1}{\text{rad}}\right]$
side force coefficient due to roll rate	C_{Q,p_A}	0.042480	$\left[\frac{1}{\text{rad}}\right]$
side force coefficient due to yaw rate	C_{Q,r_A}	0.048340	$\left[\frac{1}{\text{rad}}\right]$
side force coefficient due to rudder	$C_{Q,\zeta}$	-0.195313	$\left[\frac{1}{\text{rad}}\right]$
zero lift coefficient	$C_{L,0}$	0.055	$[-]$
side force coefficient due to roll rate	C_{L,α_A}	4.75	$\left[\frac{1}{\text{rad}}\right]$
side force coefficient due to yaw rate	$C_{L,\eta}$	-0.073242	$\left[\frac{1}{\text{rad}}\right]$
side force coefficient due to rudder	C_{L,q_A}	-3.479492	$\left[\frac{1}{\text{rad}}\right]$

Table 8.7: *Aerodynamic moment parameters used in the rigid-body air race optimization model [17, p. 219].*

Description	Symbol	Value	Unit
roll coefficient due to slip	C_{l,β_A}	0.024902	$\left[\frac{1}{\text{rad}}\right]$
roll coefficient due to roll rate	C_{l,p_A}	-0.583008	$\left[\frac{1}{\text{rad}}\right]$
roll coefficient due to yaw rate	C_{l,r_A}	0.087891	$\left[\frac{1}{\text{rad}}\right]$
roll coefficient due to aileron	$C_{l,\xi}$	-0.2126	$\left[\frac{1}{\text{rad}}\right]$
roll coefficient due to roll rate	$C_{l,\zeta}$	0.001	$\left[\frac{1}{\text{rad}}\right]$
zero pitch coefficient	$C_{m,0}$	-0.004883	$[-]$
pitch coefficient due to elevator	$C_{m,\eta}$	-0.317383	$\left[\frac{1}{\text{rad}}\right]$
pitch coefficient due to pitch rate	C_{m,q_A}	-16.930176	$\left[\frac{1}{\text{rad}}\right]$
pitch coefficient due to angle of attack	C_{m,α_A}	-0.145406	$\left[\frac{1}{\text{rad}}\right]$
yaw coefficient due to slip	C_{n,β_A}	0.149902	$\left[\frac{1}{\text{rad}}\right]$
yaw coefficient due to roll rate	C_{n,p_A}	0.014648	$\left[\frac{1}{\text{rad}}\right]$
yaw coefficient due to yaw rate	C_{n,r_A}	-0.4731	$\left[\frac{1}{\text{rad}}\right]$
yaw coefficient due to aileron	$C_{n,\xi}$	-0.0073	$\left[\frac{1}{\text{rad}}\right]$
yaw coefficient due to rudder	$C_{n,\zeta}$	0.170898	$\left[\frac{1}{\text{rad}}\right]$

8.1.4 Aerodynamic Moments

The aerodynamic moment $\left(\vec{\mathbf{M}}_A\right)_B$ in the B frame for the rotational EoMs are composed (neglecting influences of thrust and gravitation) as follows:

$$\left(\vec{\mathbf{M}}_T\right)_B = \bar{q} \cdot S \begin{bmatrix} s \cdot C_l & \bar{c} \cdot C_m & s \cdot C_n \end{bmatrix}^\mathsf{T} \tag{8.18}$$

Here, the following aerodynamic model, with the required aerodynamic moment coefficients defined in Table 8.7, is used:

$$C_l = C_{l\xi} \cdot \xi + \frac{s}{V_A} \cdot C_{lp_A} \cdot p_A + \frac{s}{V_A} \cdot C_{lr_A} \cdot r_A + C_{l\beta_A} \cdot \beta_A \tag{8.19a}$$

$$C_m = C_{m0} + C_{m\eta} \cdot \eta + \frac{\bar{c}}{2 \cdot V_A} \cdot C_{mq_A} \cdot q_A + C_{m\alpha_A} \cdot \alpha_A \tag{8.19b}$$

$$C_n = C_{n\zeta} \cdot \zeta + \frac{s}{V_A} \cdot C_{np_A} \cdot p_A + \frac{s}{V_A} \cdot C_{nr_A} \cdot r_A + C_{n\beta_A} \cdot \beta_A \tag{8.19c}$$

Thus, the aerodynamic moments are mainly depending on the controls. Additionally, the uncertain wind is influencing the dynamics by the aerodynamic velocity as well as the aerodynamic angles.

8.2 Accuracy Determination for Polynomial Chaos Expansion by Comparison to Latin-Hypercube Sampling

Due to the complexity and nonlinearity of the air race OCP, a first issue is to define a suitable gPC expansion order that represents the system dynamics with uncertainties accurately. Therefore, this section is used to find a suitable expansion order by first of all looking at and analyzing the uncertainty influence in the dynamic equations, introduced in the previous section. Additionally, a comparison to LHS results, and specifically the 95 %-confidence intervals (CIs), is made. For this case, the LHS results are obtained by calculating the exact optimal trajectories at random samples.

At first, a general look at the EoMs can be taken to estimate the required gPC expansion order: One main influence by the wind in the EoMs results in a change of the aerodynamic velocity (see (8.13)) and consequently influences e.g., the load factor calculation in (8.11a). Due to the fact that the load factor is depending on the aerodynamic velocity quadratically and thus, by the definition of the aerodynamic velocity in (8.13), the largest polynomial influence by the wind is at least quadratically as well. As the resulting forces and moments then influence the EoMs linearly, the quadratic influence remains unchanged. It should be noted that this quadratic influence should be comparably small with respect to the kinematic velocity as the wind can only be around 10 % of the kinematic velocity.

A further influence of the wind is related to the aerodynamic angles, e.g., the aerodynamic angle of attack: Here, the relation is generally described by trigonometric functions (see (8.14)). Although these functions are actually highly nonlinear (at least cubic), the application in this thesis allows for a linearized look at them, because the angles are "small" enough. Thus, the influence of the wind velocity on the aerodynamic angles is well-described by at least a linear relation.

Summarizing, the highest polynomial degree from this look at the model dynamics suggests that an expansion order of three, i.e., quadratic, should be sufficient to describe the dynamic system accurately by gPC. This simple deduction is verified in the following using a comparison to LHS.

In Figure 8.2, a comparison of the mean trajectories (flight paths) between a LHS with $n_s = 500$ successful samples (solid blue line; Figure 2.4) and the gPC orders $d_i = 2, 3, 4, 5$ is made. Here, the same gPC order is used for both the north- and eastward wind uncertainty as they have a similar influence on the dynamic system and thus, should be expanded similarly. It should be reminded here that the LHS results will not fully resemble the real statistical moments due to infeasible optimal solutions (see Figure 2.4). Additionally, the CIs are calculated based on the assumption that the considered quantity is GAUSSIAN-distributed (Subsection 2.2.3). Thus, the results must be discussed with care.

Take into account that the calculation of the LHS result took around $14,402\,\mathrm{s}\,(\approx 4h)$ on a personal computer[1], while the optimization time for the gPC expansion (sequential evaluation) ranged from $229.2\,\mathrm{s}$ ($d_i = 2$) to $1,180.5\,\mathrm{s}$ ($d_i = 5$). Thus, the required computational time is reduced by at least 91 %. Once more, this shows the necessity to apply the gPC method to calculate the DOC results, as the calculation of the LHS would be required in each iteration of the DOC connection problem (including line searches). This would not be feasible in application as the results should be obtained in a reasonable amount of time.

The analysis begins by showing the optimal trajectory through the track: Therefore, in Figure 8.2, the five gates are depicted by circles (first gate red, final gate green). Overall, the four gPC expansion orders recover the mean solution obtained from LHS very well. Thus, as already discussed in Subsection 2.3.6, even small expansion orders are suitable for a mean value estimated. This is also illustrated in Figure 8.3, which shows the trajectory position separately. It should be noted that the passing times of the gates are depicted by the vertical lines, which show that LHS and the different gPC orders match very well. Furthermore, the 95 %-CI of the LHS is given by the upper and lower bound as well as the different gPC orders. The gPC expansion orders give very good matches for all positional states. This is also seen when looking at the standard deviation for the positions

[1]Architecture: x64 Intel® Core™ i7-6700K CPU @4.00GHz, 16.0GB RAM

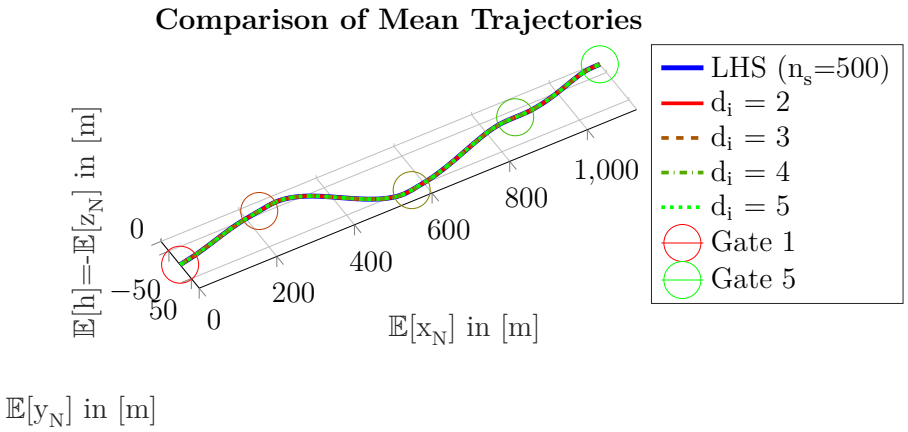

Figure 8.2: *Mean value of trajectory through gates for different generalized polynomialchaos orders compared to Latin-hypercube sampling result.*

in Figure 8.4. Once more, all higher gPC orders show a good approximation quality with minor benefits for the third order, i.e., quadratic, expansion. Only the second order expansion yields too small results, especially in the second phase.

Another interesting observation that can be made for the trajectories in Figure 8.3 and Figure 8.4 is that the altitude h variation is generally quite large, especially in the second phase. This is due to the fact that specifically the load factor constraint must be fulfilled for each uncertain parameter and therefore, the steep turns, as required during the chicane-like maneuver, need an adaptation of the flight velocity, which is here achieved by altitude changes. This is further examined in the following.

As mentioned, depending on the uncertainty value, adaptations to especially the flight velocity are required to fulfill both the load factor constraint as well as the gate (phase) constraints. Among other things, this is achieved by changing the altitude, which in turn changes the velocity. This is visualized in Figure 8.5 and Figure 8.6 that show the mean and standard deviation of the translational states. Here, it is seen that the velocity is significantly reduced after the first phase and by this creates an increased standard deviation. Thus, it can be concluded that the pilot must make significant adaptations, specifically in the velocity, for different wind conditions. It can further be seen that the kinematic angles of attack as well as sideslip remain on a similar level, especially in the standard deviation, over the different phases. Specifically, the angle of attack does not reach its allowed maximum value, which shows that the wind also reduces the allowable range due to the constraint on the aerodynamic angle of attack to avoid stall (see (8.2)). Additionally, this already suggests that the load factor is the restricting constraint. Thus, the pilot must make very specific adaptations, based on the current environmental conditions, to follow an optimal trajectory and avoid critical stall situations. Once more, all gPC orders give good results compared to the 95 %-LHS CI, with the third order being slightly superior and the second order being the worst.

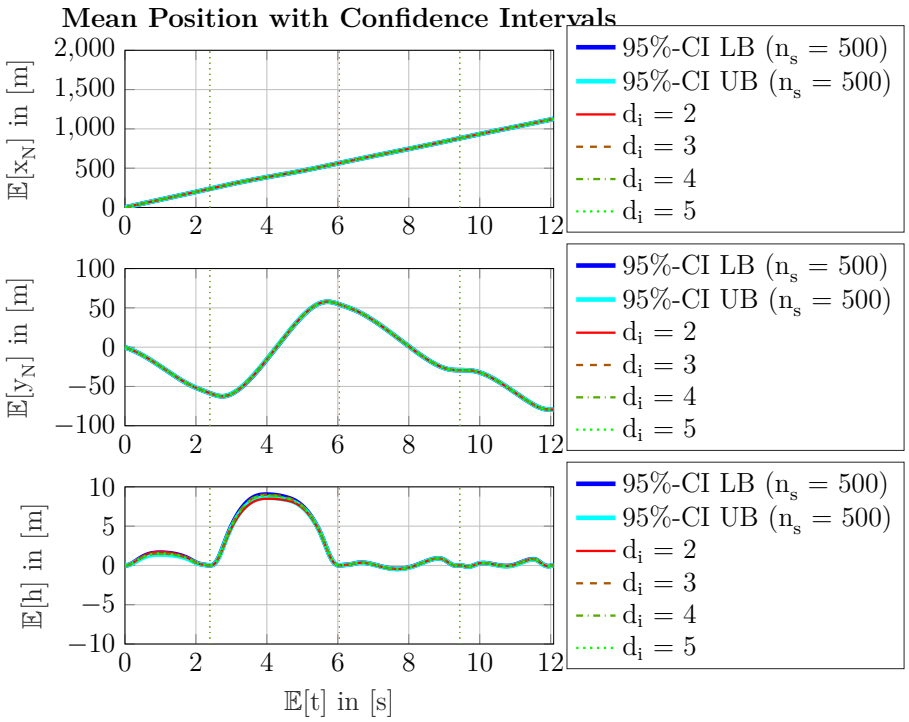

Figure 8.3: *Mean value of position through air race track for different generalized polynomial chaos orders compared to 95 %-confidence intervals of Latin-hypercube sampling result.*

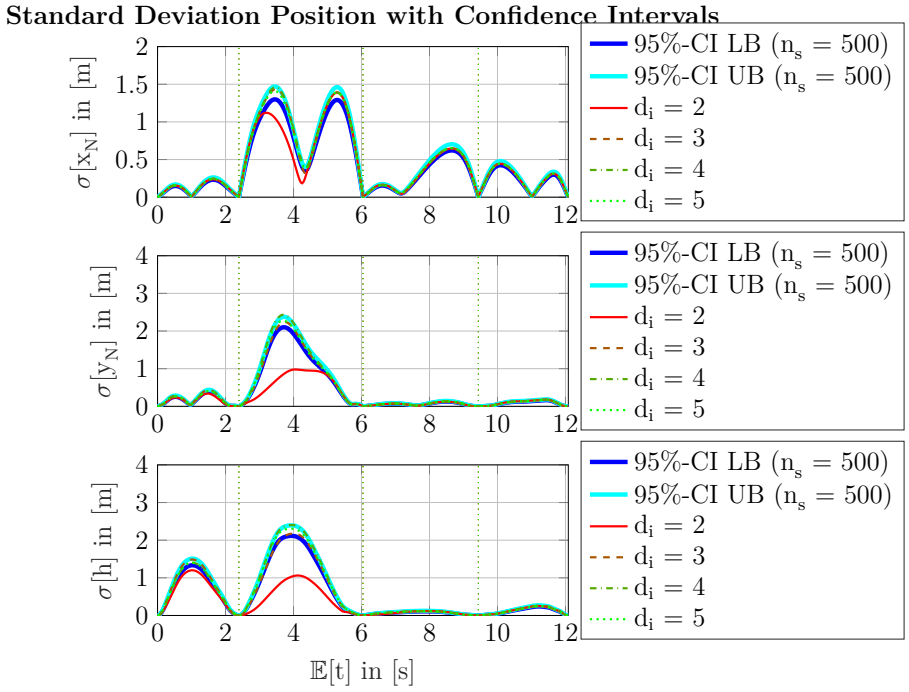

Figure 8.4: *Standard deviation of position through air race track for different generalized polynomial chaos orders compared to 95 %-confidence intervals of Latin-hypercube sampling result.*

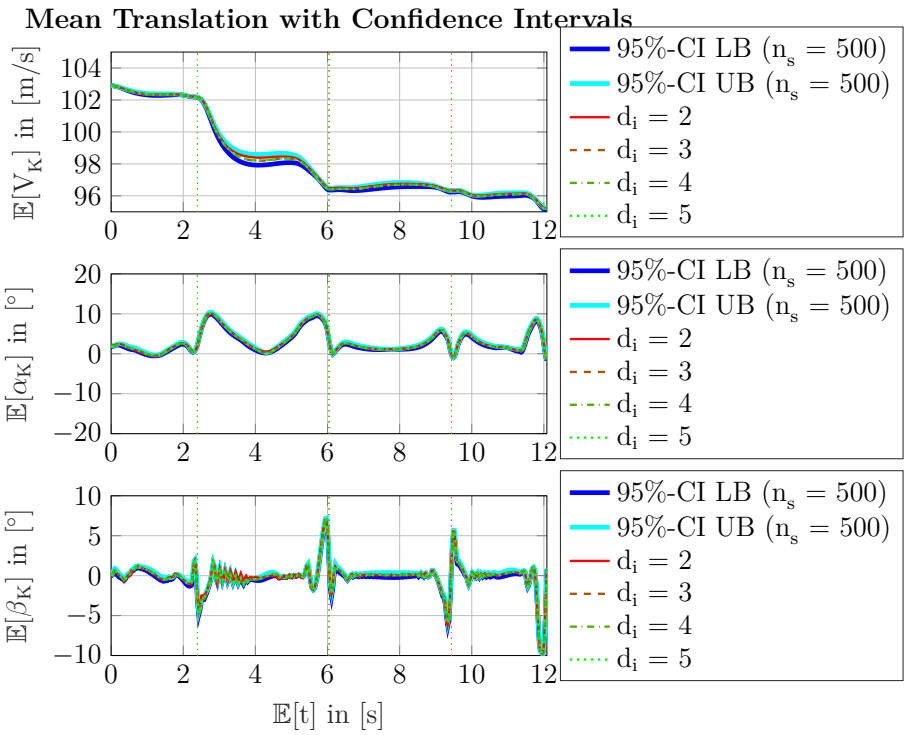

Figure 8.5: *Mean value of translational states on air race track for different generalized polynomial chaos orders compared to* 95 %*-confidence intervals of Latin-hypercube sampling result.*

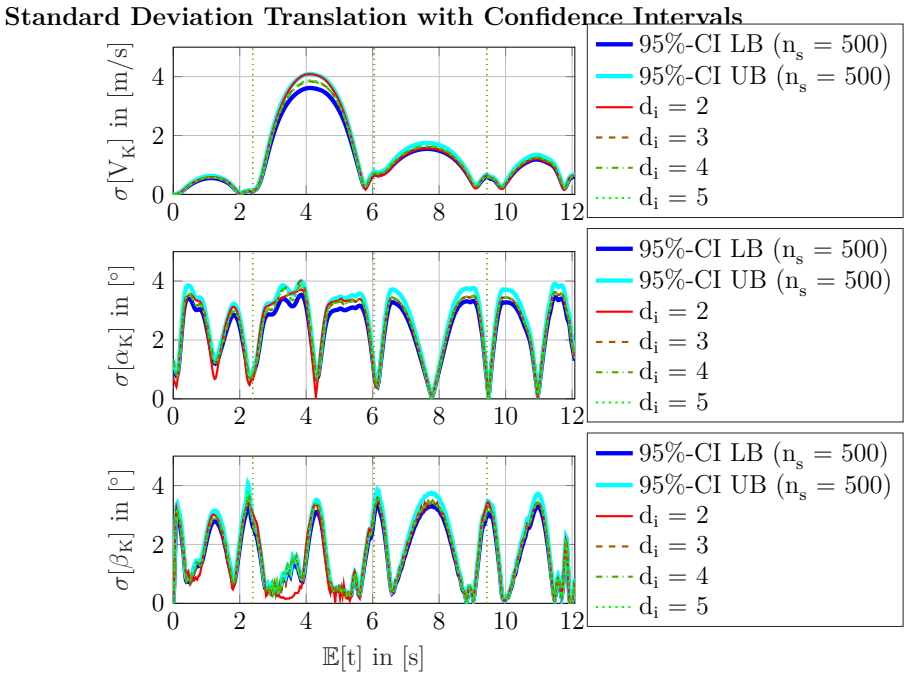

Figure 8.6: *Standard deviation of translational states on air race track for different generalized polynomial chaos orders compared to* 95 %*-confidence intervals of Latin-hypercube sampling result.*

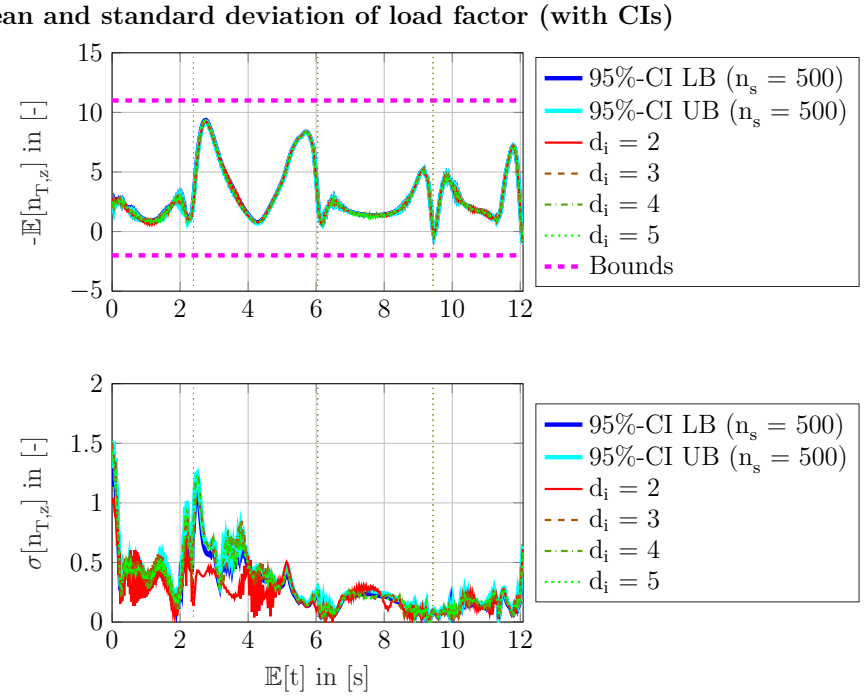

Figure 8.7: *Mean and standard deviation of load factor through air race track for different generalized polynomial chaos orders compared to* 95 %*-confidence intervals of Latin-hypercube sampling result (horizontal, dashed magenta lines: lower and upper bound).*

As a final part of the comparison of gPC and LHS, as well as the derivation of a suitable gPC expansion order, a look at the statistical moments of the load factor is made in Figure 8.7. This quantity is of special interest for the pilot as exceeding the limits in (8.1), which are depicted by the dashed, horizontal magenta lines, yields penalties, which reduce the chances of winning for the pilot. Thus, fulfilling the bounds with a high probability is generally desired. First of all, it can again be seen that the third order gPC expansion once more yields the best approximation results. The second order gives the worst approximation (especially in the standard deviation). Therefore, the third order gPC expansion is the final choice for the order used for the DOC in the following as it has proven to be the best approximator compared to the results of LHS with its CIs, which can be assumed to be a good approximation of the real solution. Furthermore, it is seen in Figure 8.7 that the standard deviation of the load factor is especially large in the first two phases. In these two phases, the mean value is also particularly close to the bound values (visualized by horizontal dashed magenta lines), which in turn means that the probability to violate the constraint will be fairly large here. Thus, robustifications to reduce the violation probability are introduced in the following.

But first of all, to further show the applicability of the DOC algorithm to the air race problem and also to validate the methodology, the next section introduces a DOCP with only a mean cost function, here, the time. This DOCP is perfectly decoupled, as derived in (4.7), Section 4.2, and Section 4.4, and thus, can be compared to the standard gPC-SC.

8.3 Comparison of Distributed and Standard Polynomial Chaos Result

This section introduces a comparison of the DOC algorithm, developed in Chapter 4, to the standard gPC-SC of Subsection 2.3.5, i.e., the sequential optimization of the OCPs at the stochastic collocation (SC) nodes. This is done to verify the DOC methodology based on a distributed mean value in the cost function, where it is still possible to have a comparison by the gPC-SC (Subsection 2.3.5). Additionally, this section verifies the Karush-Kuhn-Tucker (KKT) condition based derivation in Section 4.4. In specific, the mean optimal flight time through the air race track is distributed in the cost function. From (4.7) it is known that this DOCP is perfectly decoupled and thus, no connection variables must be introduced as well as that the DOCP can be solved in a single step. Consequently, the DOCP for each SC node is given as follows:

$$\begin{aligned} \min_{\mathbf{z}^{(j)}} \quad & J^{(j)}\left(\mathbf{z}^{(j)}\right) = \alpha^{(j)} \cdot {t_f}^{(j)} \\ \text{s.t.} \quad & \mathbf{z}_{\text{lb}}^{(j)} \leq \mathbf{z}^{(j)} \leq \mathbf{z}_{\text{ub}}^{(j)}, \\ & \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)}; \boldsymbol{\theta}^{(j)}\right) = \dot{\mathbf{x}}^{(j)}, \\ & \mathbf{c}^{(j)}\left(\mathbf{z}^{(j)}\right) \leq \mathbf{0}, \\ & \boldsymbol{\psi}^{(j)}\left(\mathbf{z}^{(j)}\right) = \mathbf{0} \end{aligned} \tag{8.20}$$

Here, the decoupled natural of the DOCP with only a mean value cost function is seen and a comparison to the gPC-SC is possible because the OCPs are mathematically equivalent. The main difference in the DOCP case is that the SC weight $\alpha^{(j)}$ is present in the cost function as the mean value should be optimized. Take into account that, although mathematically equivalent, this changes the scaling of the NLP and thus, can converge to a slightly different optimum based on the feasibility and optimality tolerances.

In the following, results for a third order gPC expansion are depicted as this has shown to be viable in Subsection 8.2. It should be noted that the connected OCP with a third order expansion and $n_\tau = 101$ would already have $54,576$ optimization variables and $47,197$ constraints and therefore, would be hard to solve on a general desktop computer (due to memory and solver restrictions). Thus, the DOC approach is viable for this application, as here only the unconnected problem with $6,064$ optimization parameters and $5,245$ constraints must be solved.

After solving the OCPs, only minor differences can be seen, when looking at the calculated cost function, i.e., the mean final time: The result from the DOC framework calculates is $\hat{t_f}_{,\text{distr}}^{(0)} \approx 12.08261\,\text{s}$, while the reference gPC-SC problem results in $\hat{t_f}_{,\text{ref}}^{(0)} \approx 12.08260\,\text{s}$. Thus, the minor difference is in the order of the NLP tolerances and therefore, the results show very good resemblance. Overall, this also shows that no systematic error is introduced by the DOC approach.

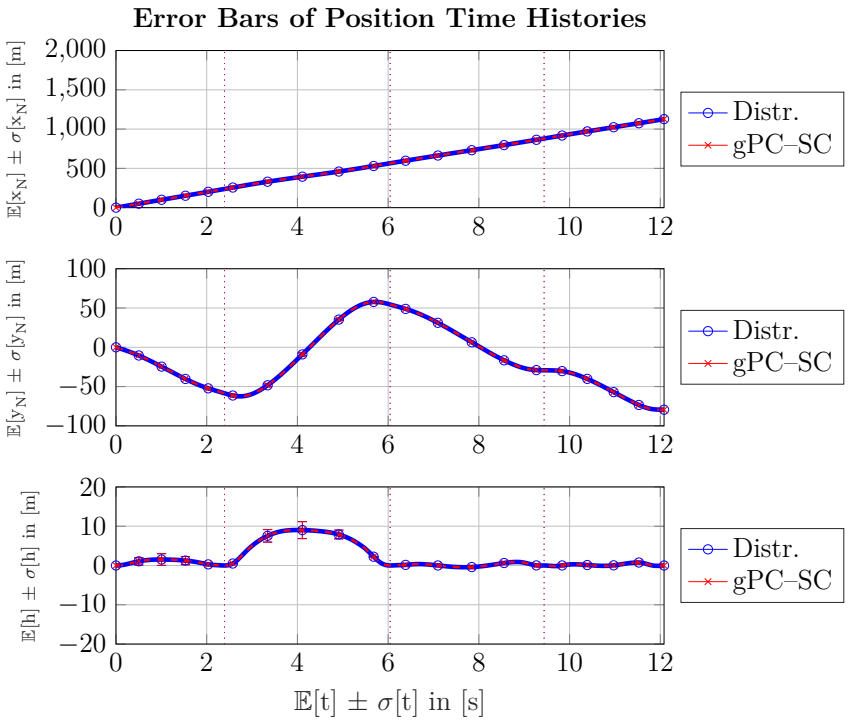

Figure 8.8: *Mean trajectory with error bars on air race track comparing standard generalized polynomial chaos expansion with distributed solution for mean final time.*

This is further illustrated by Figure 8.8, which shows the comparison of the position states for the air race course. In the following, the solid blue line with circles denotes the results of the DOCP in (8.20) ("Distr."), while the dashed red line with crosses is the result from the gPC-SC ("gPC–SC."). The results show the errors bars, i.e., the mean value and the standard deviation. These error bars are used for the states as well as the time, thus giving the spread in both dimensions. It is seen in Figure 8.8 that the positional accuracy of the DOC as well as the gPC-SC is good. The magnitude of the standard deviation is also well matching over the phases, which is depicted by the error bars. It should be noted that the dotted, vertical lines show the phase times, which are also well matching, i.e., overlying.

After the verification of the DOC algorithm via the positional states, a further verification by means of the translational states is given in Figure 8.9: Here, the error bars of the translational states, i.e., kinematic velocity, angle of attack, and angle of sideslip, are shown. Once more, the match for the mean values is good for the optimal trajectory. The standard deviations visualized by the error bars show a good match in the magnitude as well. It can especially be seen that the velocity is largely varying, which is mainly a consequence of the wind uncertainty and the constraint fulfillment.

Continuing with the verification, Figure 8.10 shows the comparison of the load factor statistics for the distributed and the standard gPC solution over the mean time. The matching is very well for both mean and standard deviation. Some minor differences between the distributed and the reference standard deviation occur in the middle of the first and

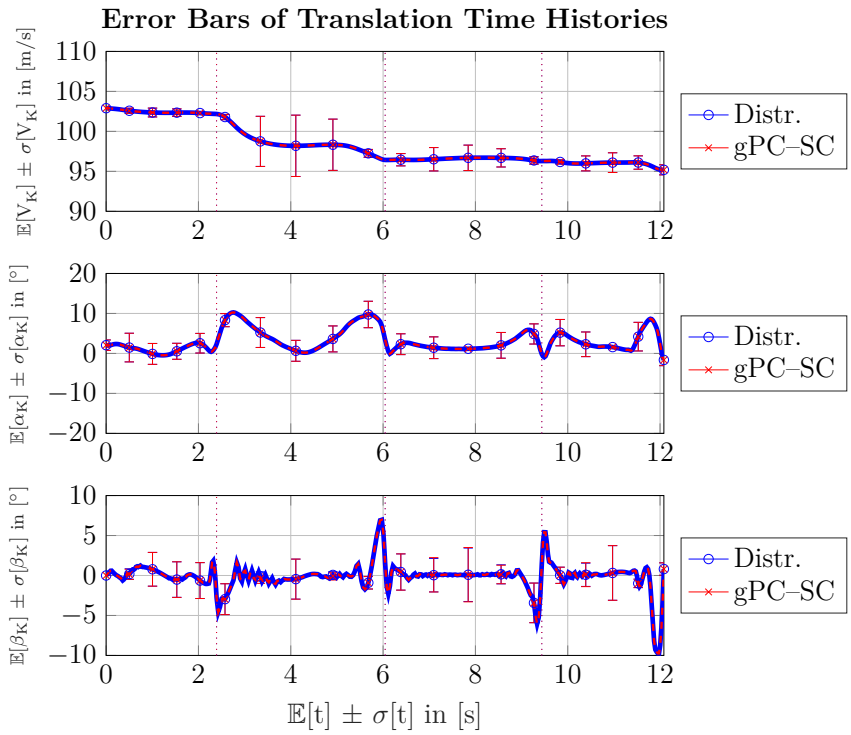

Figure 8.9: *Translational states with error bars comparing standard generalized polynomial chaos expansion with distributed solution for mean final time.*

beginning of the third phase, where the distributed result seems to be slightly more aggressive, i.e., creating sharper changes ("high frequency oscillations") and overall larger values in the standard deviation. This high frequency oscillation might also be based on the NLP tolerances. It can be seen that the mean load factor is close to its limits in the first and second phase, where there is also a significant standard deviation. Thus, exceeding the desired limits becomes fairly likely.

This is also visualized in Figure 8.11 for only the distributed results and over the mean time: Here, the dashed-black lines denote the interval $\pm 3\sigma\,[n_{T,z}]$ of the load factor. This interval is standard in robustness analysis, as it creates a high confidence in the safety of the results: For instance, a Gaussian-distributed variable would imply that at least 99.7 % are within this interval. As the assumption of having a Gaussian-distributed variable is normally a strong one, probability inequalities such as the Chebyshev's inequality (Theorem B.14) or the Vysochanskij–Petunin inequality (Theorem B.15) can be applied as well, which give probabilities of at least 88.8 % and 95.0 % respectively. As these probability inequalities, compared to the assumption of a Gaussian PDF, are very conservative, the $\pm 3\sigma$ has proven to be viable in stochastic applications. If it is fulfilled, the result is generally considered to be robust. As seen in Figure 8.11, the optimal trajectory fulfills the desired probability interval for most parts of the trajectory and thus, a high confidence that the trajectory will not exceed the load factor bounds

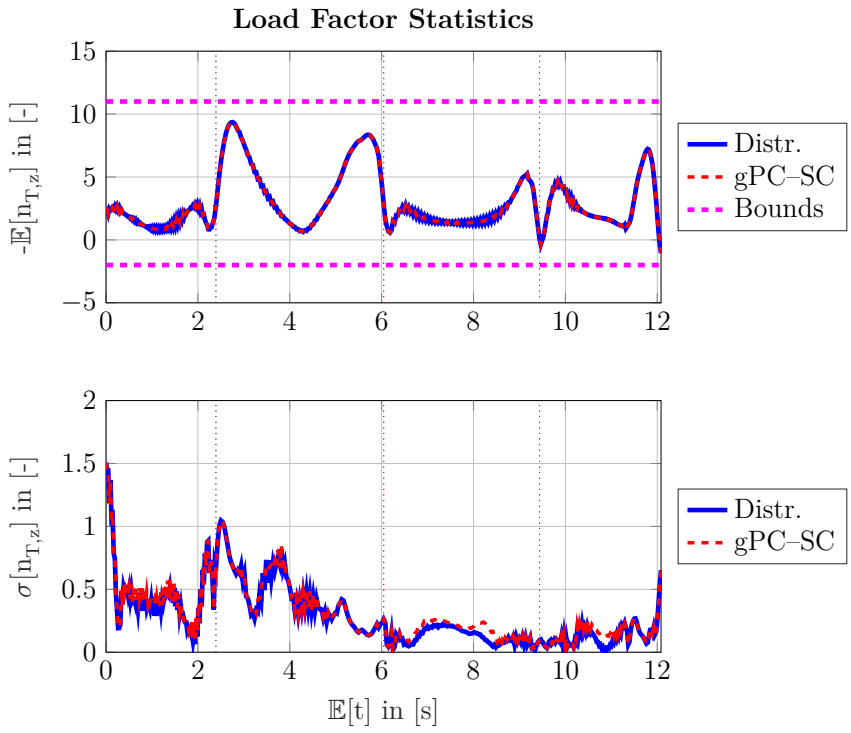

Figure 8.10: *Comparison of load factor statistics between distributed and standard generalized polynomial chaos for distributed solution with mean final time (horizontal, dashed magenta lines: lower and upper bound).*

is established. Still, especially in the first and second phase, there are multiple longer time periods of exceedance, which are dealt with in the following by applying robustness modifications on the basic air race DOCP in (8.20).

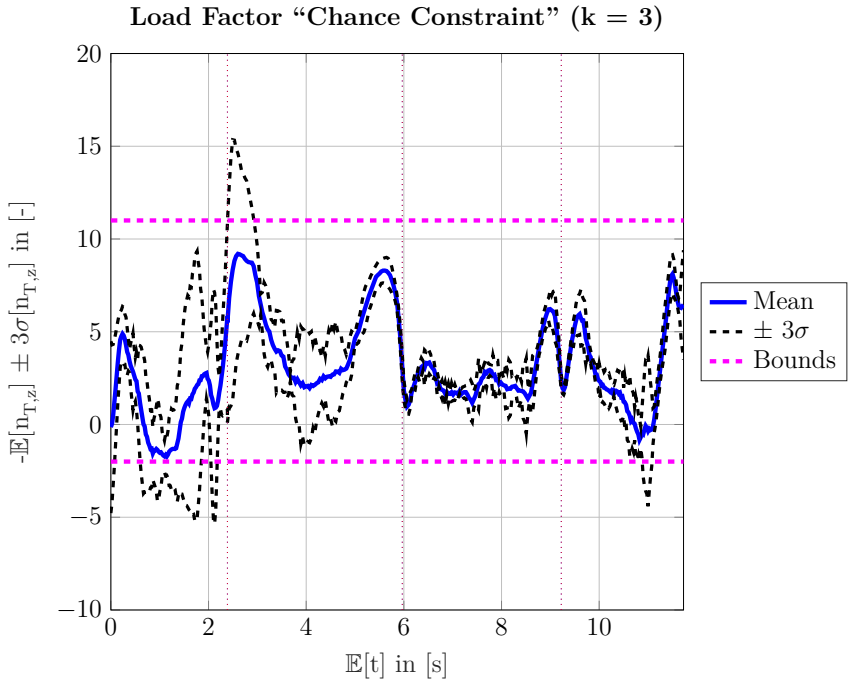

Figure 8.11: *Mean load factor with three standard deviation interval and bound values for distributed solution (horizontal, dashed magenta lines: lower and upper bound).*

8.4 Distributed Optimal Control with Robustifying Load Factor Cost

Compared to the previous section, which only considered a distributed mean value as the cost function, this section introduces an additional robustness modification. Thus, this DOCP is no longer solvable by the standard gPC-SC in Subsection 2.3.5, but can be solved efficiently using the DOC approach. Generally, the variance of the load factor is added to the cost function to inherently reduce the probability of violating the load factor constraint.

To this end, the basic DOCP in (8.20) is adapted as follows:

$$\begin{aligned} \min_{\mathbf{z}^{(j)}} \quad & J^{(j)}\left(\mathbf{z}^{(j)}\right) + \widetilde{J}^{(j)}\left(\mathbf{z}^{(j)};\mathbf{q}\right) \\ \text{s.t.} \quad & \mathbf{z}_{\text{lb}}^{(j)} \leq \mathbf{z}^{(j)} \leq \mathbf{z}_{\text{ub}}^{(j)}, \\ & \mathbf{f}^{(j)}\left(\mathbf{z}^{(j)};\boldsymbol{\theta}^{(j)}\right) = \dot{\mathbf{x}}^{(j)}, \\ & \mathbf{c}^{(j)}\left(\mathbf{z}^{(j)}\right) \leq \mathbf{0}, \\ & \boldsymbol{\psi}^{(j)}\left(\mathbf{z}^{(j)}\right) = \mathbf{0} \end{aligned} \tag{8.21}$$

Here, the cost function addends are defined as follows:

$$
\begin{aligned}
J^{(j)}\left(\mathbf{z}^{(j)}\right) &= \alpha^{(j)} \cdot {t_f}^{(j)} + w \cdot \int_{t=t_0{}^{(j)}}^{t_f{}^{(j)}} \left[n_{T,z}^{(j)}\right]^2 \left[\alpha^{(j)}\right]^2 \sum_{m=1}^{M-1} \left[\Phi^{(m)}\left(\boldsymbol{\theta}^{(\boldsymbol{j})}\right)\right]^2 \mathrm{d}t \\
\widetilde{J}^{(j)}\left(\mathbf{z}^{(j)}; \mathbf{q}\right) &= w \cdot \int_{t=t_0{}^{(j)}}^{t_f{}^{(j)}} \sum_{j \neq i} n_{T,z}^{(j)} \boldsymbol{\nu}^{(i)} \alpha^{(j)} \alpha^{(i)} \sum_{m=1}^{M-1} \Phi^{(m)}\left(\boldsymbol{\theta}^{(\boldsymbol{j})}\right) \Phi^{(m)}\left(\boldsymbol{\theta}^{(\boldsymbol{i})}\right) \mathrm{d}t
\end{aligned}
\tag{8.22}
$$

The factor w is a weighting factor to weigh the cost influences. Thus, ultimately a Pareto problem (Subsection 2.4.1) between the mean time and the load factor variance is solved.

It should be noted that the connection problem, required to update the connection variables, $\mathbf{q} = \boldsymbol{\nu}$, for the DOCP in (8.21), is solved using the optimization problem formulation in (4.5). Therefore, the NLP solver IPOPT is used with a feasibility tolerance of $\epsilon_{\text{feas,CP},*} = 10^{-7}$ as well as an optimality tolerance of $\epsilon_{\text{opt,CP},*} = 10^{-3}$. Take into account that it is difficult to achieve a higher optimality tolerance due to the tolerances of the DOCPs themselves. Still, the chosen tolerance is small enough for the DOC application case, especially considering the normal magnitude of the load factor.

In the following, the results of the robust DOC are looked at: At first, Table 8.8 shows the different cases that were evaluated in this example. Here, the reference case ($w = 0$), i.e., no robustification in the cost function by the variance of the load factor, is the DOC result from Section 8.3. The other five cases describe different weights ($w = 0.001, 0.005, 0.01, 0.05, 0.1$) used for the robust cost function in (8.22). Generally, the table also shows the optimal time through the specified track (Table 8.3 and Figure 8.1) as well as the integrated area of the mean and standard deviation of the load factor. These integrals were calculated in a post-processing step: Here, it can first of all be seen that the optimal time is increased by less than 0.2 %, while increasing the robustness of the trajectory (with increasing Pareto weight) is increased by up to 44 %. This is the already mentioned trade-off between optimality and robustness and is also visualized in Figure 8.12 by an estimated Pareto frontier. Here, it becomes clear that a reduction in the load factor standard deviation requires an increase in the flight time. The desired trade-off must be specified by the user. It should be noted that the time increase of the robust results is smaller than the time penalty for the pilot, if e.g., exceeding the load factor limit (at least 1 s; see[2]). Thus, the applicability of the robust trajectories in the real air race is reasonable and of no disadvantage for the pilot.

Generally, this behavior, i.e., increasing the robustness by reducing the optimality, is an indicator of a less aggressive and thus, safer and more robust trajectory, which is the goal that should be achieved by the robust DOC. It should be further noted that the mean load factor area in Table 8.8 is not significantly altered by the robustification, which suggests that a similar mean trajectory is calculated, while the robustification mainly changes the spread around this trajectory.

[2] `https://airrace.redbull.com/de/rules` "OVER G" (retrieved on 20.07.2019)

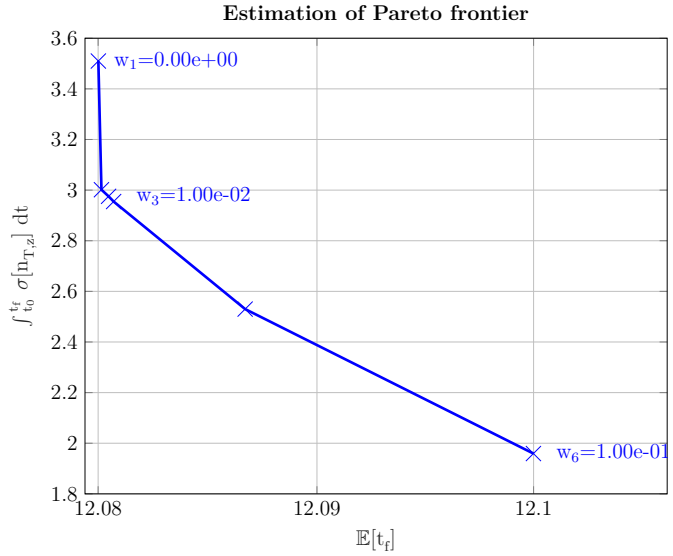

Figure 8.12: *Estimation of Pareto frontier of the distributed optimal control air race results showing the trade-off between the optimal flight time and the integrated load factor standard deviation.*

Table 8.8: *Comparison of robust distributed optimal control results with different Pareto weights and reference solution.*

Case	Optimal Time	Relative Optimality Reduction	Mean Load Factor Area	Standard Deviation Load Factor Area	Relative Robustness Improvement
$w = 0$ (Ref.)	12.082610 [s]	–	−36.172372 [–]	3.509992 [–]	–
$w = 0.0010$	12.082750 [s]	0.001155 %	−36.302507 [–]	3.001506 [–]	14.486801 %
$w = 0.0050$	12.083073 [s]	0.003833 %	−36.387159 [–]	2.975478 [–]	15.228353 %
$w = 0.0100$	12.083302 [s]	0.005721 %	−36.412630 [–]	2.954847 [–]	15.816121 %
$w = 0.0500$	12.089251 [s]	0.054950 %	−36.531652 [–]	2.529981 [–]	27.920588 %
$w = 0.1000$	12.102292 [s]	0.162901 %	−36.556320 [–]	1.959223 [–]	44.181556 %

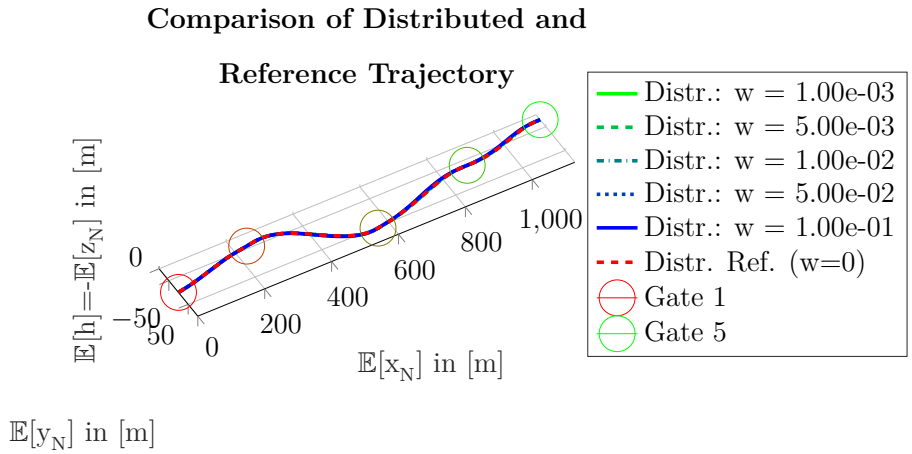

Figure 8.13: *Comparison of robust and reference trajectories with distributed robust cost function.*

The fact that mainly the spread of the trajectories is influenced by the DOC, which is also desired, can furthermore be seen in Figure 8.13 and Figure 8.14. These show the mean position states of the trajectories, which are overall fairly similar. It should be noted once more that the dashed-vertical lines depict the phase times, i.e., when the gates are passed by the different trajectories. The reference DOC result (denoted by "Distr. Ref. ($w = 0$)") is further on depicted by the dashed red line, while the robust trajectories are shown in colors from green (denoted by "Distr."; smallest Pareto weight: $w = 0.001$) to blue (largest Pareto weight: $w = 0.1$) with different line types. The main difference is in the altitude state, which is, as already introduced before, the main influence factor of the optimization for the thrust/velocity control and thus, robustness.

This behavior is also seen in Figure 8.15, which shows the standard deviation of the position states. Here, it is visible that the standard deviation in x_N, y_N, and z_N direction is reduced for the robust DOC results.

The already mentioned decrease of the kinematic velocity is also visualized in Figure 8.16 by error bars: Here, it is seen that the robust trajectories achieve a smaller mean velocity than the reference case. Furthermore, it can be seen that the standard deviation for the robust cases is reduced in the final two phases compared to the reference case. To achieve this reduction, it is necessary to allow a wider range of velocities, i.e., an increased standard deviation, in the second phase. This is a further trade-off in robust optimization, i.e., that improvements on the global behavior (here in the context of the load factor standard deviation) normally lead to deterioration in local behavior. For the robust DOC results only minor differences can be seen, but generally smaller Pareto weights remain at a higher mean value, i.e., closer to the reference trajectory, with larger standard deviations.

Other than the kinematic velocity, Figure 8.16 also shows the kinematic angle of attack error bar. Here, it can be seen that the robust results are less aggressive, i.e., do not change the magnitude as fast and also do not reach as high angles of attack as the reference (e.g., end of third phase as well as final gate). This improves the robustness as well, because

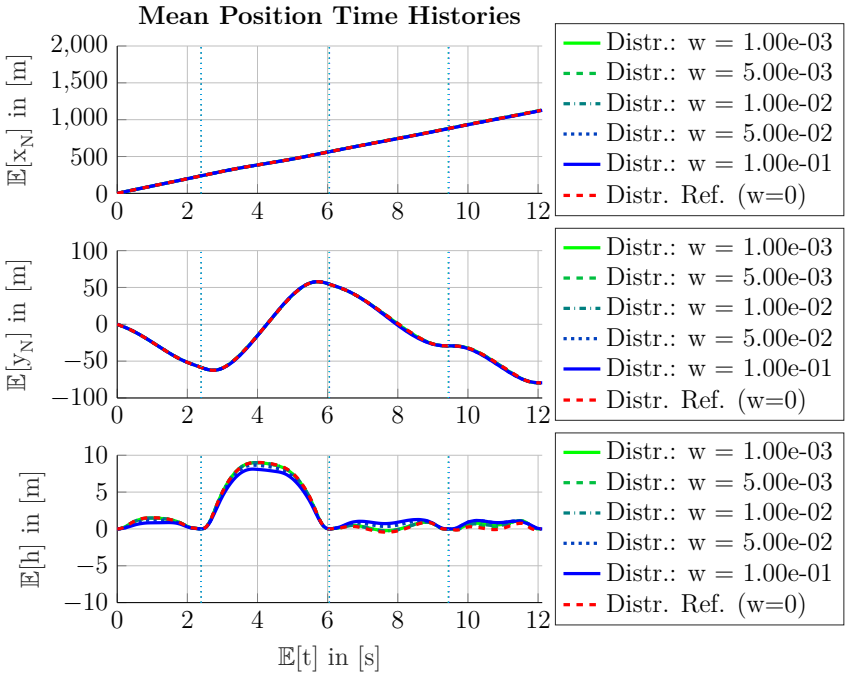

Figure 8.14: *Mean position states for robust and reference trajectories with distributed robust cost function.*

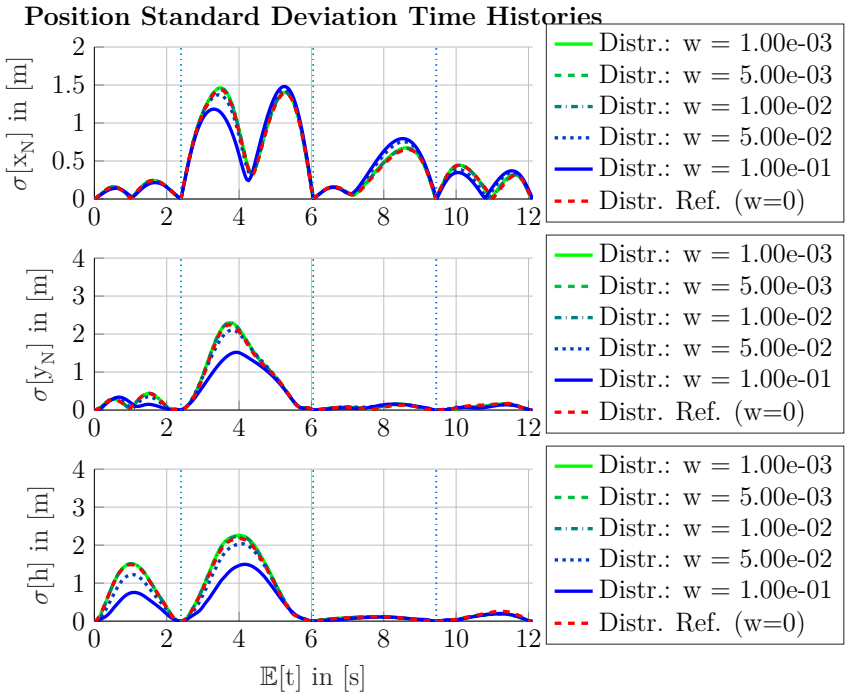

Figure 8.15: *Standard deviation position states for robust and reference trajectories with distributed robust cost function.*

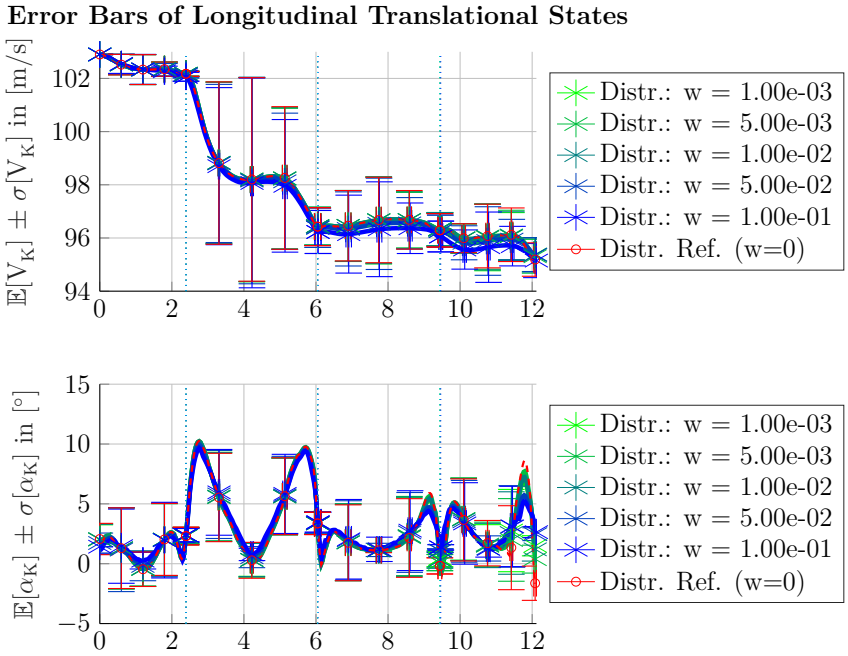

Figure 8.16: *Error bars of kinematic angle of attack and velocity for robust and reference trajectories with distributed robust cost function.*

the probability of exceeding the allowed angle of attack (both aerodynamic as well as kinematic) becomes less likely. Thus, the robustification in the load factor inherently has a positive effect on another critical quantity for a safe flight, which could be expected as load factor and angle of attack (i.e., the lift) are closely related. It should be noted that it is not necessarily the case that a robustification has a positive effect on other critical system quantities. Also negative effects can be encountered and thus, it is required to critically evaluate the made robustness modifications.

As already mentioned and seen in Figure 8.10 and Figure 8.11 of Section 8.3, the load factor constraint is critical to fulfill with the wind uncertainty. Here, Figure 8.11 showed that the load factor constraints are difficult to fulfill in especially the first and the second phase. Thus, the robustness modification in (8.22) was introduced in the DOC framework to reduce the standard deviation of the load factor and improve the robustness of the solution. The effect of the robustness modification is illustrated in Figure 8.17, which shows the mean and standard deviation of the load factor. It can be observed in the figure that the mean load factor is less aggressive especially in the first and final phase. Together with the reduction of the standard deviation (there is an increased reduction when increasing the Pareto weight), this results in a more robust trajectory.Additionally, this makes it possible to fly fairly similar trajectories, which was also already observed in Figure 8.15 by looking at the positional standard deviations. Take into account that the less aggressive robust trajectories also make the aircraft easier to maneuver for the pilot.

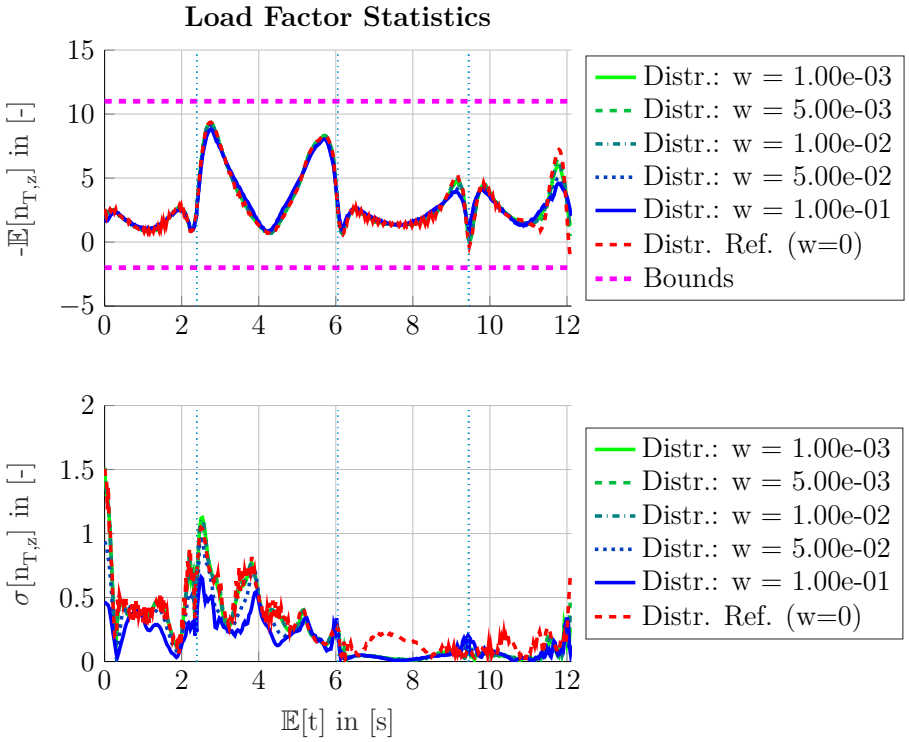

Figure 8.17: *Load factor statistics for robust and reference trajectories with distributed robust cost function (horizontal, dashed magenta lines: lower and upper bound).*

To conclude the robust DOC example, Figure 8.18 shows the load factor fulfillment probability, i.e., the probability with whom the deterministic lower and upper bound of the load factor are fulfilled. The probability is calculated by sampling the optimal trajectory, specifically the gPC expansion, with $n_s = 5000$ random samples in a post-processing step and checking for failures. It can be seen that the non-robust, reference trajectory may fail the load factor bounds by around 5 % in the first phase. The Pareto weights $w = 0.05$ already significantly reduce the failure probability, with around 2 % of the failures occurring in the beginning of the second phase. This is natural as here a large load factor is required to turn (Figure 8.17) and a significant load factor standard deviation is still present. Additionally, the velocity and its standard deviation is still large (Figure 8.16). Finally, the larger Pareto weights, e.g., $w = 0.1$, then even further improve the fulfillment probability and no more failures in the load factor bounds are detected. This is mainly a consequence of the further reduction in both the mean as well as the standard deviation of load factor (Figure 8.17). It should be noted that this result does not suggest that there are absolutely no failure scenarios remaining, but merely that the failure probability is relatively small. A more detailed analysis of small failure probabilities requires the subset simulation (SubSim), which is used for chance-constrained open-loop direct optimal control (CC-OC) in the following example.

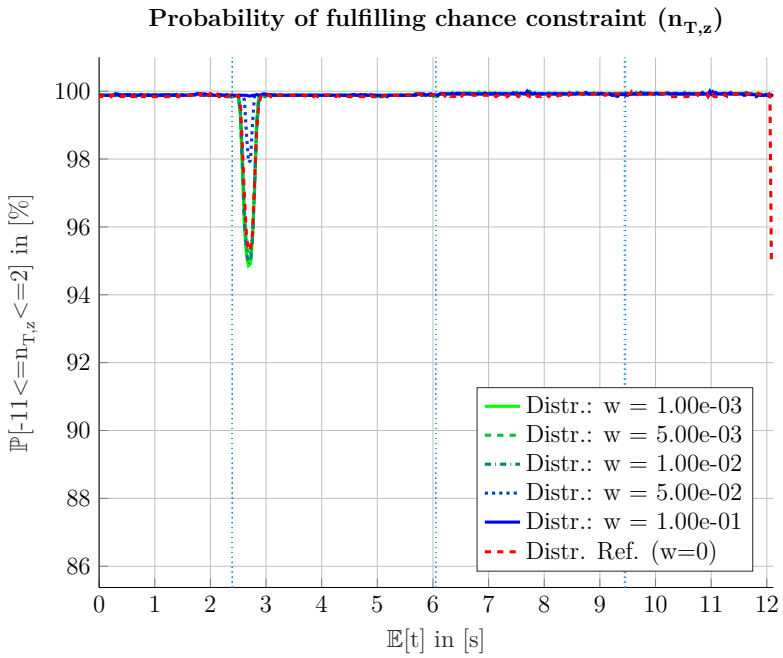

Figure 8.18: *Load factor chance constraint fulfillment for robust and reference trajectories with distributed robust cost function (horizontal, dashed magneta lines: lower and upper bound).*

Concluding, the DOC framework (Contribution 3) has proven to be viable in application, especially for large-scale OCPs like the air race. The method has shown that it can be applied to calculate robust, optimal trajectories. The next chapter continues with a further example of the robust open-loop direct optimal control (ROC) frameworks of this thesis, this time mainly in the SubSim-based CC-OC context for obstacle avoidance.

Chapter 9

Quadcopter Obstacle Avoidance Maneuver using Chance Constraints

This chapter introduces the robust open-loop direct optimal control (ROC) of a quadcopter using the chance-constrained open-loop direct optimal control (CC-OC) framework introduced in Chapter 6. Both the Monte Carlo analysis (MCA) based procedure for "frequent" events (Section 6.1) as well as the subset simulation (SubSim) based approach for rare-events (Section 6.2) are looked at. Thus, results for Contribution 4 and Contribution 5 of this thesis are examined. Here, the quadcopter is optimized to achieve a time-optimal trajectory around obstacles. Both quadcopter as well as obstacle parameters are considered uncertain and chance constraints (CCs) are applied to minimize the probability of hitting any obstacle.

At first, Section 9.1 introduces the dynamic model of quadcopter as well as the model of the obstacles. Then, the CC-OC of the quadcopter is dealt with: Here, Section 9.2 determines the required generalized polynomial chaos (gPC) expansion order for the CC-OC by a comparison with Latin hypercube sampling (LHS). Afterward, Section 9.3 shows the results for the CC-OC using the MCA-based ("frequent" event) sampling approach from Section 6.1, while Section 9.4 extends the previous results to the SubSim-based CC-OC from Section 6.2.

9.1 Definition of Dynamic Model

The dynamic model of the quadcopter is based on a rigid-body moving on a fixed-flat earth (FFE) (i.e., the earth is non-rotating and a plane). These assumptions are standard in aviation, especially when mainly being interested in the dynamic as well as performance behavior [122, p. 41f.] and were also already used for the air race application (Section 8.1). For the quadcopter model, the states are defined as given in Table 9.1, while the controls are as given in Table 9.2. These tables also directly include the bounds, the scalings, and the offsets used in the open-loop direct optimal control problem (OCP) formulation.

Table 9.1: *State definition of the rigid-body quadcopter model with bounds, scalings, and offsets for the optimization.*

Description	Symbol	Unit	Lower Bound	Upper Bound	Scaling	Offset
x position of quadcopter	x_N	[m]	0	40	10^{-1}	0
y position of quadcopter	y_N	[m]	-10	10	10^{0}	0
z position of quadcopter	z_N	[m]	-130	0	10^{0}	0
x body velocity of quadcopter	u_K	$[\frac{\text{m}}{\text{s}}]$	0	10	10^{0}	5
y body velocity of quadcopter	v_K	$[\frac{\text{m}}{\text{s}}]$	-2	2	10^{0}	0
z body velocity of quadcopter	w_K	$[\frac{\text{m}}{\text{s}}]$	-5	5	10^{0}	0
Roll angle of quadcopter	Φ	[rad]	$-\frac{\pi}{4}$	$\frac{\pi}{4}$	10^{0}	0
Pitch angle of quadcopter	Θ	[rad]	$-\frac{\pi}{4}$	$\frac{\pi}{4}$	10^{0}	0
Yaw angle of quadcopter	Ψ	[rad]	$-\infty$	∞	10^{0}	0
Roll rate of quadcopter	p_K	$[\frac{\text{rad}}{\text{s}}]$	$-\frac{\pi}{8}$	$\frac{\pi}{8}$	10^{0}	0
Pitch rate of quadcopter	q_K	$[\frac{\text{rad}}{\text{s}}]$	$-\frac{\pi}{8}$	$\frac{\pi}{8}$	10^{0}	0
Yaw rate of quadcopter	r_K	$[\frac{\text{rad}}{\text{s}}]$	$-\frac{\pi}{4}$	$\frac{\pi}{4}$	10^{0}	0

Table 9.2: *Control definition of the rigid-body quadcopter model with bounds, scalings, and offsets for the optimization.*

Description	Symbol	Unit	Lower Bound	Upper Bound	Scaling	Offset
Speed of rotor 1	ω_1	$[\frac{\text{rad}}{\text{s}}]$	0	785.4	10^{-2}	350
Speed of rotor 2	ω_2	$[\frac{\text{rad}}{\text{s}}]$	0	785.4	10^{-2}	350
Speed of rotor 3	ω_3	$[\frac{\text{rad}}{\text{s}}]$	0	785.4	10^{-2}	350
Speed of rotor 4	ω_4	$[\frac{\text{rad}}{\text{s}}]$	0	785.4	10^{-2}	350

A schematic of the quadcopter, with its body-fixed frame B and the rotation direction of the rotors, $\omega_{1,\dots,4}$, is given Figure 9.1. It should be noted that the thrust of each rotor points in negative body-fixed z-axis direction. Additionally, the reference point for each equation of motion (EoM) R coincides with the center of gravity.

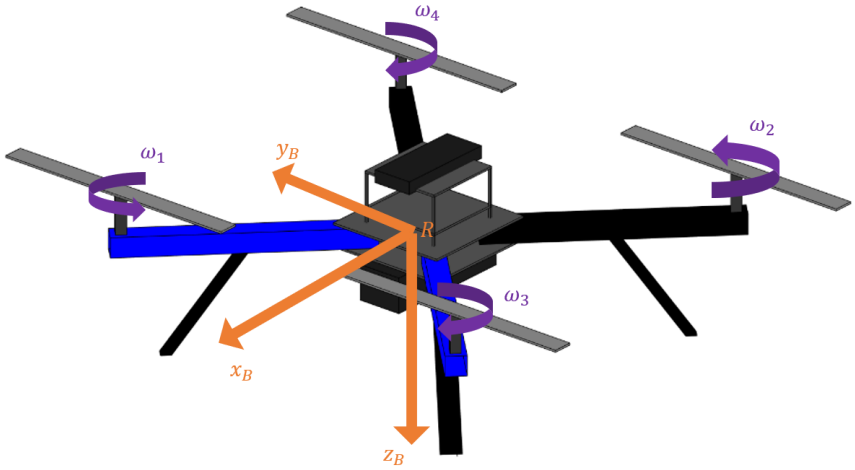

Figure 9.1: *Quadcopter schematic with body-fixed frame (orange; z-axis points downwards) and rotation directions of rotors (purple) used as controls (after [100, p. 32]).*

Take into account that a local navigation frame N for the propagation of the position is used. This frame is fixed at the initial position of the quadcopter and oriented in the direction of the initial orientation with the z axis pointing downwards. In the considered application this is the north direction and thus, the N frame coincides with the O frame (Section A.5). Additionally, the body-fixed frame B is used to denote the translational and rotational properties of the quadcopter. A more detailed introduction on the used frames and transformation matrices can again be found in Appendix A. Furthermore, Table 9.3 introduces the constants used within the quadcopter model. It should be noted that the introduced model is based on a model developed by the author in [100], which is therefore a well tested and researched model.

The forces and moments in the B frame, created by the rotors, can be defined using the following relation [100, p. 38]:

$$\begin{bmatrix} L \\ M \\ N \\ T \end{bmatrix} = \begin{bmatrix} l \cdot k_T & -l \cdot k_T & -l \cdot k_T & l \cdot k_T \\ -l \cdot k_T & l \cdot k_T & -l \cdot k_T & l \cdot k_T \\ k_M & k_M & -k_M & -k_M \\ k_T & k_T & k_T & k_T \end{bmatrix} \cdot \begin{bmatrix} (\theta_{\omega_1} \cdot \omega_1)^2 \\ (\theta_{\omega_2} \cdot \omega_2)^2 \\ (\theta_{\omega_3} \cdot \omega_3)^2 \\ (\theta_{\omega_4} \cdot \omega_4)^2 \end{bmatrix} \tag{9.1}$$

Here, L, M, and N are the roll, pitch, and yaw moment respectively, while T is the thrust force. The arm length is given by $l > 0$, while $k_T < 0$ and $k_M > 0$ are the thrust and moment factor respectively (see Table 9.3). The rotor efficiency factors $\theta_{\omega_{1,\dots,4}} > 0$ are multiplicative factors for the rotor speed command that can be used to simulate e.g., (partial) rotor failures. These are assigned a probability density function (PDF) for the chance-constrained open-loop direct optimal control problem (CC-OCP) in the following.

Additionally, the model uses a constant gravitational model as the quadcopter is not flying a large altitude envelope, but mainly remains at the same altitude. The resulting force in the B frame is given by:

$$\left(\vec{\mathbf{F}}_G\right)_B = \mathbf{M}_{BN} \cdot \begin{bmatrix} 0 & 0 & m_B \cdot g \end{bmatrix}^\mathsf{T} \tag{9.2}$$

Here, g is the gravitational constant and m_B is the (constant) mass of the quadcopter. These constants are defined in Table 9.3.

Finally, the inertia matrix $\mathbf{I}_{BB}$ is required for the propagation of the rotational dynamics. It is defined, using the constants in Table 9.3, as follows:

$$\mathbf{I}_{BB} = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{zx} & 0 & I_{zz} \end{bmatrix} \tag{9.3}$$

Then, the EoMs can be evaluated: To begin with, the EoMs for the position propagation, which is defined on the FFE using the transformation matrix in (A.1) and the velocity states in Table 9.1 [122, p. 41ff.], is looked at:

Table 9.3: *Constants used in the rigid-body quadcopter model for the optimization.*

Description	Symbol	Unit	Value
Reference length	l	[m]	0.1945
Thrust factor	k_T	$[\mathrm{kg} \cdot \mathrm{m}]$	$-1.4019 \cdot 10^{-5}$
Moment factor	k_M	$[\mathrm{kg} \cdot \mathrm{m}^2]$	$1.9023 \cdot 10^{-7}$
Gravitational constant	g	$[\frac{\mathrm{m}}{\mathrm{s}^2}]$	9.80665
Mass	m_B	[kg]	1.488
Moment of inertia around x axis	I_{xx}	$[\mathrm{kg} \cdot \mathrm{m}^2]$	0.0184
Moment of inertia around y axis	I_{yy}	$[\mathrm{kg} \cdot \mathrm{m}^2]$	0.019
Moment of inertia around z axis	I_{zz}	$[\mathrm{kg} \cdot \mathrm{m}^2]$	0.0342
Coupling moment of inertia around xz axis	$I_{xz} = I_{zx}$	$[\mathrm{kg} \cdot \mathrm{m}^2]$	$-1.03 \cdot 10^{-7}$

$$\begin{bmatrix} \dot{x}_N & \dot{y}_N & \dot{z}_N \end{bmatrix}^\mathsf{T} = \mathbf{M}_{NB} \cdot \begin{bmatrix} u_K & v_K & w_K \end{bmatrix}_B^\mathsf{T} \tag{9.4}$$

The EoMs for the orientation angles require the rate states (Table 9.1) and are given as follows [122, p. 19f.]:

$$\begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\Phi) \cdot \tan(\Theta) & \cos(\Phi) \cdot \tan(\Theta) \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \frac{\sin(\Phi)}{\cos(\Theta)} & \frac{\cos(\Phi)}{\cos(\Theta)} \end{bmatrix} \cdot \begin{bmatrix} p_K \\ q_K \\ r_K \end{bmatrix} \tag{9.5}$$

Take into account that EULER angles are used for the attitude propagation in this context instead of the kinematic angles as for the air race in (8.8a)–(8.8c). This is due to the fact that the quadcopter has e.g., no aerodynamic forces and moments acting on it for the low flight speeds looked at in this thesis. Thus, the definition of the attitude with respect to the body attitude is more meaningful and easier to calculate as no intermediate kinematic or aerodynamic angles are required.

Then, the velocity (translational) EoMs are given using the thrust (see (9.1)) and gravitational force (see (9.2)) as follows [122, p. 39ff.]:

$$\begin{bmatrix} \dot{u}_K \\ \dot{v}_K \\ \dot{w}_K \end{bmatrix} = \frac{1}{m_B} \cdot \left\{ \begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \left(\vec{\mathbf{F}}_G \right)_B \right\} - \begin{bmatrix} p_K \\ q_K \\ r_K \end{bmatrix} \times \begin{bmatrix} u_K \\ v_K \\ w_K \end{bmatrix} \tag{9.6}$$

It should be noted that for the quadcopter, the body-fixed velocity is used instead of the absolute velocity and the kinematic angles like for the air race (see (8.6a)–(8.6c)). This is due to the fact that a hover flight condition (i.e., $V_K = 0\frac{\mathrm{m}}{\mathrm{s}}$) should also be covered.

Finally, the last set of EoMs consist of the rate (rotational) dynamics, which are dependent on the moments (see (9.1)) and the inertia matrix (see (9.3)) [122, p. 35ff.]:

$$\begin{bmatrix} \dot{p}_K \\ \dot{q}_K \\ \dot{r}_K \end{bmatrix} = \mathbf{I}_{BB}^{-1} \cdot \left\{ \begin{bmatrix} L \\ M \\ N \end{bmatrix} - \begin{bmatrix} p_K \\ q_K \\ r_K \end{bmatrix} \times \left(\mathbf{I}_{BB} \cdot \begin{bmatrix} p_K \\ q_K \\ r_K \end{bmatrix} \right) \right\} \tag{9.7}$$

Table 9.4: *Initial and final boundary condition ("steady-state hover") for the quadcopter robust optimal control problem.*

State	Unit	IBC	FBC	State	Unit	IBC	FBC
x_N	[m]	0	40	Φ	[rad]	0	0
y_N	[m]	0	0	Θ	[rad]	0	0
z_N	[m]	-10	-10	Ψ	[rad]	0	0
u_K	$[\frac{\mathrm{m}}{\mathrm{s}}]$	0	0	p_K	$[\frac{\mathrm{rad}}{\mathrm{s}}]$	0	0
v_K	$[\frac{\mathrm{m}}{\mathrm{s}}]$	0	0	q_K	$[\frac{\mathrm{rad}}{\mathrm{s}}]$	0	0
w_K	$[\frac{\mathrm{m}}{\mathrm{s}}]$	0	0	r_K	$[\frac{\mathrm{rad}}{\mathrm{s}}]$	0	0

Thus, (9.1)–(9.7), together with (A.1), provide the rigid-body representation of the quadcopter on a FFE. These equations are used within the ROC formulation.

In addition, to the standard EoMs, the model uses circular obstacles. These obstacles can be regarded as "no-fly zones" and thus, should have a very low probability of getting entered even with uncertainty influences. Generally, the obstacles are given by right circular cylinders with infinite height and are defined by an $x_{N,\mathrm{obs}}$ and $y_{N,\mathrm{obs}}$ position as well as a radius r_{obs}, and a desired minimal obstacle distance threshold $d_{\mathrm{obs,lb},*}$ (the maximal separation is not constrained). Then, the distance to an obstacle i is defined by the following relations:

$$\begin{aligned} d_{\mathrm{obs},i} &= \sqrt{\left(x_N - \widetilde{x_N}_{,\mathrm{obs},i}\right)^2 + \left(y_N - \widetilde{y_N}_{,\mathrm{obs},i}\right)^2} - \widetilde{r_{\mathrm{obs}}}_{,i} - d_{\mathrm{obs,lb},*} \\ \text{with} \quad \widetilde{x_N}_{,\mathrm{obs},i} &= x_{N,\mathrm{obs},i} + \theta_{x_{N,\mathrm{obs},i}} \\ \widetilde{y_N}_{,\mathrm{obs},i} &= y_{N,\mathrm{obs},i} + \theta_{y_{N,\mathrm{obs},i}} \\ \widetilde{r_{\mathrm{obs}}}_{,i} &= r_{\mathrm{obs},i} + \theta_{r_{\mathrm{obs},i}} \end{aligned} \tag{9.8}$$

Thus, a constraint can be enforced such that no trajectory intersects an obstacle, i.e., $d_{\mathrm{obs},i} > 0$. Depending on the optimization case, this constraint is enforced using a deterministic path constraint or a CC. To introduce a non-deterministic behavior, the obstacles can be defined uncertain by the additive uncertainties $\theta_{x_{N,\mathrm{obs},i}}$ and $\theta_{y_{N,\mathrm{obs},i}}$ for the x_N and y_N position respectively and by $\theta_{r_{\mathrm{obs},i}}$ for the radius.

Furthermore, the initial boundary condition (IBC) and final boundary condition (FBC) are defined in Table 9.4 for the optimization: Here, the quadcopter is in a hover condition at the IBC and FBC as well as oriented in north direction. Overall, the OCP has one phase (see Subsection 2.1.2.2) and the IBC as well as FBC are equality constraints.

To solve the robust open-loop direct optimal control problem (ROCP), the following general definitions as well as settings for the OCP and the nonlinear program (NLP) solver Interior Point Optimizer (IPOPT) are applied:

- Cost function: $J = t_f$
- Number of discretization steps: $n_\tau = 75$

- Minimal safety distance to obstacle (approximately quadcopter arm length plus half blade): $d_{\text{obs,lb,}*} = d_{\text{obs,1/2,lb}} = 0.25\,\text{m}$
- Position and dimension of obstacle 1: $x_{N,\text{obs,1}} = 10\,\text{m}$, $y_{N,\text{obs,1}} = 1\,\text{m}$, $r_{\text{obs,1}} = 3\,\text{m}$
- Position and dimension of obstacle 2: $x_{N,\text{obs,2}} = 25\,\text{m}$, $y_{N,\text{obs,2}} = -3\,\text{m}$, $r_{\text{obs,2}} = 5\,\text{m}$
- Feasibility tolerance of IPOPT: $\epsilon_{\text{feas}} = 10^{-5}$
- Optimality tolerance of IPOPT: $\epsilon_{\text{opt}} = 10^{-5}$
- Linear solver of IPOPT: `ma97`

Once more, it is reminded that the minimal time cost function generally yields a large control effort (especially including steps). Thus, adding a control cost or actuators in the model might be required for real applications. As noted before, this is omitted in this thesis to make the result analysis straightforward, especially considering the developed robustness modifications.

In the following examples, two uncertainties are used as follows:

$$\begin{aligned} \theta_{r_{\text{obs,2}}} &\sim \mathcal{U}\left(a = -0.1 \cdot r_{\text{obs,2}},\, b = 0.1 \cdot r_{\text{obs,2}}\right) \\ \theta_{\omega_2} &\sim \mathcal{N}\left(\mu = 0.9,\, \sigma = 0.03\right) \end{aligned} \tag{9.9}$$

Thus, a dynamic model uncertainty (efficiency factor of rotor two: θ_{ω_2}) is combined with an external uncertainty (radius of obstacle two: $\theta_{r_{\text{obs,2}}}$) to show the general applicability of the developed CC-OC framework with different types of uncertainties. These are also the most common types of uncertainties in those kinds of applications, as there are first of all normally only incomplete information on the obstacles available a priori (i.e., when planning the trajectory offline). Additionally, the dynamic model is never modeled exactly, i.e., it has unmodeled and simplified dynamics. It should be noted that the combination of uncertainties, and specifically the combination of model as well as external uncertainties, makes it inevitable to use the CC-OC approach as a simple approach, i.e., looking at extremal points, is no longer straightforward. This is due to the nonlinear model as well as the nonlinear interaction between the uncertainties.

Further take into account that the CC-OCP is discretized using the method introduced in Section 5.1 (with both Jacobian and Hessian). Here, both the state as well as the controls are expanded as introduced. This is meaningful as the calculated control expansion can e.g., be used in an online application to update the currently used control strategy. As this can be done by the gPC expansion, i.e., a matrix-vector operation, this procedure is very efficient. This is a further benefit of the CC-OC. It should be noted that this requires the online estimation of the assumed uncertainties, i.e., obstacle parameters and rotor efficiency. This is generally possible in these kinds of application as the exact obstacle

position must be known for safety reason (e.g., vision-based systems can be used), while the model errors are often estimated to check for failures and make updates in the controller (e.g., the gains [43]).

In the following, the CC-OC for the quadcopter OCP (Section 9.1) is looked at: Here, initially the required accuracy of the gPC expansion is derived by mean of influence quantification of the uncertainties in the dynamic model and LHS in Section 9.2. Afterward, the "frequent" event CC-OC (Section 6.1), and specifically the homotopy procedure for the sigmoid CC approximation, is described in Section 9.3. Concluding, Section 9.4 introduces the rare-event CC-OC (Section 6.2) to ensure non-hitting the obstacles with a high probability.

9.2 Polynomial Chaos Expansion Order Determination for Quadcopter Obstacle Avoidance by Comparison to Latin Hypercube Sampling

Once more, the analysis starts with the derivation of the required gPC order to accurately model the uncertainties within the ROCP: A first observation from the OCP definition in Section 9.1 is that the external uncertainty influence (radius of obstacle two) influences the obstacle distance constraint in (9.8) linearly. Secondly, the rotor efficiency is influencing the forces and moments of the hexacopter quadratically (see (9.1)). Due to the fact that the forces and moments are then influencing the translational and rotational EoMs linearly, it can be deduced that the gPC expansion should be at least quadratically.

The validity of the quadratic gPC expansion is verified in the following by comparison to LHS (again, it is reminded that the LHS results will not fully resemble the real statistical moments due to infeasible optimal solutions (Figure 2.4); thus, the results must be discussed with care). Once more, the LHS results are obtained by calculating the exact optimal trajectories at random samples. Here, the calculation of LHS result took around $1,600\,\mathrm{s}$ on a personal computer[1], while the optimization time for the gPC expansion (sequential evaluation) ranged from $34.5\,\mathrm{s}$ ($d_i = 2$) to $212\,\mathrm{s}$ ($d_i = 5$). Thus, the required computational time is reduced by at least $87\,\%$. Once more, this shows the necessity to apply the gPC method to calculate the ROC results. Furthermore, the number of required evaluation points (i.e., the LHS samples and stochastic collocation (SC) nodes) is significantly reduced by the gPC expansion which initially enables the used of the proposed transcription method (Chapter 5).

[1] Architecture: x64 Intel® Core™ i7-6700K CPU @4.00 GHz, 16.0 GB RAM

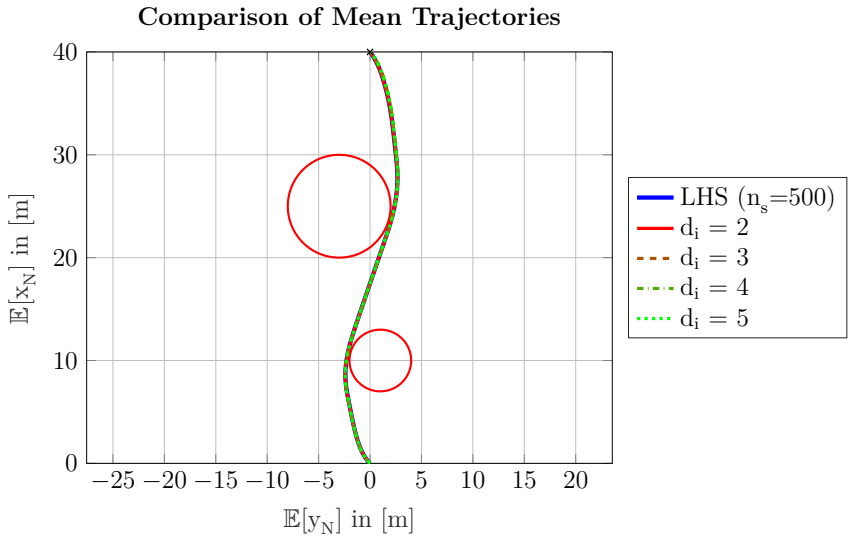

Figure 9.2: *Comparison of mean trajectories by Latin-hypercube sampling and generalized polynomial chaos around obstacles.*

At first, the mean trajectory around the obstacles is shown in Figure 9.2 (the obstacle uncertainties is plotted for its mean value). Here, the LHS results is depicted in solid blue and is calculated from $n_s = 500$ successful optimizations. The gPC expansion is shown for orders of $d_i = 2, \ldots, 5$ (with colors from red to green). It can be seen that the mean results are on top of each other and therefore, well-matching as expected.

To get a better overview on the accuracy of the statistical moment approximation for the gPC schemes, Figure 9.3 shows the positional standard deviation from LHS, including confidence intervals (CIs) (lower bound: solid blue; upper bound: solid cyan), compared to the different gPC expansion results. Overall, the gPC expansions of all orders yield good results for the y_N and z_N position. There are differences in the x_N position, which mainly occur in between the obstacles. As described in the MCA introduction (Subsection 2.2.1), these are partially related to the infeasible results encountered during the LHS solution process. Additionally, the CIs are calculated based on the assumption that the considered quantity is GAUSSIAN-distributed (Subsection 2.2.3), which is normally not the exactly the case. Thus, the matching is still acceptable especially considering that the trend is very similar and the overall magnitude of the values is small.

After having looked at the positional accuracy, Figure 9.4 introduces the error bars of the kinematic states (i.e., roll, pitch, and yaw). Here, the mean values are depicted by the solid lines and the standard deviation around it is shown by the bars in both axes directions. Thus, both the standard deviation in the state as well as the time direction is visualized. Once more, it can be seen that the mean value matching is very good. Additionally, the error bars are also matching fairly well over time. It is further interesting to see that the standard deviation of the yaw angle (Ψ) is comparably large in the time horizon 2 s–5 s (which is the time between the obstacles). This is natural as, depending on

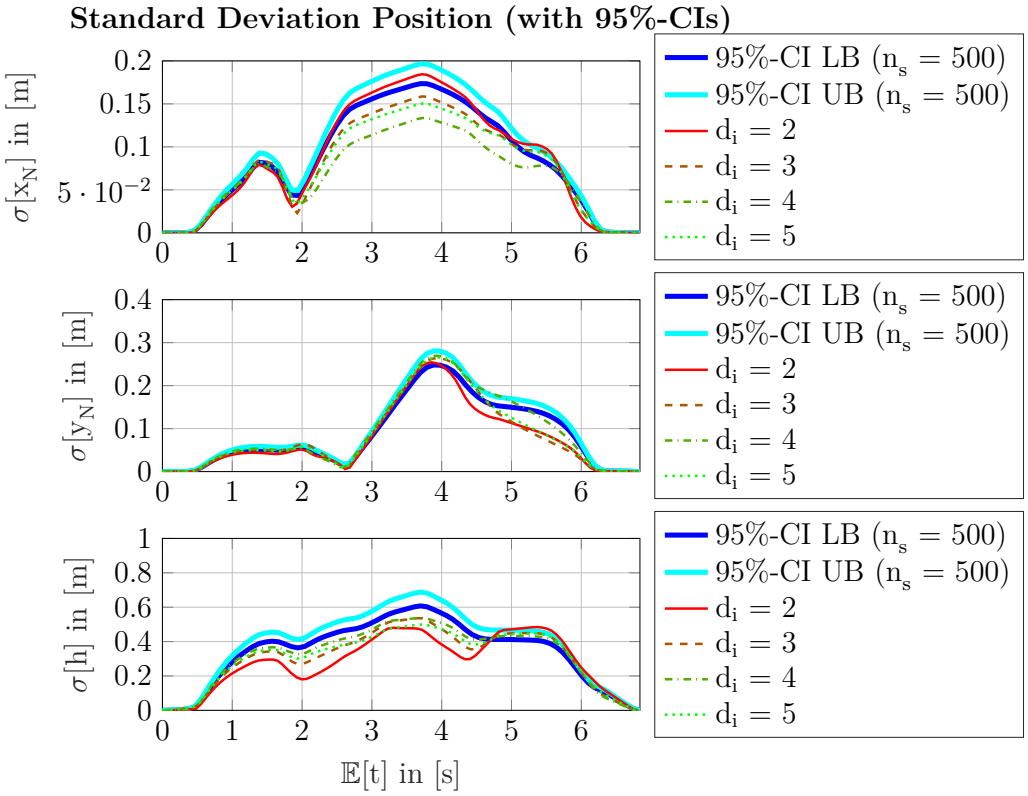

Figure 9.3: *Comparison of position standard deviation between Latin-hypercube sampling (95 %-confidence interval) and different polynomial chaos expansion orders.*

the current size of the second obstacle, the steepness of the turn must be adapted. Other than that, it can be seen that the quadcopter initially pitches (Θ) forward to accelerate and at the end pitches backward to decelerate and achieve the hover position at the FBC.

Additionally, the pitching motion controls the velocity of the quadcopter, which is shown by the error bars in Figure 9.5: Here, the behavior of the sideward velocity v_K resembles the behavior of the yaw, i.e., a significant standard deviation. Furthermore, it can be seen that the quadcopter accelerates to the maximal allowed forward velocity u_K, while the downward velocity w_K shows the influence of both pitch as well as roll angle (Figure 9.4). Generally, all gPC orders, once more, recover the reference statistics from LHS very good.

After this general look into the accuracy of the states, the next paragraphs deal with the accuracy of the obstacle distance, which is very important in the CC-OC framework as hitting any of the obstacles should be avoided with a high probability. Here, Figure 9.6 shows the development of the mean and standard deviation of the distance to the first obstacle, while Figure 9.7 does the same for the second obstacle. The figures show the lower (solid blue) and upper bounds (solid cyan) of the 95 %-CI obtained from the LHS with $n_s = 500$ successful optimization samples. Additionally, the gPC expansion orders of $d_i = 2, \ldots, 5$ are once more shown to compare the accuracy. The minimal required obstacle separation is depicted by the dashed, horizontal magenta line. As already mentioned

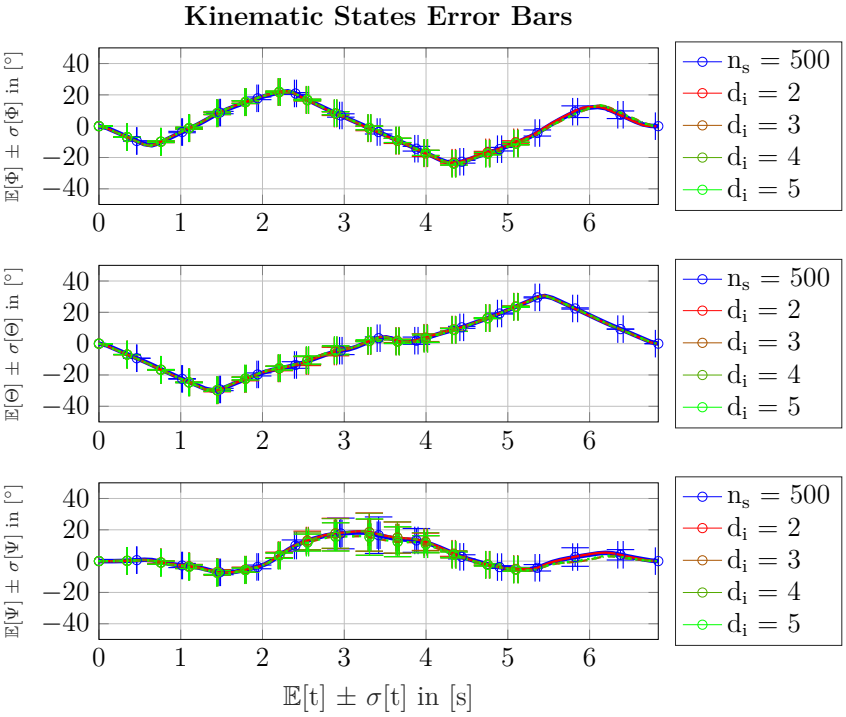

Figure 9.4: *Error bars of kinematic states for Latin-hypercube sampling and different generalized polynomial chaos orders.*

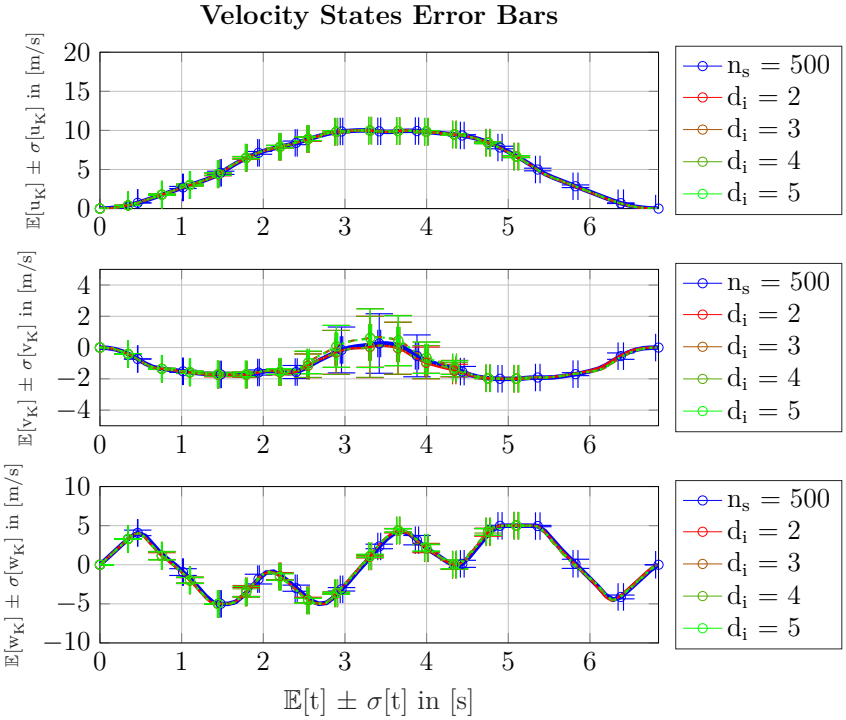

Figure 9.5: *Error bars of velocity states for Latin-hypercube sampling and different generalized polynomial chaos orders.*

before, the results of the LHS have to be considered regarding the fact that some of the optimization yielded infeasible results and thus, the uncertain parameter PDF is not accurately covered by the LHS (see Figure 2.4). Additionally, the CIs are calculated based on the assumption that the considered quantity is GAUSSIAN-distributed, which is generally not the case for the obstacle distances.

Still, Figure 9.6 and Figure 9.7 show a good approximation quality of the mean obstacle distance by all gPC expansion orders compared to the 95 %-LHS CI. It should be noted that the magenta, dashed horizontal line visualizes the desired minimal distance to the obstacle, which is naturally fulfilled for all plotted cases. The area close to the minimal separation is also shown by the zoomed-in axes within Figures 9.6 and 9.7. It is clearly seen that all trajectories reach the minimal distance to the obstacle, which is still feasible. This is natural as it results in a faster trajectory due to the smaller evasive maneuver.

Furthermore, it can be seen in Figure 9.6 that the standard deviation approximation for the first obstacle by the gPC expansion is generally too small. As the standard deviation to the obstacle is overall of a small magnitude (because the obstacle itself is not uncertain), the introduced errors can also be considered small. The best approximation is given by the third order, i.e., quadratic, gPC expansion, which was already deduced in the first paragraph of this section.

A better approximation of the standard deviation by the gPC expansion is given for the second obstacle (Figure 9.7). Here, only the second order, i.e., linear, approximation is outside the CI for a longer period, while especially the third order approximation is very good. Additionally, a larger magnitude of the standard deviation can be seen because the obstacle radius itself is also uncertain. Thus, approximating this distance accurately is of paramount importance. It is important to note that the standard deviation of the distance to the second obstacle is non-zero even at the IBC and FBC: This is due to the fact that obstacle two is uncertain and thus, the distance is varying even at the fixed hover condition. The seen standard deviation is then exactly the standard deviation of the UNIFORM PDF defined for obstacle two in (9.9).

Concluding the accuracy estimation section, Figure 9.8 show the failure probability of hitting the obstacles. This probability estimation is calculated as proposed in the CC-OC Chapter 6 by (6.2): Here, the expansion coefficients, obtained from the generalized polynomial chaos-stochastic collocation framework (gPC-SC) solution for the different gPC expansion orders, are sampled with random samples ($n_s = 5000$) of the uncertainties. It is imminent from Figure 9.8 that close to the two obstacles almost all samples fail the minimum safety distance requirement. Take into account that this is due to the fact that the gPC-SC OCP only enforces the distance constraint at the SC nodes (i.e., on four to twenty-five trajectories depending on the gPC expansion order) with only the minimal required separation distance. Thus, even small parameter deviations, made in an MCA,

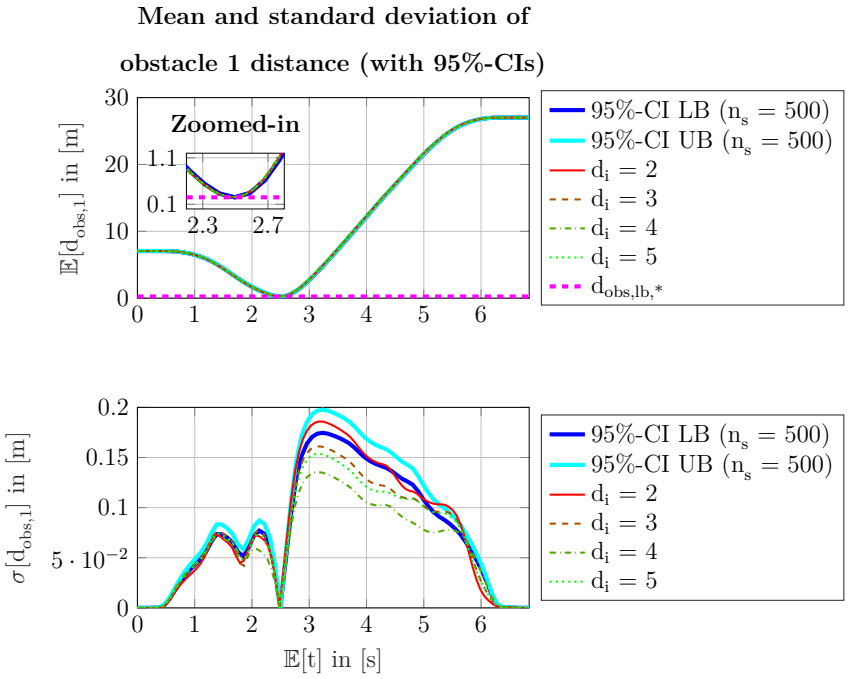

Figure 9.6: *Mean distance and standard deviation to obstacle one calculated by Latin-hypercube sampling (95 %-confidence interval) and different polynomial chaos expansion orders.*

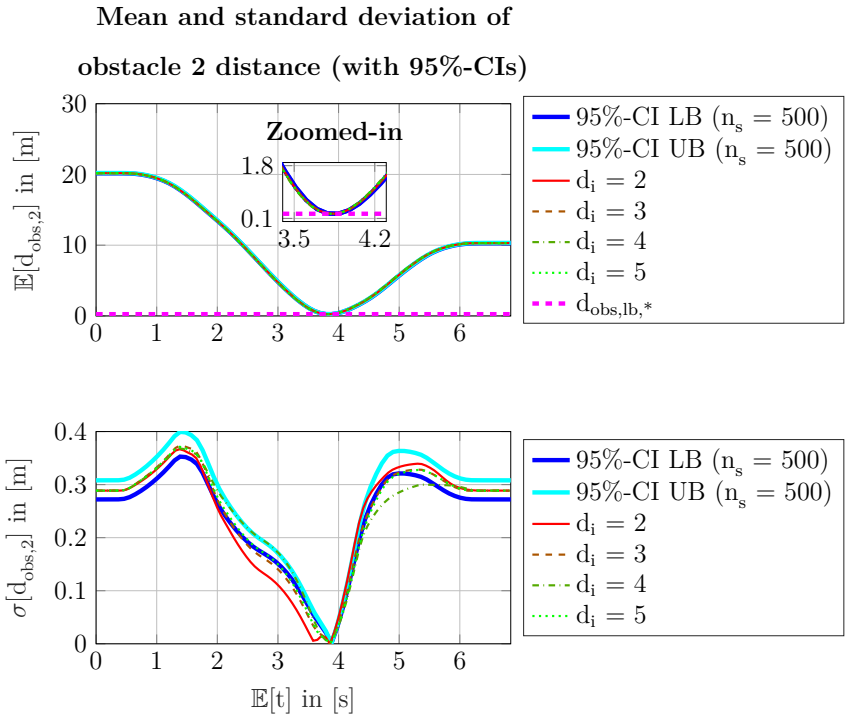

Figure 9.7: *Mean distance and standard deviation to obstacle two calculated by Latin-hypercube sampling (95 %-confidence interval) and different polynomial chaos expansion orders.*

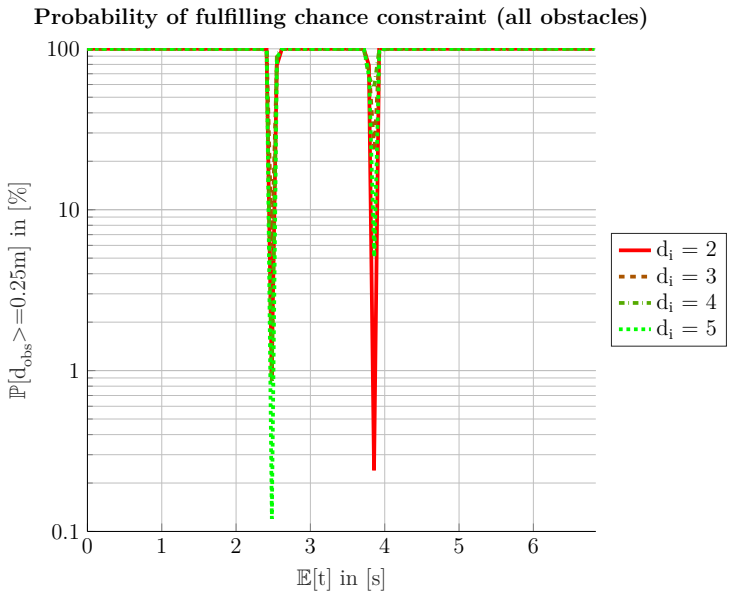

Figure 9.8: *Probability of fulfilling safety distance to obstacles based on random sampling of generalized polynomial chaos expansion.*

normally yield to a violation of the safety distance. Additionally, the NLP solver is allowed to fail the distance constraint by its tolerances. Consequently, the probability of fulfilling the safety distance at the obstacles is close to zero.

Summarizing, a robustification of the OCP by CCs is required in this example to get a safe and robust trajectory around the obstacles. The required gPC expansion order for the resulting CC-OCP can be determined as $d_i = 3$, because Figure 9.8 once more shows that the third order expansion gives similar results like the higher order expansion, while not being as conservative as the second order expansion. Thus, considering the results shown in this section as a whole, the third order expansion is sufficiently accurate for the CC-OCP with obstacle avoidance.

9.3 Polynomial Chaos Sampling-based Obstacle Avoidance for Frequent Event by Monte Carlo Chance Constraint

This subsection introduces initial results for the CC-OCP of the quadcopter with obstacle avoidance. Here, the obstacle avoidance is modeled by CCs. For this purpose, the gPC collocation transcription (including the analytic Hessian provided by FSD optimal control tool for MATLAB® (FALCON.m)) with CCs in (6.1) is used. Thus, the implemented procedure is based on the framework presented in Section 6.1 and related to Contribution 4 of this thesis. The section should mainly serve as an overview on the capabilities of

the proposed CC-OC approach and introduces the developed CC approximation including the homotopy procedure. This is extended in Section 9.4 to the rare-event failure probability application.

The ROCP formulation is then given as follows (for the simplicity of writing only the i-th discretized time step is written):

$$
\begin{aligned}
&\min_{\hat{\mathbf{z}}} \quad J = \hat{t_f}^{(0)} \\
&\text{s.t.} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \hat{\mathbf{z}}_{\text{lb}} \leq \hat{\mathbf{z}} \leq \hat{\mathbf{z}}_{\text{ub}}, \\
&\qquad \mathbf{c}(\hat{\mathbf{z}};\boldsymbol{\theta},\mathbf{q}) = \begin{bmatrix} d_{\text{obs,lb},*} - d_{\text{obs},1,i}^{(j)} \\ d_{\text{obs,lb},*} - d_{\text{obs},2,i}^{(j)} \\ \mathbf{x}_{i,\text{lb}} - \mathbf{x}_i^{(j)} = \mathbf{x}_{i,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{x}}_i^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\ \mathbf{x}_i^{(j)} - \mathbf{x}_{i,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{x}}_i^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{x}_{i,\text{ub}} \\ \mathbf{u}_{i,\text{lb}} - \mathbf{u}_i^{(j)} = \mathbf{u}_{i,\text{lb}} - \sum_{m=0}^{M-1} \hat{\mathbf{u}}_i^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \\ \mathbf{u}_i^{(j)} - \mathbf{u}_{i,\text{ub}} = \sum_{m=0}^{M-1} \hat{\mathbf{u}}_i^{(m)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) - \mathbf{u}_{i,\text{ub}} \end{bmatrix} \leq \mathbf{0}, \\
&\qquad \boldsymbol{\psi}(\hat{\mathbf{z}};\boldsymbol{\theta},\mathbf{q}) = \begin{bmatrix} \hat{\mathbf{x}}_{i+1}^{(m)} - \hat{\mathbf{x}}_i^{(m)} - \frac{h_\tau}{2}\left(\sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_{i+1}^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right. \\ \left. + \sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_i^{(j)} \Phi^{(m)}\left(\boldsymbol{\theta}^{(j)}\right) \alpha^{(j)} \right) \end{bmatrix} = \mathbf{0}, \\
&\qquad \begin{bmatrix} \mathbb{P}\left[\left(d_{\text{obs},1,i}\left(\hat{\mathbf{z}};\boldsymbol{\theta}\right) = \sum_{m=0}^{M-1} d_{\text{obs},1,i}^{(m)}\left(\hat{\mathbf{z}}\right) \Phi^{(m)}\left(\boldsymbol{\theta}\right) \right) \geq d_{\text{obs,lb},*} \right] \\ \mathbb{P}\left[\left(d_{\text{obs},2,i}\left(\hat{\mathbf{z}};\boldsymbol{\theta}\right) = \sum_{m=0}^{M-1} d_{\text{obs},2,i}^{(m)}\left(\hat{\mathbf{z}}\right) \Phi^{(m)}\left(\boldsymbol{\theta}\right) \right) \geq d_{\text{obs,lb},*} \right] \end{bmatrix} \geq \boldsymbol{\xi}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, n_\tau
\end{aligned}
\tag{9.10}
$$

Thus, the ROCP is designed to optimize the mean final time and enforces deterministic constraints (Chapter 5) on e.g., the physical states as well as controls. In addition, the expansion states are propagated using the trapezoidal collocation defect (see (5.3)). Furthermore, CCs enforce that the robust trajectory must satisfy the minimum distance to the obstacle with a probability level of $\boldsymbol{\xi}$. In this example, the fulfillment probability is chosen to be $\xi = \xi_i = \boldsymbol{\xi} = 0.99$ ($= 99\%$), i.e., the same probability level is enforced at each discretized time step. Thus, only at maximum $1\,\%$ of the trajectories are allowed to fail, i.e., hit, either of the obstacles at any point of the trajectory. Here, the number of samples to evaluate the CC within the CC-OCP is chosen to be $n_s = 5000$ (as this has shown to be sufficient). Additionally, the boundary level value of the sigmoid is chosen to be $\text{BL}_s = 1 - \frac{1-\xi}{1000}$, i.e., three magnitudes smaller than the desired probability, which is sufficient for the sigmoid to approximate the real sharp (logical) CC.

In the following, the results for the ROCP in (9.10) are discussed. Therefore, Table 9.5 introduces the three MCA-based CC-OC cases studied in this section and the reference gPC results. The three exemplary MCA-based cases specifically look at different values for the sigmoid scaling value, i.e., the homotopy procedure in Algorithm 6.1. It should be noted that these values have been chosen according to the results in Subsection 9.2, by looking for a sufficiently large scaling factor that can accurately reproduce the CC probability in Figure 9.8 for the reference gPC results. Here, orders of $\mathcal{O}(10^7)$ have proven to be viable. As stated in Subsection 9.2 a third order gPC expansion is sufficient for the approximation and thus, used for the optimization.

Furthermore, Table 9.5 shows the resulting sigmoid offsets, which are in the order $\mathcal{O}(10^{-7})$ and thus, reasonably small especially considering the chosen feasibility tolerance of $\epsilon_{\text{feas}} = 10^{-5}$ and optimality tolerance of $\epsilon_{\text{opt}} = 10^{-5}$.

Finally, Table 9.5 also introduces the optimal results for the mean final time as well as the reduction in the optimality compared to the reference case. First of all, it is clear that the three CC-OC cases give the same optimal mean final time and thus, the approximation quality of the three cases is similar. Compared, to the mean time of the reference gPC case, a reduction in the optimal final time of around 1 % is necessary to achieve a robustness improvement and especially fulfill the CC in (9.10). This is the well-known and already mentioned trade-off between optimality and robustness. On the other hand, the robustness improvement, which is here illustrated by the minimal mean separation to any obstacle along the trajectory, is approximately 55 % for the robust results. Compared to the optimality reduction, which is still comparably small, the robustness improvement is thus significant and will be discussed in the following results in more detail.

Regarding the trade-off between optimality and robustness, Table 9.5 also illustrates very well that the used CC-OC algorithm does reduce the required conservatism that is normally present in robust design/optimization: For instance, robust design method could suggest to at least avoid 99 % of the UNIFORM obstacle uncertainty PDF. This would suggest to have a minimum separation of at least $d_{\text{obs,min,req}} = 0.25\,\text{m} + 0.49\,\text{m} = 0.74\,\text{m}$ (the value can be calculated from the inverse of the UNIFORM PDF). It is clear that this value is almost twice as large as the value in Table 9.5, which suggests a more optimal trajectory by the CC-OC. This improved optimality is first of all based on the fact that the CC-OC considers the interactions between the uncertainties (e.g., there are not only extremely large obstacles but also smaller ones), which might be helpful in fulfilling the CC. Additionally, the CC-OC expands the control history, which allows an adaptation of the control command with respect to the uncertainty. This allows to be more optimal rather than being very conservative.

Before looking at the results, Table 9.6 gives an overview on the tensor grid SC nodes used for the calculation of the gPC expansion coefficient. It is seen that there are nine nodes, which combine the three unique nodes from each of the uncertainties. As a third order expansion is used, the PDF mean value is always a part of the gPC expansion

Table 9.5: *Overview of Monte Carlo-based chance-constrained optimal control results with comparison to reference polynomial chaos results.*

Case	Sigmoid Offset	Mean Optimal Time	Relative Optimality Reduction	Minimal Mean Separation	Relative Robustness Improvement
$a_s = 5 \cdot 10^7$ $(= 5\mathrm{e}+7)$	$b_s \approx 2.3026 \cdot 10^{-7}$	$\mathbb{E}[t_f] \approx 6.8865\,\mathrm{s}$	$\approx 0.9114\,\%$	$d_{\mathrm{obs,min}} \approx 0.3875\,\mathrm{m}$	$\approx 55\,\%$
$a_s = 7 \cdot 10^7$ $(= 7\mathrm{e}+7)$	$b_s \approx 1.6447 \cdot 10^{-7}$	$\mathbb{E}[t_f] \approx 6.8865\,\mathrm{s}$	$\approx 0.9114\,\%$	$d_{\mathrm{obs,min}} \approx 0.3875\,\mathrm{m}$	$\approx 55\,\%$
$a_s = 9 \cdot 10^7$ $(= 9\mathrm{e}+7)$	$b_s \approx 1.2792 \cdot 10^{-7}$	$\mathbb{E}[t_f] \approx 6.8865\,\mathrm{s}$	$\approx 0.9114\,\%$	$d_{\mathrm{obs,min}} \approx 0.3875\,\mathrm{m}$	$\approx 55\,\%$
Reference: $d_i = 3$	–	$\mathbb{E}[t_f] \approx 6.8243\,\mathrm{s}$	$= 0\,\%$	$d_{\mathrm{obs,min}} \approx 0.2500\,\mathrm{m}$	$= 0\,\%$

Table 9.6: *Definition of random parameters for the stochastic collocation trajectories used in Monte Carlo-based chance-constrained optimal control.*

SC Node	Obstacle Uncertainty	Rotor Uncertainty
$j = 1$	$\theta^{(1)}_{r_{\mathrm{obs},2}} \approx -0.3873$	$\theta^{(1)}_{\omega_2} \approx 0.8480$
$j = 2$	$\theta^{(2)}_{r_{\mathrm{obs},2}} \approx -0.3873$	$\theta^{(2)}_{\omega_2} = 0.9$
$j = 3$	$\theta^{(3)}_{r_{\mathrm{obs},2}} \approx -0.3873$	$\theta^{(3)}_{\omega_2} \approx 0.9520$
$j = 4$	$\theta^{(4)}_{r_{\mathrm{obs},2}} = 0$	$\theta^{(4)}_{\omega_2} \approx 0.8480$
$j = 5$	$\theta^{(5)}_{r_{\mathrm{obs},2}} = 0$	$\theta^{(5)}_{\omega_2} = 0.9$
$j = 6$	$\theta^{(6)}_{r_{\mathrm{obs},2}} = 0$	$\theta^{(6)}_{\omega_2} \approx 0.9520$
$j = 7$	$\theta^{(7)}_{r_{\mathrm{obs},2}} \approx 0.3873$	$\theta^{(7)}_{\omega_2} \approx 0.8480$
$j = 8$	$\theta^{(8)}_{r_{\mathrm{obs},2}} \approx 0.3873$	$\theta^{(8)}_{\omega_2} = 0.9$
$j = 9$	$\theta^{(9)}_{r_{\mathrm{obs},2}} \approx 0.3873$	$\theta^{(9)}_{\omega_2} \approx 0.9520$

calculation. Due to the fact that the rotor uncertainty is defined to be a Gaussian PDF, it is also seen that the SC nodes are spread wider compared to the obstacle uncertainty, which is distributed as a Uniform PDF.

The trajectories at these SC nodes (see Table 9.6), are also depicted in Figure 9.9 for the exemplary homotopy scaling factor $a_s = 9 \cdot 10^7$ (i.e., the largest scaling): Here, it is clear that the trajectories are very similar until passing the first obstacle. After the first obstacle, the trajectories spread due to the fact that the second obstacle has different radii depending on the uncertainty (this is visualized by the three circles), which either allows faster or enforces slower trajectories. The introduced spread by the second obstacle continues to tighten until the final point of the trajectory as here the same FBC must be fulfilled by all SC trajectories. It can additionally be observed in Figure 9.9 that there is a safety distance to the obstacles, which is seemingly larger than for the non-robust, optimal result (e.g., Figure 9.2) and was also already seen in Table 9.5.

This increased distance is also visible looking at the mean optimal trajectories in Figure 9.10. The figure shows the optimal trajectories for the three homotopy steps of the CC-OCP (colors from green to blue) as well as the reference case (solid black). Here, it can be seen that the non-robust trajectory gets closer to the obstacle, which makes the trajectory faster (as the covered distance is smaller), but also decreases the robustness. It should be noted that these small distances also illustrate once more that only a small optimality reduction (Table 9.5) is required to increase the robustness.

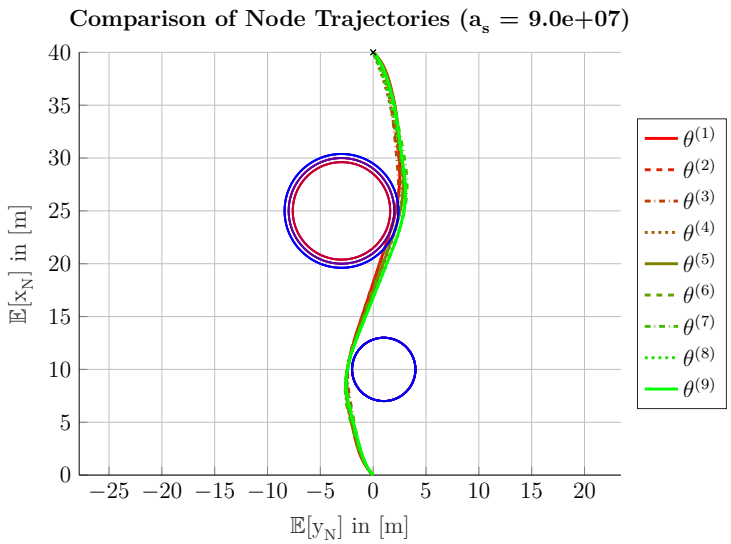

Figure 9.9: *Comparison of Monte Carlo-based chance-constrained, robust, stochastic collocation node quadcopter trajectories for exemplary sigmoid scaling factors.*

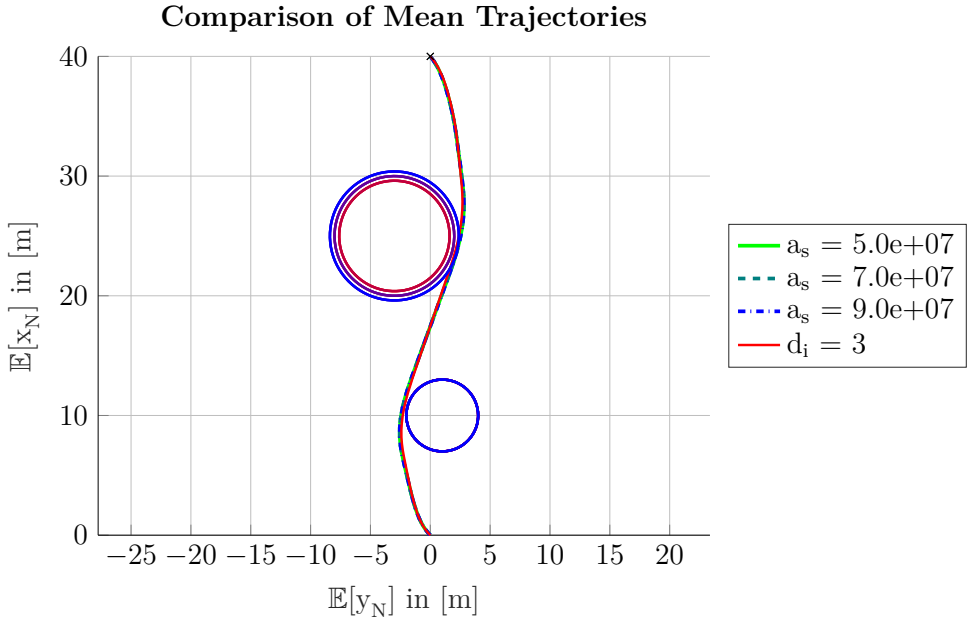

Figure 9.10: *Comparison of Monte Carlo-based chance-constrained, robust, mean quadcopter trajectories for different sigmoid scaling factors.*

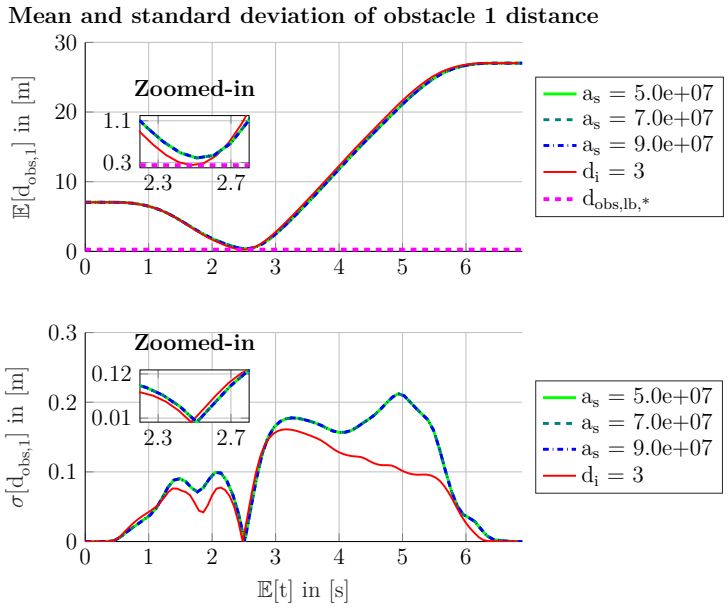

Figure 9.11: *Mean distance and standard deviation of quadcopter distance to obstacle one for Monte Carlo-based chance-constrained, robust optimal control.*

To illustrate the changes in the obstacle separation better, Figure 9.11 and Figure 9.12 show the mean and standard deviation of the distance to obstacle one and obstacle two respectively (once more, the minimum separation limit is depicted by dashed, horizontal magenta line). As already noted the mean values are fairly similar, but the zoomed-in portion of the mean value plots shows that the reference gPC-SC trajectory reaches the minimum separation limit (dashed, horizontal, magenta line), while the CC-OCP results all keep an additional safety distance. Additionally, the minimal separation is reached at a later time instant, which corresponds to the increased optimal time. Furthermore, Figure 9.11 and Figure 9.12 show that the standard deviation to the obstacles is increased as well. This is first of all based on the fact that evading the obstacles requires larger distances to the obstacle, which are generally nonlinear due to the uncertainty in the rotor speed. Thus, the trajectories and distances spread up. Additionally, a reason for the increased spread comes with the fact that the mean final time should still be minimized. This results in the trajectories to spread even further, because some obstacle/uncertainty combinations are more beneficial to this goal than others.

Concluding the results of the MCA-based CC-OC, Figure 9.13 depicts the probability of violating the obstacle distance constraint for the three homotopy factors. Once more, this probability is calculated in a post-processing step using $n_s = 5000$ random samples (different for each homotopy factor) and the exact, sharp CC (i.e., a logical decision). It is clear that the minimum probability level (dashed, horizontal, magenta line in Figure 9.13) is fulfilled for all homotopy factors. There would even be some margin to reduce robustness and improve optimality. It should be noted that this trade-off is, in this context, also strongly depending on the tolerances of the NLP solver. Thus, this trade-off can partly

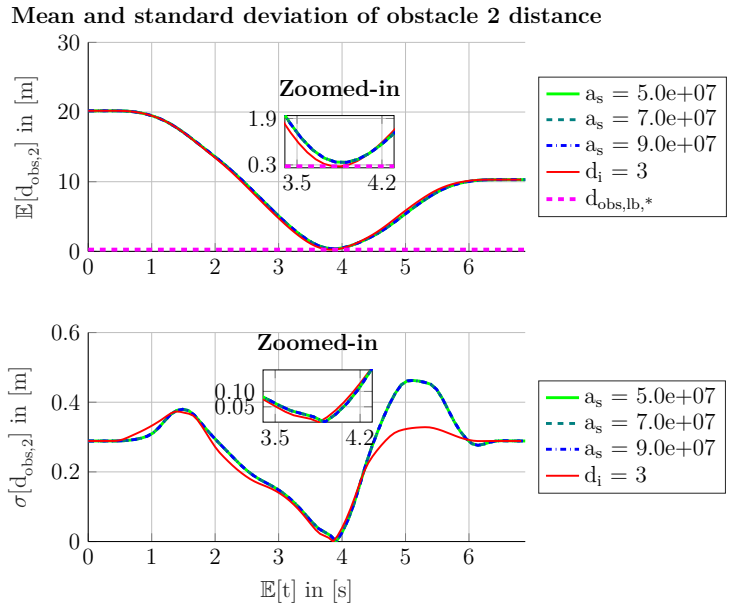

Figure 9.12: *Mean distance and standard deviation of quadcopter distance to obstacle two for Monte Carlo-based chance-constrained, robust optimal control.*

be made by adapting their tolerances. Still, it is seen that the CC-OC does not remove the failure probability completely, as it would also be expected to remain as optimal as possible. It should be further noted that a complete reduction of the failure probability would also not possible as the rotor efficiency uncertainty is distributed by a GAUSSIAN PDF and thus, also infinite values are possible, which will generally lead to failure and infeasibilities inherently. Further take into account that, once more, the homotopy factors also produce similar results. Thus, all values appropriately approximate the CC and can be used with the rare-event CC-OCP discussed in the next section.

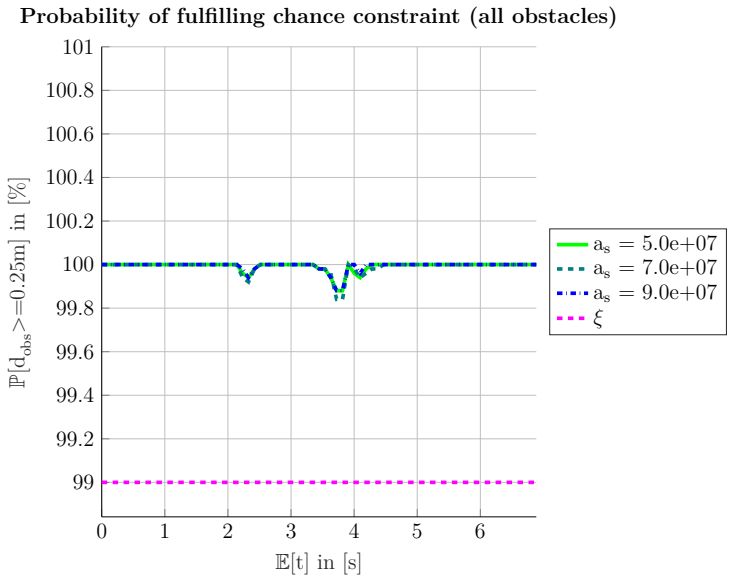

Figure 9.13: *Probability of fulfilling quadcopter distance constraint to both obstacles for Monte Carlo-based chance-constrained, robust optimal control.*

9.4 Subset Simulation based Rare-Event Obstacle Avoidance for Quadcopter

This section introduces the CC-OCP with SubSim of the quadcopter to avoid violating the minimal safety distance to the obstacles with a high probability. Thus, results for Contribution 5 of this thesis are shown. Specifically, this framework is a major development to the state of the art provided by this thesis as it connects open-loop direct optimal control (OC) with rare-event CC evaluation by SubSim in a generic approach, which enhances available ROC methods. Once more, it is reminded here that the efficient evaluation in the SubSim during the NLP is possible due to the efficient sampling provided by the gPC expansion.

Therefore, the ROCP in (9.10) is extended by assigning a large fulfillment probability for $\boldsymbol{\xi}$, i.e., only a small amount of failures is allowed. In this section, this fulfillment probability is based on the proposed European Aviation Safety Agency (EASA) specifications for vertical take-off and landing (VTOL) vehicles [40], for which the quadcopter in this application is one example for. Here, the EASA is defining maximal failure probabilities based on the number of passengers and the failure condition [40, p. 26]. As the used quadcopter in this example carries no passengers, the least restrictive failure probabilities apply. These are defined per flight hour and reach from 10^{-3} for minor failures until 10^{-6} for catastrophic failures. Take into account that the CC-OCP does not optimize the failure probability per flight hour but per trajectory time instant. Still, reaching a high safety level is beneficial both per flight hour as well as per trajectory. Thus, the proposed approach enables the possibility to specify (certification) requirements within the ROC, which is a development compared to the state of the art.

Then, the MCA-based CC-OCP in (9.10) can be extended to the form used for the SubSim-based CC-OC as follows:

$$
\begin{aligned}
\min_{\hat{\mathbf{z}}} \quad & J = \hat{t_f}^{(0)} \\
\text{s.t.} \quad & \hat{\mathbf{z}}_{\text{lb}} \leq \hat{\mathbf{z}} \leq \hat{\mathbf{z}}_{\text{ub}}, \\
& \mathbf{c}(\hat{\mathbf{z}}; \boldsymbol{\theta}, \mathbf{q}) = \begin{bmatrix} d_{\text{obs,lb},*} - d_{\text{obs},1,i}^{(j)} \\ d_{\text{obs,lb},*} - d_{\text{obs},2,i}^{(j)} \\ \mathbf{x}_{i,\text{lb}} - \mathbf{x}_i^{(j)} \\ \mathbf{x}_i^{(j)} - \mathbf{x}_{i,\text{ub}} \\ \mathbf{u}_{i,\text{lb}} - \mathbf{u}_i^{(j)} \\ \mathbf{u}_i^{(j)} - \mathbf{u}_{i,\text{ub}} \end{bmatrix} \leq \mathbf{0}, \\
& \boldsymbol{\psi}(\hat{\mathbf{z}}; \boldsymbol{\theta}, \mathbf{q}) = \begin{bmatrix} \hat{\mathbf{x}}_{i+1}^{(m)} - \hat{\mathbf{x}}_i^{(m)} - \frac{h_\tau}{2} \left(\sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_{i+1}^{(j)} \Phi^{(m)} \left(\boldsymbol{\theta}^{(j)} \right) \alpha^{(j)} \right. \\ \left. + \sum_{j=1}^{Q} {t_f}^{(j)} \dot{\mathbf{x}}_i^{(j)} \Phi^{(m)} \left(\boldsymbol{\theta}^{(j)} \right) \alpha^{(j)} \right) \end{bmatrix} = \mathbf{0}, \\
& \begin{bmatrix} \mathbb{E}\left[\mathbb{P}\left[d_{\text{obs},1,i}\left(\hat{\mathbf{z}}; \boldsymbol{\theta}\right) \geq d_{\text{obs,lb},*}\right]\right] + w \cdot \sigma\left[\mathbb{P}\left[d_{\text{obs},1,i}\left(\hat{\mathbf{z}}; \boldsymbol{\theta}\right) \geq d_{\text{obs,lb},*}\right]\right] \\ \mathbb{E}\left[\mathbb{P}\left[d_{\text{obs},2,i}\left(\hat{\mathbf{z}}; \boldsymbol{\theta}\right) \geq d_{\text{obs,lb},*}\right]\right] + w \cdot \sigma\left[\mathbb{P}\left[d_{\text{obs},2,i}\left(\hat{\mathbf{z}}; \boldsymbol{\theta}\right) \geq d_{\text{obs,lb},*}\right]\right] \end{bmatrix} \geq \boldsymbol{\xi}, \\
& \qquad\qquad i = 1, \ldots, n_\tau
\end{aligned}
\tag{9.11}
$$

Here, the basic structure is not changed while the definition of the CC is adapted: First of all, the CC is now evaluated using the SubSim-based CC-OC framework (Section 6.2). Additionally, the possibility of SubSim to provide both the mean as well as the standard deviation of the failure/fulfillment probability is used and therefore, the CC itself can be robustified as well. This robustification is adapted by the weighting factor w and can be generally viewed as a Pareto problem (Subsection 2.4.1). In the context of the SubSim, this formulation also describes a control of coefficient of variation (CoV) (see (2.45)) and thus, the certainty in the CC-OC results.

It should be noted that the largest sigmoid scaling from Section 9.3, i.e., $a_s = 9 \cdot 10^7$, is used in the following for the SubSim CC evaluation. Furthermore, the SubSim conditional probability level is defined to be $p_0 = 0.1$ and there are a total of $n_s = 5000$ samples used. These values have proven to give reasonable results and convergence of the SubSim. It should be further noted that the proposal PDF (Step 0 in Algorithm 2.2) for the Gaussian uncertainty, i.e., the second rotor efficiency, is defined to be a Gaussian PDF given by: $\rho^*_{\Theta_{r_{\text{obs},2}}}\left(\tilde{\theta}_{r_{\text{obs},2}} | \theta_{r_{\text{obs},2}}\right) = \mathcal{N}\left(\mu = \theta_{r_{\text{obs},2}},\, \sigma = 1\right)$. On the other hand, the proposal PDF for the Uniform uncertainty, i.e., the second obstacle, is a Uniform PDF defined as follows: $\rho^*_{\Theta_{\omega_2}}\left(\tilde{\theta}_{\omega_2} | \theta_{\omega_2}\right) = \mathcal{U}\left(a = \theta_{\omega_2} - 1,\, b = \theta_{\omega_2} + 1\right)$. Take into account that the created samples from the Uniform proposal PDF are enforced to be in the interval $[-1; 1]$ to ensure that the gPC evaluation is always in the standardized domain (see Table 2.2). Once more, these choices have proven to be viable in application.

With the inputs to Algorithm 6.2 defined, the CC-OC results can be calculated and Table 9.7 summarizes the results obtained for the SubSim-based CC-OC with different fulfillment probability levels: $\xi = 99.9\,\%$ (with four weight cases for the standard deviation: $w = 0$, 0.25, 0.5, 0.75), $\xi = 99.99\,\%$ (without the standard deviation influence in the constraint function), and $\xi = 99.999\,\%$ (without the standard deviation influence in the constraint function). Here, also the reference, i.e., non-robust, gPC result as well as the MCA-based CC-OC from Section 9.3 is shown. Overall, the table at first shows the different sigmoid offsets for the different SubSim fulfillment levels obtained with the sigmoid scaling $a_s = 9 \cdot 10^7$ for the MCA-based results. These results suggest that the scaling is chosen sufficiently large for all probability levels as the offset is in the order of $\mathcal{O}(10^{-7})$.

Furthermore, Table 9.7 shows the mean final time, which is the cost function, as well as the minimal mean separation to any of the obstacles along the trajectory. For these values, there is also a column showing the relative difference to the reference gPC-SC case. Here, it can e.g., be seen that for a fulfillment probability level of $\xi = 99.9\,\%$ a reduction of the optimality ranging from 1.25%–3.9 % is seen depending on the weight (i.e., the optimal flight takes longer). On the other hand, this reduced optimality results in an improved robustness, where the minimal mean separation is used as an indicator for, ranging from 82%–268 %. Thus, the well-known trade-off in ROC between optimality and robustness becomes clear once more.

A similar behavior can also be seen for the larger fulfillment probabilities of $\xi = 99.99\,\%$ and $\xi = 99.999\,\%$. Here, the optimality is reduced by up to 6.8 %, while the robustness, i.e., the minimal separation, is improved by more than 450 %. It should be noted that for these fulfillment probabilities no further CCs weighted with the standard deviation are analyzed as the results behave similar as in the case with $\xi = 99.9\,\%$. Additionally, the standard deviation for small failure probabilities generally gets fairly large and thus, weighting the CC with the standard deviation is generally equivalent to just increasing the fulfillment probability by one magnitude directly.

Finally, the last column of Table 9.7 shows the minimal fulfillment probability encountered along the optimal trajectory as calculated in the post-processing: These probabilities are connected to the assigned minimal failure probability and fulfilled considering the NLP tolerances. It can especially be seen that the minimal fulfillment probability is increased by at least 94 % while the optimality is only reduced by approximately 7 % considering the non-robust, gPC-SC result. This shows the significant capabilities of the CC-OC approach when trading of optimality and robustness.

The following detailed discussion of the results starts with the with the probability level of 99.9 % including the different weights in Subsection 9.4.1 and concludes with the different probability levels in Subsection 9.4.2

Table 9.7: *Overview of Subset Simulation-based chance-constrained optimal control results with comparison to reference polynomial chaos and Monte Carlo-based chance-constrained results.*

Case	Sigmoid Offset b_s	Mean Optimal Time $\mathbb{E}[t_f]$	Relative Optimality Reduction	Minimal Mean Separation $d_{\text{obs,min}}$	Relative Robustness Improvement	Minimal Fulfillment Probability of CC along Trajectory
$\xi = 99.9\,\%$ $(w = 0.0)$	$\approx 1.5351 \cdot 10^{-7}$	$\approx 6.9099\,\text{s}$	$\approx 1.2543\,\%$	$\approx 0.4562\,\text{m}$	$\approx 82\,\%$	$\approx 99.89952\,\%$
$\xi = 99.9\,\%$ $(w = 0.25)$	$\approx 1.5351 \cdot 10^{-7}$	$\approx 6.9647\,\text{s}$	$\approx 2.0574\,\%$	$\approx 0.6342\,m$	$\approx 154\,\%$	$\approx 99.95978\,\%$
$\xi = 99.9\,\%$ $(w = 0.5)$	$\approx 1.5351 \cdot 10^{-7}$	$\approx 7.0184\,\text{s}$	$\approx 2.8442\,\%$	$\approx 0.7656\,\text{m}$	$\approx 206\,\%$	$\approx 99.91961\,\%$
$\xi = 99.9\,\%$ $(w = 0.75)$	$\approx 1.5351 \cdot 10^{-7}$	$\approx 7.0925\,\text{s}$	$\approx 3.9301\,\%$	$\approx 0.9203\,\text{m}$	$\approx 268\,\%$	$\approx 99.94673\,\%$
$\xi = 99.99\,\%$ $(w = 0.0)$	$\approx 1.7909 \cdot 10^{-7}$	$\approx 7.1038\,\text{s}$	$\approx 4.0957\,\%$	$\approx 0.9813\,\text{m}$	$\approx 292.5\,\%$	$\approx 99.99798\,\%$
$\xi = 99.999\,\%$ $(w = 0.0)$	$\approx 2.0467 \cdot 10^{-7}$	$\approx 7.2928\,\text{s}$	$\approx 6.8652\,\%$	$\approx 1.3768\,\text{m}$	$\approx 450\,\%$	$\approx 99.99924\,\%$
gPC-SC: $d_i = 3$	–	$\approx 6.8243\,\text{s}$	$= 0\,\%$	$d_{\text{obs,min}} \approx 0.2500\,\text{m}$	$= 0\,\%$	$\approx 5.88\,\%$
MCA-CC: $a_s = 9 \cdot 10^7$	$\approx 1.2792 \cdot 10^{-7}$	$\approx 6.8865\,\text{s}$	$\approx 0.9114\,\%$	$\approx 0.3875\,\text{m}$	$\approx 55\,\%$	$\approx 99.88005\,\%$

9.4.1 Fulfillment Probability Level 99.9% with Different Standard Deviation Weights

At first, the fulfillment probability of $\xi = 0.999$ $(= 99.9\%)$ is looked at (this is related to a failure probability of 10^{-3}, i.e., a minor failure, in the EASA specifications [40, p. 26]): Although this is not yet considered to "really" be a rare-event, it is used in this thesis to show the capabilities of the SubSim-based CC-OC and specifically the different optimal trajectories obtained by the weighting factor in (9.11). To start with the description of the robust, optimal results, Figure 9.14 shows the mean trajectories of the SubSim-based CC-OC results for the different weighting factors $w = 0$ (solid green), $w = 0.25$ (dashed dark red), $w = 0.5$ (dash-dotted dark green), and $w = 0.75$ (dotted light green). Additionally, the MCA-based result from Section 9.3 with the same homotopy factor as used in SubSim-based CC-OCP is shown as a reference (solid black line). A first observation from Figure 9.14 is that the trajectories get pushed further away from the obstacles compared to the MCA-based result to fulfill the increased probability level. A significant difference is also seen with different weighting factors, which shows that the confidence, i.e., the safety, in the results is improved by introducing the weighting.

To get a more detailed overview on the obstacle separation constraint Figure 9.15 shows the mean and standard deviation for the distance to obstacle one, while Figure 9.16 shows the same for obstacle two (once more, minimal separation limit by dashed, horizontal magenta line). Once more, the mean plots (upper ones in the figures) have a zoomed-in part that shows the time horizon in the area around the minimal separation. The dashed magenta line in this zoomed-in plots denotes the minimal separation limit. Specifically, the zoomed-in part shows that the non-weighted, i.e., $w = 0$, SubSim-based result and the MCA-based reference result show similar characteristics. Still, the non-weighted case maintains a slightly larger separation and reaches its minimal separation later. This similar result could be expected as the MCA-based case already fulfilled a higher probability level as required, which was close to the 99.9 % of this case (see Figure 9.13). More significant

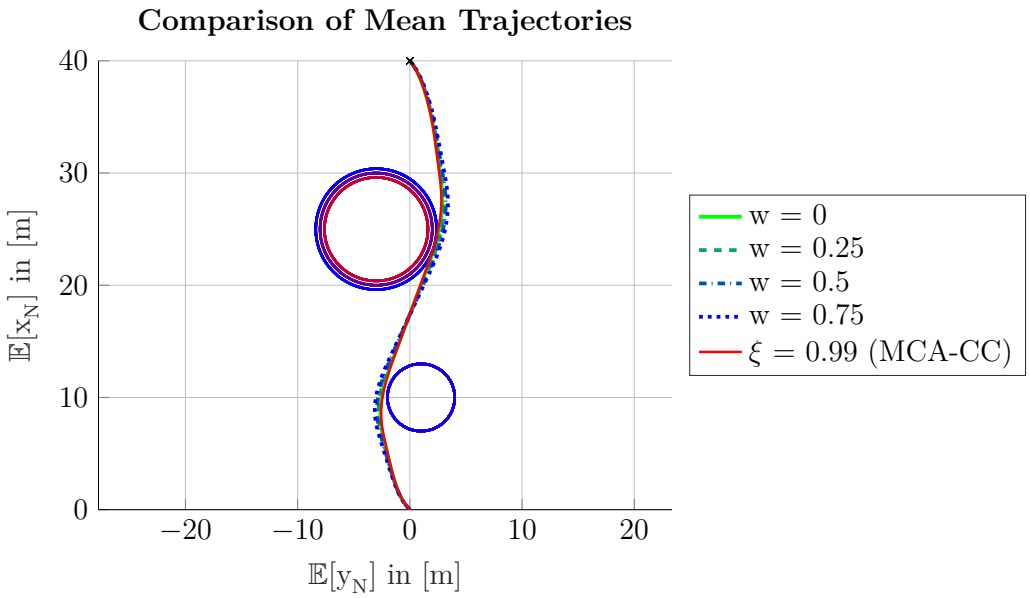

Figure 9.14: *Comparison of mean trajectories around obstacles of subset simulation (fulfillment probability level:* 99.9 %*) and Monte Carlo-based chance-constraint optimal control with different standard deviation weight factors.*

differences can be seen in the weighted case, where the distance to the minimal separation is larger. This comes at the cost of being less optimal, i.e., slower, which is as seen in Table 9.7.

Furthermore for the larger weights, looking at Figure 9.15 and Figure 9.16 it is seen that the standard deviation of the distance to the first obstacle shows no clear dependence or development with the assigned probability level and/or weighting. On the other hand, for the second obstacle there is a clearer structure: Before the obstacle is reached, the standard deviation is generally larger than for the reference MCA-based result, while it is smaller after the obstacle. It also is flatter close to the obstacle (around 4 s), which suggests a smoother trajectory around the obstacle and a smaller spread of the trajectories after the obstacle is passed.

To show the robustness improvement, Figure 9.17 depicts the CC fulfillment probability for the different SubSim-based CC-OC results. Take into account that this result was calculated in a post-processing step and thus, is calculated with the exact indicator function rather than the sigmoid approximation and independent parameter samples. Here, the horizontal magenta line denotes the desired CC fulfillment level. It is clear that all SubSim results fulfill the probability level. As it could be expected from the previous results, the non-weighted case (i.e., where the standard deviation of the fulfillment probability is not considered in the CC) is least robust, i.e., it reaches the minimal CC fulfillment bound. On the other hand, the weighted SubSim results have a safety distance to the bound, which is based on the incorporation of the failure probability standard deviation. Furthermore, it can be seen that the occurrence time of the failures is varying,

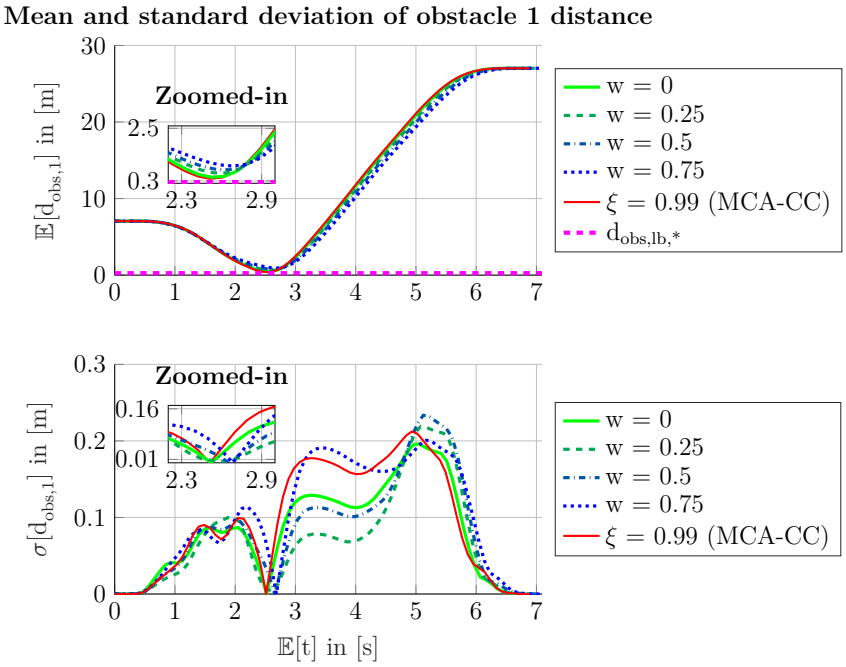

Figure 9.15: *Mean and standard deviation of obstacle one by subset simulation (fulfillment probability level: 99.9 %) and Monte Carlo-based chance-constraint optimal control with different standard deviation weight factors.*

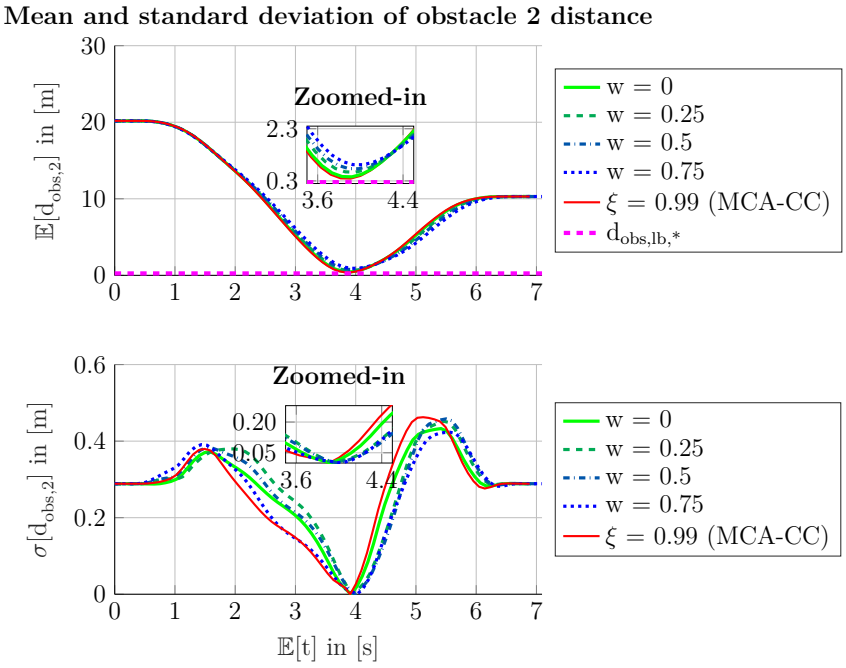

Figure 9.16: *Mean and standard deviation of obstacle two by subset simulation (fulfillment probability level: 99.9 %) and Monte Carlo-based chance-constraint optimal control with different standard deviation weight factors.*

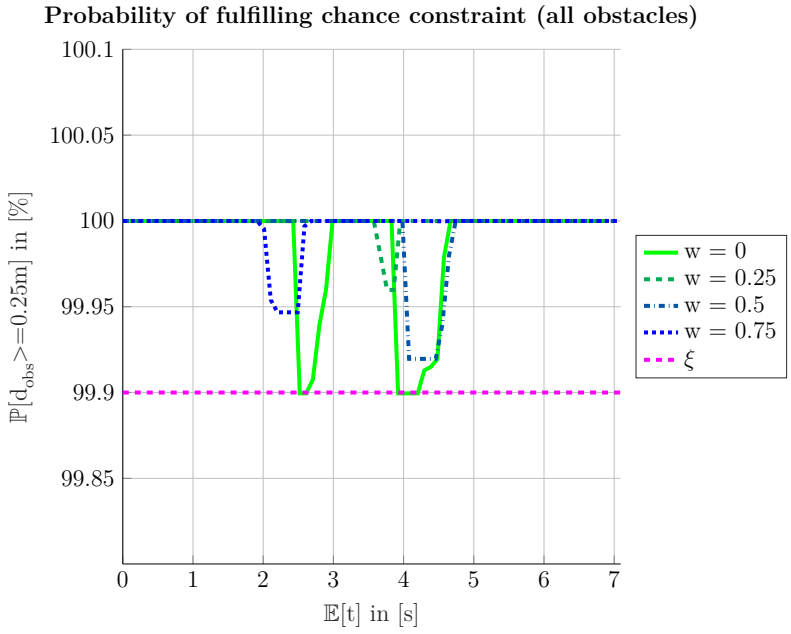

Figure 9.17: *Fulfillment probability ("expected value") of obstacle distance separation chance constraint by subset simulation-based (fulfillment probability level:* 99.9 %*) chance-constraint optimal control with different standard deviation weight factors.*

which suggests that there are some local minima encountered. For instance, the errors for the weight $w = 0.75$ occur mainly when entering the proximity of the first obstacle (starting from 2 s; see Figure 9.15), while e.g., in the case of $w = 0$, the errors are more likely to occur when exiting the obstacle proximity (see Figure 9.15 and Figure 9.16). This suggests different locally optimal points that can be explored on in further studies by e.g., globalization strategies.

It should be noted that there is no clear connection between the magnitude of the weight and the safety distance to the obstacle in Figure 9.17. This is mainly based on the connection of the failure probability standard deviation and its mean value, which is depicted by the CoV in Figure 9.18: As introduced in (2.45), the CoV is the relative magnitude of the failure probability standard deviation with respect to its mean value. Thus generally small values are desired to have a high confidence in the results. It can be seen in Figure 9.18 that the CoV in the failure areas is approximately 10 %, which suggests a high confidence. Here, both the weighted results with $w = 0.5$ and $w = 0.75$ suggest, in connection with Figure 9.17, that the tolerance of the minimum failure probability is reached and thus, the "most optimal" result is found by the optimizer. On the other hand, the case $w = 0.25$ seems to be too conservative, because of the small failure probability (see Figure 9.17 at around 4 s).

It is important to note that the large CoVs outside the failure areas in Figure 9.18 do not suggest a small confidence in the results, but rather that no failures where detected in this area by the SubSim. This is a feature of the Beta PDF that is used to calculate the mean and standard deviation of the SubSim failure probability (Subsection 2.2.4.4): As

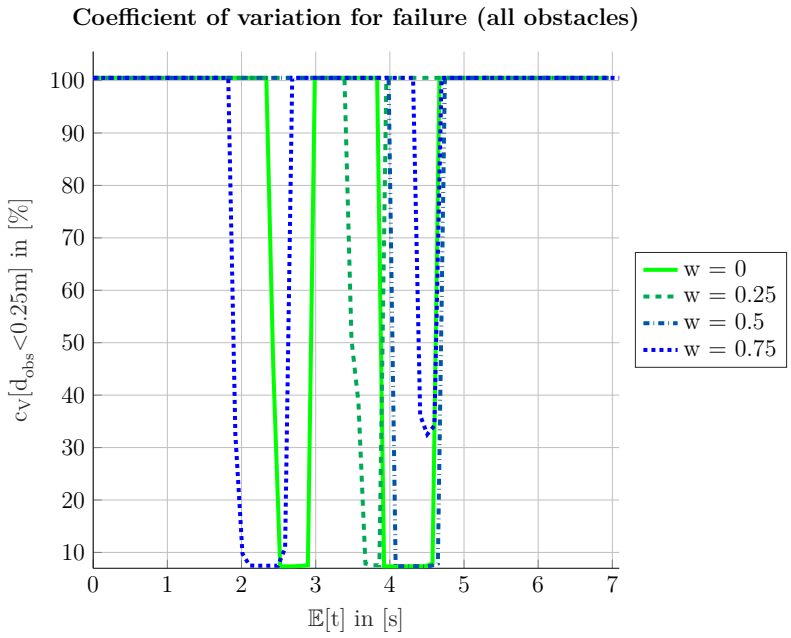

Figure 9.18: *Failure coefficient of variation of obstacle distance separation chance constraint by subset simulation-based (fulfillment probability level:* 99.9 %*) chance-constraint optimal control with different standard deviation weight factors.*

seen in (2.46) and (2.47), the mean and standard deviation will not reach exactly zero even with no detected failures. They will rather converges to very small values that are of similar magnitude, which results in large CoVs. This generally suggests that the SubSim result is inconclusive, i.e., no failures were detected, and should be reiterated with a different parameter set (e.g., different subset probability p_0 and/or different number of samples n_s). In this application, this is not necessary as there are actually points (like the IBC and FBC) that will never experience failures. Thus, large CoVs are natural when not being in the area of the obstacles.

9.4.2 Comparison of Different Fulfillment Probabilities

Concluding the SubSim-based CC-OC, the different fulfillment probabilities are compared. Therefore, Table 9.8 summarizes and compares the different fulfillment probabilities with the reference gPC and MCA-based CC-OC results. Once more, it can be seen that the optimality is reduced with increasing fulfillment probability up to 6.9 %, while the robustness is significantly increased by up to 450 %. This increase in robustness shows that significant safety distances to the obstacles must be fulfilled to assure that failures become rare-events. This behavior is mainly due to the Gaussian-distributed rotor uncertainty. In addition to this optimality and robustness statistics, Table 9.8 also shows the minimal fulfillment probabilities (i.e., the smallest value along the trajectory). It should be noted that these probabilities were calculated in a post-processing step. Here, it can be seen that especially the fulfillment probability case $\xi = 0.9999 = 99.99\,\%$ is too conservative, while the other

Table 9.8: *Comparison of subset simulation-based chance-constrained results with different probability levels and Monte Carlo-based chance-constrained results.*

Case ($w = 0.0$)	Mean Optimal Time	Relative Optimality Reduction	Minimal Mean Separation	Relative Robustness Improvement	Minimal Probability on Trajectory (Post)
$\xi = 99.9\,\%$	$\mathbb{E}[t_f] \approx 6.9099\,\mathrm{s}$	$\approx 1.2543\,\%$	$d_{\mathrm{obs,min}} \approx 0.4562\,\mathrm{m}$	$\approx 82\,\%$	$\approx 99.89952\,\%$
$\xi = 99.99\,\%$	$\mathbb{E}[t_f] \approx 7.1038\,\mathrm{s}$	$\approx 4.0957\,\%$	$d_{\mathrm{obs,min}} \approx 0.9813\,\mathrm{m}$	$\approx 292.5\,\%$	$\approx 99.99798\,\%$
$\xi = 99.999\,\%$	$\mathbb{E}[t_f] \approx 7.2928\,\mathrm{s}$	$\approx 6.8652\,\%$	$d_{\mathrm{obs,min}} \approx 1.3768\,\mathrm{m}$	$\approx 450\,\%$	$\approx 99.99924\,\%$
Ref.: $d_i = 3$	$\mathbb{E}[t_f] \approx 6.8243\,\mathrm{s}$	$= 0\,\%$	$d_{\mathrm{obs,min}} \approx 0.2500\,\mathrm{m}$	$= 0\,\%$	$\approx 5.88\,\%$
$\xi = 99\,\%$ (MCA-CC)	$\mathbb{E}[t_f] \approx 6.8865\,\mathrm{sunit}$	$\approx 0.9114\,\%$	$d_{\mathrm{obs,min}} \approx 0.3875\,\mathrm{m}$	$\approx 55\,\%$	$\approx 99.88005\,\%$

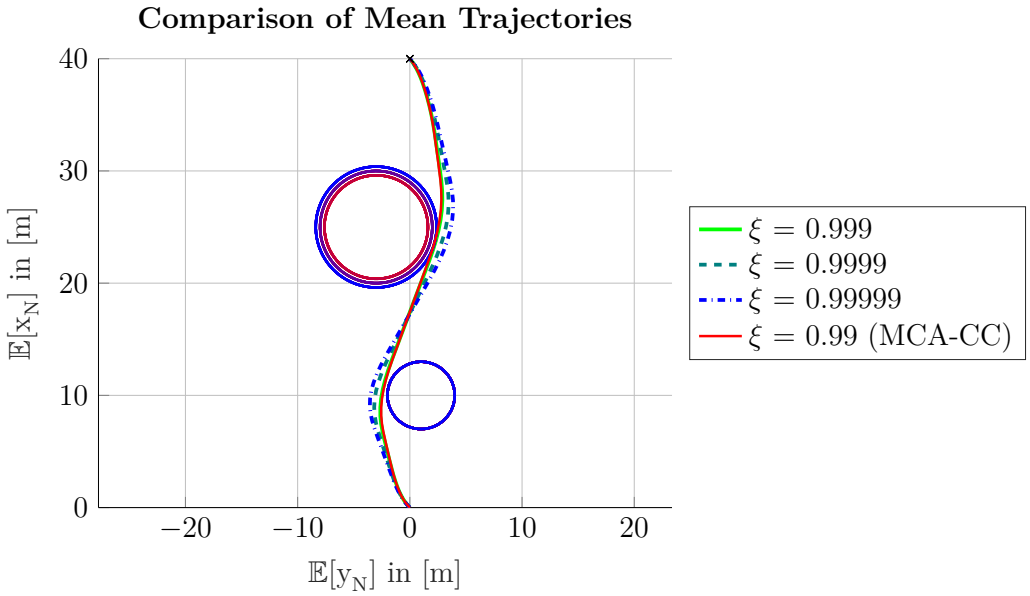

Figure 9.19: *Comparison of mean trajectories around obstacles of subset simulation and Monte Carlo-based chance-constraint optimal control with different probability levels.*

two SubSim-based CC-OC are very close to the minimal CC level, especially considering that failures, e.g., for $\xi = 0.999 = 99.9\,\%$, are allowed in the magnitude of the NLP tolerances.

As a first result, Figure 9.19, once more, compares the mean trajectories. Here, the three SubSim results as well as the MCA result is shown. While the trajectory with $\xi = 0.999 = 99.9\,\%$ is still close to the reference MCA-based results, a significantly increased deviation can be seen for the higher fulfillment probabilities. This is natural as the extremely rare values of the GAUSSIAN PDF (e.g., more than four standard deviations from the mean value), which was chosen as the rotor uncertainty, become of major influence and thus, the CC needs to consider these especially.

The obstacle related statistical moments are next covered by Figure 9.20 for the first obstacle and Figure 9.21 for the second obstacle (minimal separation limit: dashed, horizontal magenta line). Once more, the zoomed-in part shows that the minimal separation is significantly increased with increasing fulfillment level, while this minimum is reached

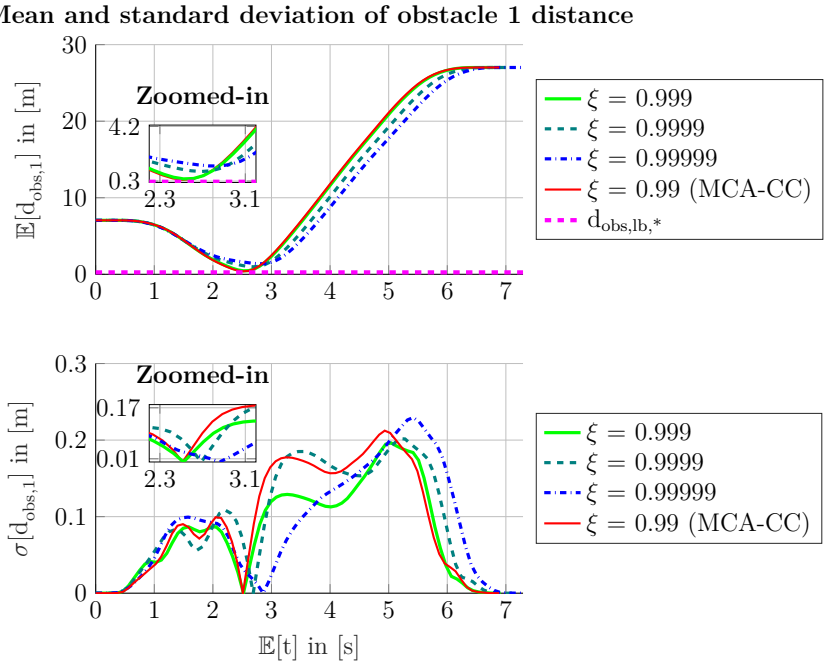

Figure 9.20: *Mean and standard deviation of obstacle one by subset simulation and Monte Carlo-based chance-constraint optimal control with different probability levels.*

later and the curve is generally smoother. Otherwise, a similar, but more robust, behavior like in Figures 9.15 and 9.16 can be seen. Specifically, the standard deviation once more shows to be larger ahead and smaller after the obstacle compared to the reference case.

Furthermore, Figure 9.22 shows, this time, the failure probability with the corresponding CoV for the failures depicted in Figure 9.23. It can be seen in Figure 9.22 that the case with $\xi = 0.999 = 99.9\,\%$ reaches the maximum allowable failure probability, i.e., $0.1\,\%$ at both obstacles. Generally, these maximal allowed failure probabilities are depicted by the horizontal dashed magenta lines. The CoV here is comparably small (see Figure 9.23), which suggests a high confidence. Still, as the failure probability is already at its boundary value, it cannot be ruled out completely that calculated trajectory might fail the CC level in real application. This behavior could e.g., be further evaluated using different SubSim parameters and additional confidence could be gained.

For the results with probability $\xi = 0.9999 = 99.99\,\%$ in Figure 9.22, the already mentioned conservatism can be seen as failures, first of all, only occur at the first obstacle (after 2 s) and, second of all, the failure probability is significantly smaller than what would be allowed. Once more, this behavior is related to the NLP solution process and here specifically the scalings and tolerances. Thus, less conservative results could also be achieved by adapting parameter settings suitably. It should be noted that the CoV for this failure is still comparably small, as seen in Figure 9.23, and thus, the confidence in the results and the fulfillment of the desired probability threshold is very high. This

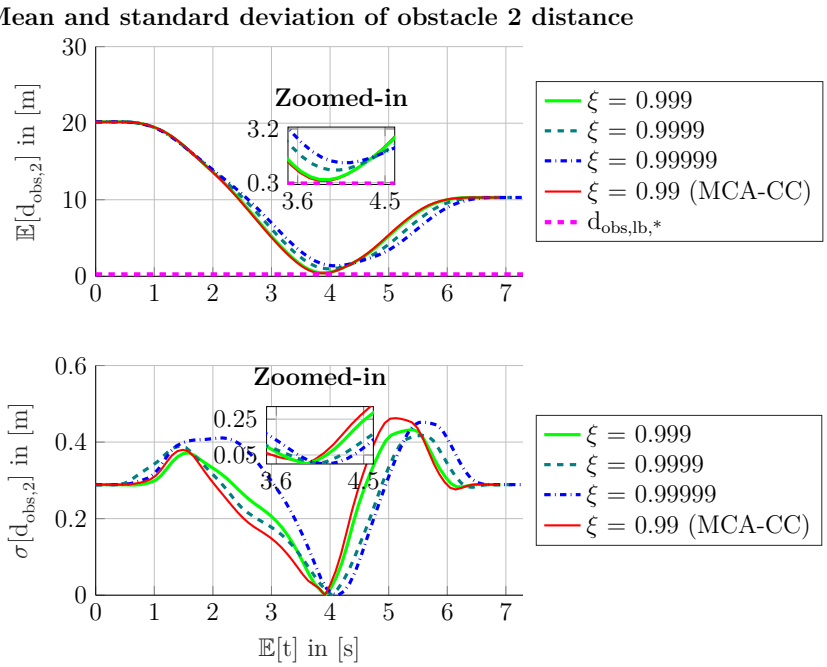

Figure 9.21: *Mean and standard deviation of obstacle two by subset simulation and Monte Carlo-based chance-constraint optimal control with different probability levels.*

is specifically true in this case as the failure probability in Figure 9.22 has a significant distance to its bound value. Thus, the conservatism is of benefit in this case as it improves the confidence in the results.

Finally, the probability level $\xi = 0.99999 = 99.999\,\%$ is on a similar magnitude like its maximal allowed failure probability. This was also already seen in Table 9.8 and suggests that the result is as optimal as possible and no further conservatism is introduced. Still, as for the case with $\xi = 0.999 = 99.9\,\%$ failure probability, this optimality with no conservatism suggests the possibility of failures occurring in real applications when looking e.g., at the magnitude of the CoV in Figure 9.23.

Concluding, it can be stated that both the MCA as well as SubSim-based CC-OC algorithm have proven viable in application. Specifically, the desired and novel fulfillment of rare-event CC could be shown.

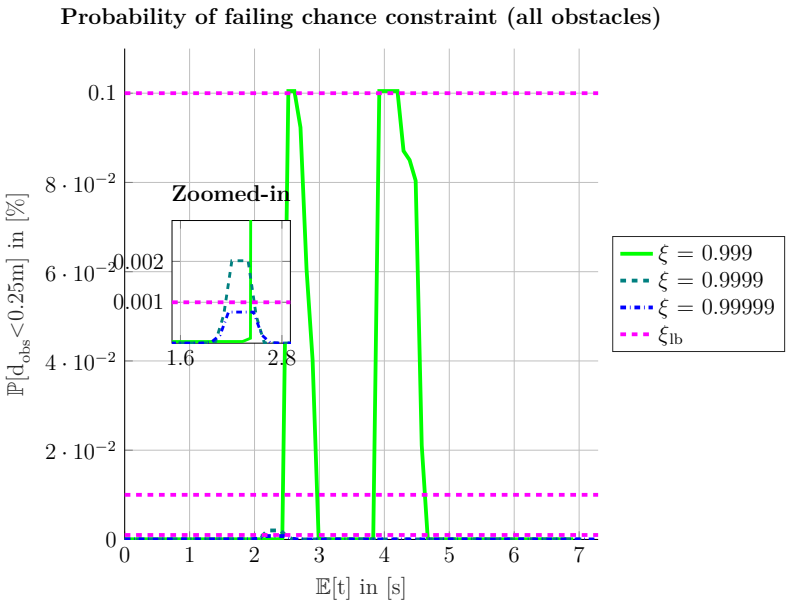

Figure 9.22: *Failure probability ("expected value") of obstacle distance separation chance constraint by subset simulation-based chance-constraint optimal control with different probability levels.*

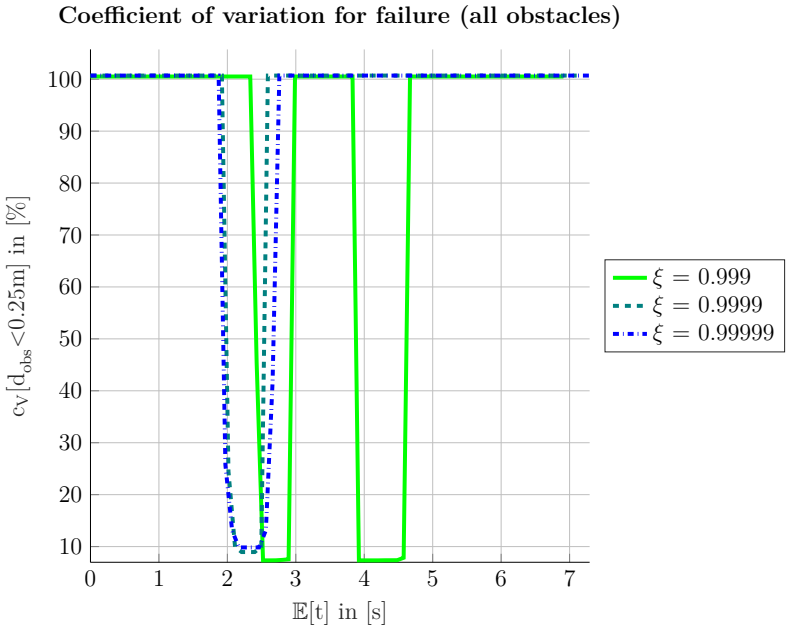

Figure 9.23: *Comparison of failure coefficient of variation of obstacle distance separation chance constraint by subset simulation-based chance-constraint optimal control for different probability levels.*

9.5 Outlook: Application of Chance-constrained Optimal Control to Landing in Urban Area

The previous examples showed the applicability and viability of the developed ROC methods for general trajectory optimization problems. Still, the applicability of the developed methods to current engineering problems might not yet be fully visible and clear. Thus, this section gives a brief outlook on the applicability of the developed CC-OC, without too much detail on the implementation and solution process, in the context of calculating robust landing trajectories in urban areas. These are specifically important for the growing air taxi industry.

The general problem setup is depicted in Figure 9.24: Here, the quadcopter, used before, is hovering at its initial position (southward orientation) and waiting to land in front of the Munich central station. The landing position is denoted by the blue cylinder and should be reached with a probability of at least 99 %. Additionally, the central station building (red wall) should be avoided with the same probability. The uncertainty is a UNIFORM wind field from east (left in the picture) towards the building, i.e., it pushes the quadcopter towards the building. The uncertainty is given by $v_W \sim \mathcal{U}\left(a = -1.5\frac{\mathrm{m}}{\mathrm{s}}, b = 0\frac{\mathrm{m}}{\mathrm{s}}\right)$, i.e., wind in west direction. It should be noted that this uncertainty is around 15 % of the quadcopter's maximal flight velocity and thus, already significant. The cost function is chosen to be a trade-off between minimal time and minimal control effort (i.e., energy), with the latter being more important. Thus, smoother, less aggressive trajectories should ensue.

In Figure 9.24, the reference optimal solution (no wind) is furthermore given by the yellow line, while the mean gPC-SC solution is given in magenta, and the mean CC-OC solution is in cyan. It can be mainly seen that the reference optimal solution without wind takes a more direct way to the landing area, while both the mean gPC-SC as well as the mean CC-OC trajectory are significantly influenced by the wind and pushed towards the building. Still, a reasonable safety distance is present. Take into account that a direct comparison between these trajectories is difficult as the wind uncertainty is not symmetric around zero wind (i.e., the value with whom the reference optimal results is calculated). Thus, the mean results for gPC-SC and CC-OC actually show results with non-zero wind influence. Therefore, the comparison should be mainly made with respect to the defined probabilistic criterions. In general, it can be stated that the reference optimal solution is one (extremal) realization of the non-robust gPC-SC solution and can therefore be analyzed as part of it.

The basic analysis of the touchdown points is given in Figure 9.25: Here, green circles denote touchdown points that are within the touchdown area while red crosses denote failures (i.e., a touchdown outside the desired magenta circle). The dashed blue lines are depicting the mean solution. The left plot shows the results when simulating the gPC-SC solution at 5000 random samples and updating the control command according to

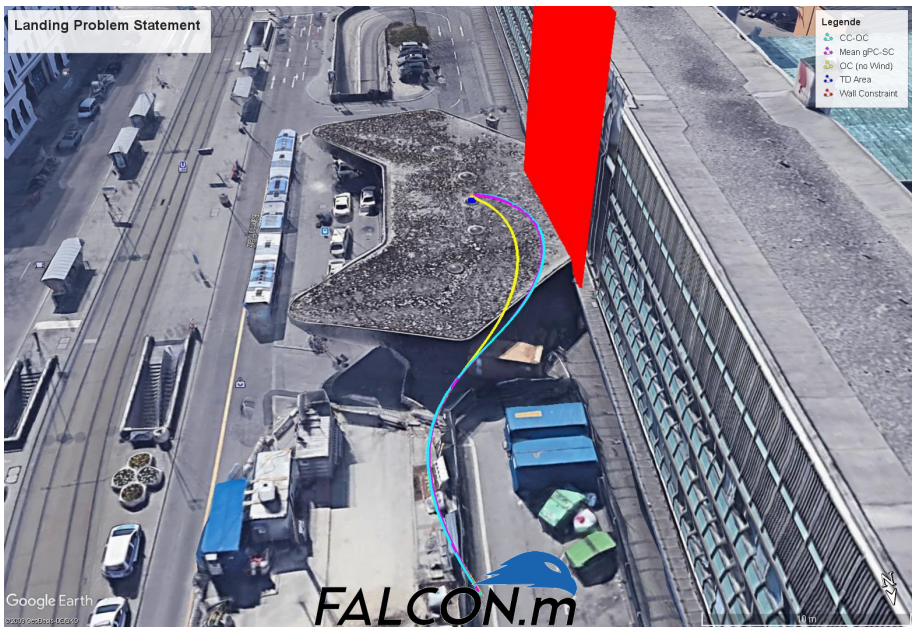

Figure 9.24: *Visualization of landing setup including constraints (blue touchdown cylinder; red area for building), optimal results with no wind (yellow), mean generalized polynomial chaos stochastic collocation solution (magenta), and chance-constrained optimal control solution (cyan).*

the current sample and the gPC expansion result. It should be noted that this procedure would require a wind measurement/estimation in the real application. The right plot shows the results when making a simulation at the same random samples with the CC-OC control expansion results. It can be seen that the CC-OC result significantly increases the number of points within the touchdown area (from approximately 50 % to 98.8 %). Additionally, the touchdown points are generally closer to the center of the touchdown area, which also increase the robustness to further uncertainties (e.g., model errors). It should be noted that the minor difference between the calculated CC-OC probability and the desired probability is based on the usage of different samples (and thus, generally the CIs of the solution) as well as the sigmoids and the NLP tolerances, as already discussed before.

To conclude, Figure 9.26 shows a subset of the sampling results (10 of 5000 random samples) using the CC-OC results: Here, red lines denote trajectories that either have a touchdown point outside the landing area or get to close to the building, while green lines fulfill the constraints. It can be seen that the trajectories exhibit a quite large spread due to the wind. Here, the statement from before that the reference optimal solution with no wind is an extremal solution of the robust problem becomes visible once more. Furthermore, it is clear that a significant number of trajectories fulfills the desired constraints and thus, applying the CC-OC approach yields safer results, because only around 1.2 % trajectories from the random sampling fail (see Figure 9.25). Additionally, it is clear that the trajectories mainly fail the landing area constraint while the building is hit by neither, which is a preferable behavior.

Thus, this brief outlook showed that the ROC frameworks developed in this thesis can be used in real engineering applications and problems to calculate safe and robust trajectories.

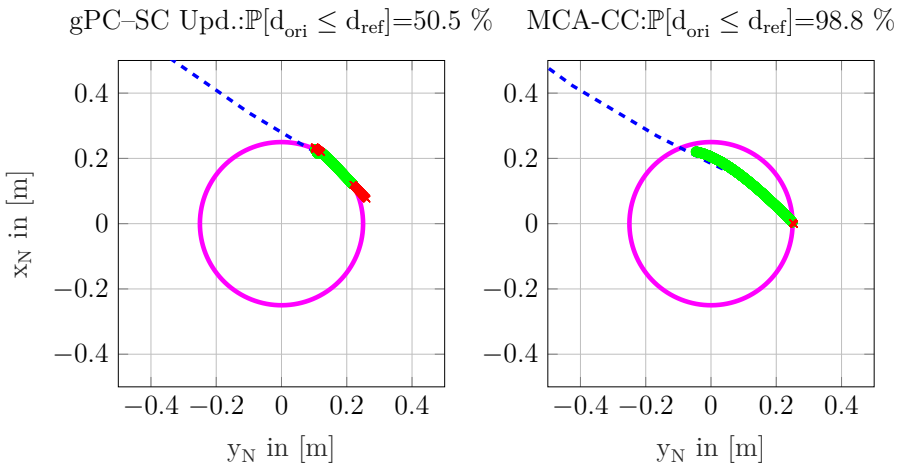

Figure 9.25: *Comparison of touchdown points for generalized polynomial chaos stochastic collocation (left plot; blue trajectory: mean) and chance-constrained optimal control (right plot; blue trajectory: mean) solution with probability of landing (*5000 *samples) in desired touchdown area (green circles: in touchdown area; red crosses; outside touchdown area).*

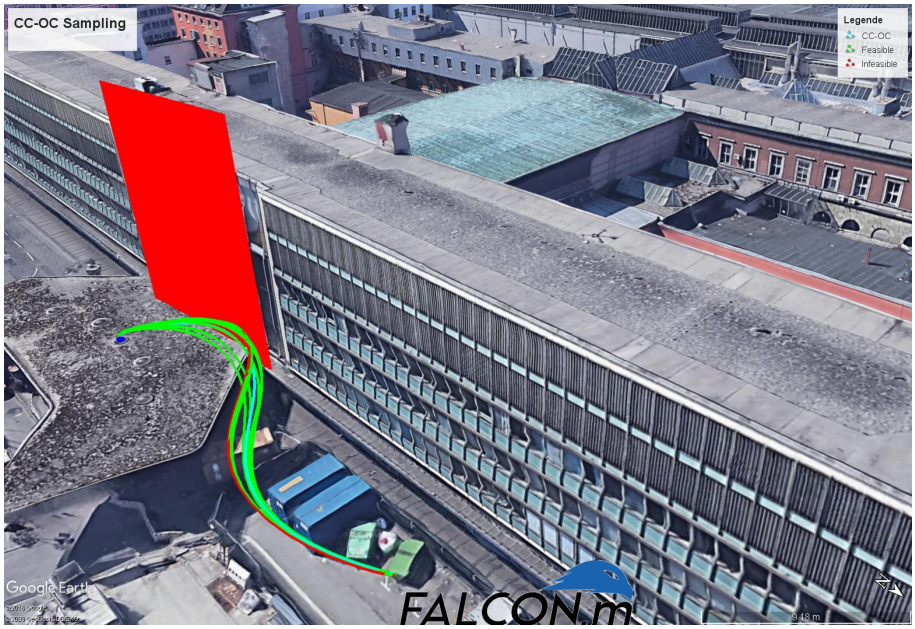

Figure 9.26: *Visualization of multiple landing trajectories for quadcopter under wind uncertainties (green lines: constraints fulfilled; red lines: constraints failed).*

Chapter 10

Conclusion and Perspective

This chapter wraps up the thesis starting with a review of the contributions from Section 1.4 in Section 10.1. Afterward, conclusive remarks and takeaways from the developed methods in this thesis are given in Section 10.2. An overview of possible future developments and perspectives for the introduced methods wraps up the thesis in Section 10.3.

10.1 Summary and Review of Contributions

The basic contribution of this thesis was to provide different methodologies to connect uncertainty quantification, by means of generalized polynomial chaos (gPC), with open-loop direct optimal control (OC) to create formulations for a robust open-loop direct optimal control (ROC) analysis of dynamic systems (e.g., by chance constraints (CCs) or distributed open-loop direct optimal control (DOC)).

In the following, the contribution list of Section 1.4 is reviewed in detail to specifically show the developments to the state of the art made in this thesis. Therefore, a short review of the current state is given and the developments made in this thesis are presented and described based on this:

Contribution 1: Direct Trapezoidal Collocation Transcription with Analytic Jacobian and Hessian using Generalized Polynomial Chaos within FSD optimal control tool for MATLAB® (FALCON.m) This OC transcription method is the basis for the further on developed ROC frameworks, especially for the chance-constrained open-loop direct optimal control (CC-OC) framework (Contribution 4 and Contribution 5). Basically, it transforms the stochastic open-loop direct optimal control problem (OCP) into a (large) deterministic OCP. Initial ideas on such a direct transcription have already been considered in [80, 86, 94]: Here, the authors always use the gPC expansion within Gauss pseudo-spectral OC transcription.

Within this thesis, these formulations and methods have been further developed and extended to collocation-based methods, specifically the trapezoidal collocation that is implemented in FALCON.m. Therefore, the methods are implemented as a special class

within the FALCON.m framework for general use within ROC. The benefit and motivation to use collocation instead of pseudo-spectral OC is the fact that collocation methods only approximate the problem locally and therefore create a very sparse gradient ("Jacobian") and Hessian, which are used in the NEWTON-type nonlinear program (NLP) iteration procedure. These sparse matrices are normally more efficient in the solution process of the NLP iteration, especially when large OCPs are considered. This is due to the fact that an efficient elimination procedure for the linear system solution of the NEWTON method can be conducted ("multi-frontal method"; [35]). Additionally, in collocation there is no necessity for the OC results (specifically the controls) to be smooth like there is in pseudo-spectral methods [15, p. 147ff.].

Furthermore, the trapezoidal collocation is rewritten in this contribution in a way that it is possible to directly propagate the expansion coefficients, i.e., solve the collocation defect (CD) equation in the gPC domain. For this, a time derivative based on the stochastic collocation (SC) method is derived and applied. Using this transcription, it is directly possible to control the accuracy of the statistical moments within the NLP iteration because of the equality constraint on the CD.

Finally, another development of this thesis for the direct transcription, using trapezoidal collocation of an OCP by gPC, is the derivation of an analytic Jacobian as well as analytic Hessian for the gPC discretization method. Especially, the availability of the analytic Hessian with gPC is novel and provides the possibility to e.g., conduct an efficient (post-optimal) sensitivity analysis, i.e., parameter influence analysis, on the optimization parameters. These are the expansion coefficients in this context and thus, the statistical moment changes of the calculated OCP result when varying parameters can be analyzed. Overall, the availability of post-optimal sensitivities then also allows for an online update of the robust trajectory.

Contribution 2: Optimal Gain Calculation for Control Loops using Generalized Polynomial Chaos and Sensitivity Analysis The calculation of (robust) optimal feedback control or adaptation gains has already been subject to research in e.g., [1, 3, 70, 93]. The studies [1, 70] use genetic algorithms to calculate the optimal adaptation gains. Although being easy to implement, genetic algorithms are normally slow in convergence and thus, reaching the optimum. Therefore, studies [3, 93] introduced NEWTON-type update rules, which are normally faster converging. Still, neither of the mentioned studies considers parametric uncertainties nor did it consider that a fast update of system parameters can be achieved by applying sensitivities.

Therefore within this contribution, a bi-level framework was developed that evaluates the closed control loop (either for an adaptive or a standard controller) by calculating a gPC representation of the system for the defined uncertainties in the lower levels. Subsequently, an upper level calculates the new set of controller gains. This framework is iterated until an optimal solution is found. As this procedure is generally time consuming, especially

using the often applied genetic algorithms in the upper level, an important feature of the proposed method in this thesis is the usage of sensitivities: These can either be calculated using post-optimal sensitivities or by propagating an additional differential equation for the sensitivities together with the dynamic system. Additionally, uncertainties (e.g., in the state matrix) are also considered within the closed-loop system using a gPC expansion. Thus, it must be considered that both the normal states as well as their sensitivities are expanded according to the gPC method. By this, the statistical moments of both the system as well as its sensitivities are available. These are then used in a parameter optimization to update the controller gains.

Ultimately, a hybrid method between the genetic algorithm and the sensitivity method is also implemented: This method initially searches for the global optimal region using the genetic algorithm and then switches to the sensitivity evaluation to find the optimal point faster and more reliable. Overall, this contribution thus provides a further development of controller design methods using ROC techniques.

It should be noted here that, although developed for gain design in this thesis, the proposed approach can easily be applied to other parameter design tasks within dynamic systems and general design tasks under uncertainties (e.g., shapes).

Contribution 3: Distributed Optimal Control of Generalized Polynomial Chaos Expansion DOC is a frequently used methodology in model predictive control (MPC)-based control loops [32, 37, 67, 88] and is the basis for a major ROC framework developed in this thesis. The general idea of many DOC methods is splitting the single, large OCP into multiple smaller OCPs that are easier to solve, but interconnected using a, generally, small set of connection variables. These connection variables are updated within a connection/coordination problem until the desired original solution for the large-scale OCP is reached. A review of current methods has recently been provided in [88]. DOC is also known as primal decomposition and is similar to the bi-level structure used in Contribution 2. Take into account that the goal of DOC here is to recover the original solution of the connected, large-scale OCP without simplifications.

Over the last couple of years, the consideration of uncertainties also became more prominent in DOC: A first study [32] considers an incorporation of gPC within a linear distributed open-loop direct optimal control problem (DOCP). Here, the gPC expansion is directly inserted within the dynamic equations and constraints and thus the specific system is transformed to the stochastic domain provided by gPC. In study [67], the sigma point approach, i.e., an unscented transformation based on a finite set of statistics, is applied to get to a stochastic OCP that is later on distributed to be solvable. Finally, study [37] once more uses the gPC method and directly inserts the gPC expansion into the DOC formulation for optimization. The authors apply their method to optimal power

flow problems, i.e., they try to minimize the operation cost while conserving the required power balance. They show that their box constraints and dynamic equations are fulfilled with high accuracy using the gPC expansion.

Within this thesis, the gPC expansion is also applied to create a DOCP, but rather than inserting the gPC expansion within the dynamic model equations, the SC method is used to rewrite the calculation of the statistical moments such that standard deterministic OCPs at the SC evaluation points ("nodes") can be solved. This has the benefit that it is possible to use the already tested deterministic model and that it is not required to reformulate the baseline OCP formulation. Consequently, this makes the method easier applicable to a broader range of standard OCP formulations (e.g., deterministic nonlinear models with nonlinear constraints). The usage of the SC method, together with a reformulation of the statistical moments, also enables to formulate a connection problem for e.g., a distributed cost functional that does not require sensitivities for the update of the connection variables. This is especially beneficial for models where e.g., the Hessian is costly to evaluate or where the Karush-Kuhn-Tucker (KKT) matrix of the NLP solver cannot be accessed to calculate the sensitivities. Thus, a development to the state of the art is made by this thesis.

Furthermore, it is also shown within this thesis that the DOCP with distributed cost function converges to the solution of the connected OCP by an analytic assessment of the optimality conditions (KKT conditions). Additionally, the distribution of constraints (e.g., CCs) is looked at, which requires the knowledge of sensitivities, but can be otherwise equally distributed using the proposed method. Here, both deterministic as well as probabilistic constraints can be applied.

Overall, this contribution proposed a general methodology for a distributed version of a robust open-loop direct optimal control problem (ROCP), which can even be applied to online OCP solution in future applications as the problem size is very good to control.

Contribution 4: Chance-constrained Optimal Control based on Monte Carlo Analysis and Generalized Polynomial Chaos CC-OC is the major topic of this thesis: The basic idea here is to introduce constraints on system variables within the OCP that are not deterministic but rather apply a probability of fulfilling/failing it. Such an OCP is clearly difficult to solve as it largely depends on the nonlinearity of the model in connection with the properties of the uncertainties, i.e., how the probability density function (PDF) of the system variables, which should fulfill the CC, look like. Therefore, many studies in the field rely on e.g.: ellipsoid relaxation, i.e., approximating a multivariate Gaussian distribution in the OCP by a cone constraint [130, 23ff.]; sampling, i.e., the efficient choice of evaluation points for the CC [147]; or density estimation techniques, i.e., estimating the PDF of the CC from a set of samples [25]. All of these techniques

try to handle the CC such that a good approximation for its value is calculated. For NEWTON-type OCPs another part, which plays a major role, is the fact that the resulting CC formulation must be smooth.

Within this thesis, a general framework that can deal with nonlinear models as well as general CC formulations is developed: Therefore, the already mentioned direct OC transcription using the gPC method can be used (Contribution 1; or by the similar idea in Contribution 3). Applying this method gives the capability to evaluate the gPC expansion very efficiently within each iteration of the NLP solution process at random samples. Thus, it is possible to conduct a Monte Carlo analysis (MCA) within each NLP step in a very efficient way as the gPC expansion reduces the MCA to a matrix-vector multiplication. The probability can then be calculated by counting the elements that are within the desired domain (i.e., as a logical decision). In order to create a smooth representation of this logic decision, a logistic function, the sigmoid (this is similar to the approach in study [147] with another function that is better suited to derive the analytic Jacobian and Hessian is used), is used as an approximation. Here, a homotopy strategy that iteratively sharpens the sigmoid is implemented as well such that finally a very good approximation of the original logic decision is achieved. Additionally, this homotopy strategy supplements convergence of the NLP solution process as loose approximations can be used in the beginning to get an initial, feasible result. In the process of sharpening the sigmoid using the homotopy, this initial result is then merely pushed towards the desired domain to reach the original tight bounds of the logical decision.

In summary, this contribution provides an initial framework to incorporate very general CCs within the ROCP, which is a development to the state of the art because very general CCs with a minimal amount of approximations can be treated. As the sampling can be done very efficient and with high accuracy, and also safety margins can be included using confidence intervals (CIs), it provides a viable option in reliability engineering and similar applications.

Contribution 5: Rare Event Chance-constrained Optimal Control based on Subset Simulation and Generalized Polynomial Chaos This contribution is a development of the previously introduced baseline CC-OC framework using MCA (Contribution 4). Within the contribution, the subset simulation (SubSim) method [9, 81] is incorporated as a way to calculate rare-event failure probabilities in the context of CC-OC. These small failure probabilities are e.g., necessary in safety critical applications and cannot be calculated reliably from MCA, as this would require way too many samples to capture the failure even only once. In turn, this then does not even return a viable probability estimation, because the number of samples is not "sufficiently" larger than the failure probability [9, p. 33]. Therefore, the central limit theorem (CLT), which is the basis of the MCA, does no longer provide a good approximation [9, p. 33]. Thus, the original CC-OC framework (Contribution 4) is augmented by the SubSim to achieve the desired

properties and be able to calculate rare-event failure probabilities. Current research in optimization with SubSim mainly relied on global optimization methods [76, 123] with only few constraints [77]. Thus, the work in this thesis provides significant developments.

Within this thesis, SubSim is incorporated within the chance-constrained open-loop direct optimal control problem (CC-OCP) and thus, the NEWTON-type direct OC scheme: One way of doing this is by adding a homotopy step that iteratively updates the samples created from SubSim until the SubSim as well as the OCP fulfill the desired maximal failure probability. It should be noted that this homotopy step can be connected with the homotopy step to update the sigmoid shape and rate parameters (Contribution 4). Thus, the computational burden is not increased compared to the anyway required adaptation of the sigmoid CC approximator function.

The created samples are then assigned as constants to the CC that uses them to calculate the probability of the constraint fulfillment within the NLP solution. Here, the probability can be either based on the Markov chain created from SubSim or a PDF estimation of the marginal failure distribution [148]. The update of the samples is done outside the NLP iteration until the SubSim as well as the CC-OCP fulfill the desired CC bound. The fact that the update is made outside the NLP also removes the stochasticity of the sample update from the NLP. Thus, deterministic NLP solvers can be used. Here, this sample update can also be made independent of the NLP process, which is based on the gPC expansion as an approximation for the analytic response, by simulating the dynamic system using the current robust control history within the sample creation algorithm (e.g., Metropolis-Hastings algorithm (MHA) or modified Metropolis-Hastings algorithm (MMHA)). Once more, this shows the benefit of the gPC expansion for ROCPs as it provides an analytic response surface that can be used for the fast solutions of the SubSim within the NLP.

Thus, it is clear from the contributions and how they were realized in this thesis that a development to the state of the art in ROC has been made. Still, their remain some open questions and further perspectives for future developments of the work at hand, which are discussed in the following.

10.2 Conclusive Remarks

In summary, the work at hand presented extensions and developments of frameworks in the context of ROC. Initially, an efficient method for uncertainty quantification relying on gPC was introduced that can be easily applied in OC applications. The benefit of the gPC method, compared to e.g., MCA, which is always used for verification of the developed methods, is the fast, efficient, and accurate calculation of the stochastic response surface. This makes it suitable for the application in OCPs, where the computational burden of solving a single OCP is normally high and thus, only the necessary, preferably small, number of OCPs should be solved.

Then, the combination of gPC and OC, which is established in research, led to the development of different ROC frameworks tailored to different application cases: A first framework (Contribution 2) introduced a bi-level methodology that can be used to design robust gains in control loops. It should be noted that the general bi-level methodology is a state-of-the-art method that gets extended in the thesis to the specific needs of the gPC methodology especially considering the gPC expansion of sensitivities. These are used within an optimization algorithm that calculates e.g., robust gains. Here, a major point of development was also the introduction of probabilistic constraints, which can e.g., be related to certification criteria of the controller architecture.

As the basic bi-level framework introduces artificial parameters, which might constrain the robust, optimal solution in an undesired way, an enhanced framework based on DOC was developed next in Contribution 3. This framework relied on a distribution of unconnected OC sub-problems that can be evaluated in a parallelized manner and a subsequent connection based on a small set of connection variables. These connection variables are chosen such that they ensure a solution of the original ROCP without simplifications and artificial constraints. For this, they are mathematically derived based on the used statistical moments and expansion coefficients in the problem formulation. Thus, they give a straightforward choice compared to the artificial parameters. Additionally, probabilistic constraints could be implemented directly.

A drawback of the developed DOC method was the fact that the OCPs had to be solved multiple times at the varying parameter set except for only very simple cases. This introduced a significant computational overhead. Additionally, it is not easily possible to calculate e.g., only a single, robust control history, because each DOCP calculates a specific optimal control history for the specific OCP. As these differ, the optimal control history is consequently also distributed, i.e., it carries statistical moments such as the standard deviation, instead of being only a single, mean robust history. To cope with this issue, a gPC collocation method was developed that introduced the gPC expansion coefficients as the decision variables of the NLP algorithm (Contribution 1). This gPC collocation framework gives the possibility to calculate robust trajectories within a single NLP optimization as the expansion coefficients are directly available and do not need to be calculated from the SC. A drawback of the method, compared to the DOC framework, is the fact that the NLP size grows very fast with the number of uncertainties and the expansion order yielding an NLP that might be difficult to solve. Thus, both the DOC as well as the gPC collocation framework supplement each other, i.e., the gPC collocation framework should be used for calculating an integrated solution result, while the DOC framework can be applied, if the gPC collocation cannot be applied due to e.g., size restrictions. Here, it should also be considered that further development can be made to connect both frameworks.

Overall, the major development of this thesis was the introduction of probabilistic constraints, i.e., CCs, for the ROC methods within Contribution 2–Contribution 5. These CCs relied on the efficient sampling that is possible by using the gPC expansion, which is always available in the developed frameworks. Thus, it is directly possible to do a MCA within the NLP iteration very efficiently as it is only necessary to solve a matrix-vector operation for this (Contribution 4). Additionally, a specifically tailored approximation algorithm based on sigmoids was developed that provided a smooth probabilistic constraint approximation for the NLP solver. In order to recover the originally desired solution, this sigmoid is iteratively adapted such that a very good approximation of the original CC was reached. Additionally, the CC could be adapted to consider errors, which might have been introduced by only using the approximate solution given by the gPC expansion and by the approximation using the sigmoids, by applying a CI optimization. Here, depending on the type of the constraint, the lower/upper bound of the CI could be used in the CC function by calculating the mean value and the standard deviation of the failure probability. Using this approach, a higher confidence in the CC fulfillment can be achieved, while still calculating an optimal trajectory.

As an extension of the standard MCA procedure, the SubSim method is also introduced in the context of the CC-OC framework (Contribution 5). This extension is particularly useful when studying rare-event failure probabilities as e.g., prominent in safety critical applications. Again, the efficient sampling of the gPC expansion could be applied in order to get fast updates of the probabilistic constraint in the NLP iteration. In general, it can be stated that especially the incorporation of SubSim in ROC provided a significant improvement on the current state of the art by this thesis.

10.3 Future Developments

Although the developed frameworks in this thesis extended available tools for modeling uncertainties within OCPs and extended the way of calculating robust trajectories, there are still areas of open question and for further research: As also stated in Contribution 1, the developed algorithms were mainly introduced for FALCON.m and thus, the trapezoidal collocation scheme used in it. Nonetheless, the proposed algorithm can be extended to general collocation methods [16, p. 132ff.], which can give improved convergence for some application cases.

Furthermore, a research direction can be the efficient choice of the SC nodes at which the deterministic OCPs are solved. Currently, these are chosen with respect to the Gaussian quadrature rules, which is a standard and well-established way. Still, the choice of nodes and weights might not be optimal for various problems and different nodes and weights might be better suited to approximate the response surface more accurate and efficient. Here, methods to calculate and update the nodes based on the closed-loop response can be applied [99]. It should be noted that this generally comes at the cost of having

a less generic framework. In this context, further developments of the current algorithms can also be made with respect to arbitrary polynomial chaos techniques [98, 136]. These are, although not explicitly covered and required in this thesis, also an important subject of research and can be used within the proposed frameworks directly.

Additionally, in the context of OC applications, (post-optimal) sensitivities can be used to improve the approximation quality of the response surface as not only the response but also the response derivative at the SC node can be calculated. This can e.g., be used to apply spline fits on the response surface, which improves the approximation order of the polynomial response surface without requiring an increase in the number of SC nodes.

Another research topic, which can be further exploited, is the incorporation of sparse grid techniques for the gPC expansion evaluation within the ROC frameworks. Although, initial work was already conducted during this thesis, there are still multiple opportunities to include better suited sparse grid, especially in the context of having a good representation of the real system response [18]. This generally yields to the incorporation of adaptive sparse grids that change based on the required accuracy for the gPC expansion [18]. These methods could significantly improve the convergence speed as well as accuracy of the methods and thus, open broader field of applications.

In addition, the sparsity pattern of the created NLP by e.g., the gPC collocation (Contribution 1) can be subject to further research especially in the context of exploitation for an efficient solution of the NLP [16, p. 134ff.]. Here, especially the structure of the gPC expansion can be exploited to be able to calculate an efficient solution of the linear system in the NLP [35]. An initial exploitation of this gPC expansion structure and the parallelization capabilities it provides, were already used within the DOC framework (Contribution 3). Additionally, a specifically tailored linear NLP solver could be implemented for this purpose.

With regard to the developed DOC framework (Contribution 3), there are also further developments possible: If interior point (IP) methods are used to solve the DOCPs, it is generally possible to adaptively update the barrier parameter. This means that in the beginning a “large” barrier parameter can be used such that the DOCPs are solved to a lower accuracy, when the connection variables are still far away from their optimal value (i.e., the connection error is still large). This could improve the convergence speed significantly as the DOCPs are easier so solve. With the connection variables approaching the optimal value, the barrier parameter of the DOCPs needs to be decreased as well to reach an accurate solution of the DOCP. By this procedure, it is also easier to ensure feasible DOCP solutions. It should be noted that this procedure can also be applied in the bi-level OC context of Contribution 2.

A further topic of research could be the incorporation of stochastic Newton-type updates of e.g., the gradients required in the NLP [4, 24]. Currently, all updates are made deterministically as the SC method provides deterministic expansion nodes that can be optimized deterministically. Nonetheless, the gPC method could also be used to calculate

the gradient in a stochastic manner, which would be e.g., of special interest in machine learning applications, but might also be applied in the context of CCs to give a better estimation on the exact convergence interval and the accuracy to which the CC has been solved, which could be extracted from the convergence area of the stochastic NEWTON method.

Regarding the CC-OC framework, one topic that may require further discussion is the connection of the desired fulfillment probability level with respect to the feasibility and optimality tolerance of the OCP. Especially with the rare-event CC-OC, a feasibility tolerance that is significantly smaller than the desired fulfillment probability is required. Otherwise, the NLP solver is allowed to fail the probability significantly because the tolerances are too large. Here, further research can thus develop appropriate scaling methodologies or rewriting the CC such that the effects of a too large feasibility tolerance are minimized. In order to reduce the computational effort to conduct the CC-OC it is also possible to improve the homotopy strategies: It is e.g., first of all possible to reformulate the homotopy on the sigmoid shape parameter by a maximization problem, i.e., finding the maximal possible sigmoid parameter that optimizes the cost functional as well as fulfills the CC. Here, the sigmoid shape should not be allowed to be smaller than a minimum threshold and it can be checked afterward if the calculated shape parameter creates a sharp-enough sigmoid. For the second homotopy, required in the SubSim, it can be viable to use (post-optimal) sensitivities on the initial SubSim to update the parameter set directly within the NLP. This assumes that the initial solution is already close to the desired rare-event and thus, only small changes in the initial parameter set are required. Additionally, sensitivities that update parameters based on changes in the trajectory are required, which requires further research.

Finally, the proposed CC-OC methods (Contribution 4 and Contribution 5) could e.g., be applied in the context of aircraft flight control law clearance [34]. Especially, the capability to efficiently calculate CCs can be used within the flight control law clearance, as the controller normally must fulfill probabilistic bounds on its constraints. These probabilistic constraints are generally also given in the context of rare-events and thus, the developed CC-OC framework with SubSim can play a role in e.g., the clearance procedure of control laws within future aircraft certification.

Bibliography

[1] A. Abraham and N. Pappa, An Improved MRAC Scheme for Non-linear Design of Adaption Gain gamma using Heuristic Algorithms, IFAC Proceedings Volumes, 47 (2014), pp. 734–739, https://doi.org/10.3182/20140313-3-IN-3024.00206.

[2] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, Tenth printing ed., 1972.

[3] B. B. Alagoz, G. Kavuran, A. Ates, and C. Yeroglu, Reference-shaping adaptive control by using gradient descent optimizers, PloS one, 12 (2017), p. e0188527, https://doi.org/10.1371/journal.pone.0188527.

[4] D. Anbar, A stochastic Newton-Raphson method, Journal of Statistical Planning and Inference, 2 (1978), pp. 153–163, https://doi.org/10.1016/0378-3758(78)90004-6.

[5] T. Arens, ed., *Mathematik*, Spektrum, Akad. Verl., Heidelberg, 2010.

[6] K. J. Aström, *Introduction to stochastic control theory*, vol. 70 of Mathematics in science and engineering, Acad. Pr, New York, 1970, http://www.sciencedirect.com/science/publication?issn=00765392&volume=70.

[7] S.-K. Au and J. L. Beck, Estimation of small failure probabilities in high dimensions by subset simulation, Probabilistic Engineering Mechanics, 16 (2001), pp. 263–277, https://doi.org/10.1016/S0266-8920(01)00019-4.

[8] S.-K. Au and E. Patelli, Rare event simulation in finite-infinite dimensional space, Reliability Engineering & System Safety, 148 (2016), pp. 67–77, https://doi.org/10.1016/j.ress.2015.11.012.

[9] S.-K. Au and Y. Wang, *Engineering Risk Assessment with Subset Simulation*, Wiley, s.l., 2014, https://doi.org/10.1002/9781118398050, http://lib.myilibrary.com/detail.asp?id=586315.

[10] J. F. Bard, *Practical Bilevel Optimization: Algorithms and Applications*, vol. 30 of Nonconvex Optimization and Its Applications, Springer US, Boston, MA, 1998.

[11] V. A. Bavdekar and A. Mesbah, Stochastic Nonlinear Model Predictive Control with Joint Chance Constraints, IFAC-PapersOnLine, 49 (2016), pp. 270–275, https://doi.org/10.1016/j.ifacol.2016.10.176.

[12] F. E. Beichelt, *Teubner-Taschenbuch der Stochastik: Wahrscheinlichkeitstheorie Stochastische Prozesse Mathematische Statistik*, Vieweg+Teubner Verlag, Wiesbaden, 2003.

[13] R. Bellman, *Dynamic programming*, Dover, 6 ed., 2003.

[14] A. Bemporad and M. Morari, Robust model predictive control: A survey, in Robustness in identification and control, A. Garulli and A. Tesi, eds., vol. 245 of Lecture Notes in Control and Information Sciences, Springer, London, 1999, pp. 207–226, https://doi.org/10.1007/BFb0109870.

[15] D. Benson, *A Gauss pseudospectral transcription for optimal control*, Phd thesis, Massachusetts Institute of Technology, Boston, 2005, http://hdl.handle.net/1721.1/28919 (accessed 04.10.2017).

[16] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*, Advances in design and control, Society for Industrial and Applied Mathematics (SIAM 3600 Market Street Floor 6 Philadelphia PA 19104), Philadelphia, Pa., 2nd ed. ed., 2010, http://epubs.siam.org/ebooks/siam/advances_in_design_control/dc19.

[17] M. Bittner, *Utilization of Problem and Dynamic Characteristics for Solving Large Scale Optimal Control Problems*, Dissertation, Technische Universität München, Garching, 06.04.2017.

[18] G. Blatman and B. Sudret, Adaptive sparse polynomial chaos expansion based on least angle regression, Journal of Computational Physics, 230 (2011), pp. 2345–2367, https://doi.org/10.1016/j.jcp.2010.12.021.

[19] G. I. Boutselis, G. de La Torre, and E. A. Theodorou, Stochastic Optimal Control using polynomial chaos variational integrators, in 2016 American Control Conference (ACC), Piscataway, NJ, 2016, IEEE, pp. 6586–6591, https://doi.org/10.1109/ACC.2016.7526707.

[20] Brad Ball, Richard Whitcomb, 20.04.2016, https://crgis.ndc.nasa.gov/historic/Richard_Whitcomb (accessed 14.11.2018).

[21] R. Brockhaus, W. Alles, and R. Luckner, *Flugregelung*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, https://doi.org/10.1007/978-3-642-01443-7.

[22] J. A. Bucklew, *Introduction to Rare Event Simulation*, Springer Series in Statistics, Springer, New York, NY, 2004, https://doi.org/10.1007/978-1-4757-4078-3, http://dx.doi.org/10.1007/978-1-4757-4078-3.

[23] C. Büskens, *Echtzeitoptimierung und Echtzeitoptimalsteuerung parametergestörter Probleme*, Habilitation, Universität Bayreuth, Bayreuth, Januar 2002, http://num.math.uni-bayreuth.de/de/research/phd/index.php (accessed 11.03.2018).

[24] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, A Stochastic Quasi-Newton Method for Large-Scale Optimization, SIAM Journal on Optimization, 26 (2016), pp. 1008–1031, https://doi.org/10.1137/140954362.

[25] J.-B. Caillau, M. Cerf, A. Sassi, E. Trélat, and H. Zidani, Solving chance constrained optimal control problems in aerospace via kernel density estimation, Optimal Control Applications and Methods, (2018), https://doi.org/10.1002/oca.2445, https://onlinelibrary-wiley-com.eaccess.ub.tum.de/doi/full/10.1002/oca.2445.

[26] R. H. Cameron and W. T. Martin, The Orthogonal Development of Non-Linear Functionals in Series of Fourier-Hermite Functionals, The Annals of Mathematics, 48 (1947), p. 385, https://doi.org/10.2307/1969178.

[27] T. S. Chihara, *An introduction to orthogonal polynomials*, vol. 13 of Mathematics and its applications, Gordon and Breach, New York [u.a.], 1978.

[28] A. Chinchuluun, A. Migdalas, and P. M. Pardalos, *Pareto Optimality, Game Theory And Equilibria*, vol. 17 of Springer Optimization and Its Applications, Springer Science+Business Media LLC, New York NY, 2008, http://dx.doi.org/10.1007/978-0-387-77247-9.

[29] B. Colson, P. Marcotte, and G. Savard, An overview of bilevel optimization, Annals of Operations Research, 153 (2007), pp. 235–256, https://doi.org/10.1007/s10479-007-0176-2.

[30] G. Cottrill, *Hybrid Solution of Stochastic Optimal Control Problems Using Gauss Pseudospectral Method and Generalized Polynomial Chaos Algorithms*, Dissertation, Air Force Institute of Technology, Ohio, 2012.

[31] G. G. Dahlquist, A special stability problem for linear multistep methods, BIT Numerical Mathematics, 3 (1963), pp. 27–43, https://doi.org/10.1007/BF01963532.

[32] L. Dai, Y. Xia, and Y. Gao, Distributed Model Predictive Control of Linear Systems with Stochastic Parametric Uncertainties and Coupled Probabilistic Constraints, SIAM Journal on Control and Optimization, 53 (2015), pp. 3411–3431, https://doi.org/10.1137/140994290.

[33] M. Diehl and J. Bjornberg, Robust Dynamic Programming for Min–Max Model Predictive Control of Constrained Uncertain Systems, IEEE Transactions on Automatic Control, 49 (2004), pp. 2253–2257, https://doi.org/10.1109/TAC.2004.838489.

[34] J. Diepolder, P. Piprek, B. Grüter, T. Akman, and F. Holzapfel, eds., *Aircraft Safety Analysis Using Generalized Polynomial Chaos: Air Traffic Management and Systems III*, Springer Singapore, 2019, https://link.springer.com/chapter/10.1007/978-981-13-7086-1_5 (accessed 25.07.2019).

[35] I. S. Duff and J. K. Reid, The Multifrontal Solution of Indefinite Sparse Symmetric Linear, ACM Transactions on Mathematical Software, 9 (1983), pp. 302–325, https://doi.org/10.1145/356044.356047.

[36] M. Eldred, Recent Advances in Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Analysis and Design, in 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Reston, Virigina, 2009, American Institute of Aeronautics and Astronautics, https://doi.org/10.2514/6.2009-2274.

[37] A. Engelmann, T. Muhlpfordt, Y. Jiang, B. Houska, and T. Faulwasser, Distributed Stochastic AC Optimal Power Flow based on Polynomial Chaos Expansion, in 2018 Annual American Control Conference (ACC), Piscataway, NJ, 2018, IEEE, pp. 6188–6193, https://doi.org/10.23919/ACC.2018.8431090.

[38] O. G. Ernst, A. Mugler, H.-J. Starkloff, and E. Ullmann, On the convergence of generalized polynomial chaos expansions, ESAIM: Mathematical Modelling and Numerical Analysis, 46 (2012), pp. 317–339, https://doi.org/10.1051/m2an/2011045, https://www.esaim-m2an.org/articles/m2an/pdf/2012/02/m2an110045.pdf.

[39] Europäische Kommission, *Flightpath 2050: Europe's vision for aviation ; maintaining global leadership and serving society's needs ; report of the High-Level Group on Aviation Research*, Policy / European Commission, Publ. Off. of the Europ. Union, Luxembourg, 2011.

[40] European Aviation Safety Agency, Proposed Special Condition for small-category vertical take-off and landing (VTOL) aircraft: SC-VTOL-01, 15 October 2018, https://www.easa.europa.eu/sites/default/files/dfu/SC-VTOL-01%20proposed.pdf (accessed 12 June 2019).

[41] European Aviation Safety Agency, Certification Specifications for Normal, Utility, Aerobatic, and Commuter Category Aeroplanes: CS-23, 20 July 2012, https://www.easa.europa.eu/sites/default/files/dfu/agency-measures-docs-certification-specifications-CS-23-CS-23-Amdt-3.pdf (accessed 12 June 2019).

[42] European Aviation Safety Agency, Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes: CS-25, 5 November 2018, https://www.easa.europa.eu/sites/default/files/dfu/CS-25%20Amendment%2022.pdf (accessed 12 June 2019).

[43] G. P. Falconí, J. Angelov, and F. Holzapfel, Hexacopter Outdoor Flight Test Results Using Adaptive Control Allocation Subject to an Unknown Complete Loss of One Propeller, in 3rd Conference on Control and Fault-Tolerant Systems (SysTol), 2016, pp. 367–374, https://doi.org/10.1109/SYSTOL.2016.7739779.

[44] V. Feoktistov, *Differential Evolution: In Search of Solutions*, Springer, New York, 2006.

[45] A. V. Fiacco, Sensitivity analysis for nonlinear programming using penalty methods, Mathematical Programming, 10 (1976), pp. 287–311, https://doi.org/10.1007/BF01580677.

[46] F. Fisch, *Development of a Framework for the Solution of High-Fidelity Trajectory Optimization Problems and Bilevel Optimal Control Problems*, Dissertation, Technische Universität München, München, 2011, http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20110221-1001868-1-4 (accessed 11.08.2016).

[47] F. Fisch, J. Lenz, F. Holzapfel, and G. Sachs, On the Solution of Bilevel Optimal Control Problems to Increase the Fairness in Air Races., in AIAA Atmospheric Flight Mechanics Conference, 2010.

[48] C. A. Floudas, ed., *Encyclopedia of optimization: With 247 tables*, Springer reference, Springer, New York, NY, 2009.

[49] G. Foderaro and S. Ferrari, Necessary conditions for optimality for a distributed optimal control problem, in 2010 49th IEEE Conference on Decision and Control, Piscataway, NJ, 2010, IEEE, pp. 4831–4838, https://doi.org/10.1109/CDC.2010.5718021.

[50] F. Gavilan, R. Vazquez, and E. F. Camacho, Chance-constrained model predictive control for spacecraft rendezvous with disturbance estimation, Control Engineering Practice, 20 (2012), pp. 111–122, https://doi.org/10.1016/j.conengprac.2011.09.006.

[51] M. Gerdts, *Optimal Control of ODEs and DAEs*, De Gruyter textbook, Walter de Gruyter GmbH Co.KG, s.l., 1. aufl. ed., 2012, https://doi.org/10.1515/9783110249996, http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10527877.

[52] M. Gerdts, *Optimal Control of Ordinary Differential Equations and Differential-Algebraic Equations*, Habilitation, Universität Bayreuth, Bayreuth, December 2006, https://www.unibw.de/lrt1/gerdts/forschung/publikationen/habilitation.pdf (accessed 11.03.2018).

[53] P. E. Gill, E. Wong, W. Murray, and M. A. Saunders, User's Guide for SNOPT Version 7.4: Software for Large-Scale Nonlinear Programming.

[54] C. Graham and D. Talay, *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation*, vol. 68 of Stochastic Modelling and Applied Probability, Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 2013, https://doi.org/10.1007/978-3-642-39363-1, http://dx.doi.org/10.1007/978-3-642-39363-1.

[55] S. Gros, A Newton algorithm for distributed Semi-Definite Programs using the primal-dual interior-point method, in IEEE 53rd Annual Conference on Decision and Control (CDC), 2014, Piscataway, NJ, 2014, IEEE, pp. 3222–3227, https://doi.org/10.1109/CDC.2014.7039887.

[56] D. Guignard and F. Nobile, A Posteriori Error Estimation for the Stochastic Collocation Finite Element Method, SIAM Journal on Numerical Analysis, 56 (2018), pp. 3121–3143, https://doi.org/10.1137/17M1155454.

[57] E. Hairer and G. Wanner, Stiff differential equations solved by Radau methods, Journal of Computational and Applied Mathematics, 111 (1999), pp. 93–111, https://doi.org/10.1016/S0377-0427(99)00134-X.

[58] F. G. Harmon, Hybrid solution of nonlinear stochastic optimal control problems using Legendre Pseudospectral and generalized Polynomial Chaos algorithms, in 2017 American Control Conference (ACC), Piscataway, NJ, 2017, IEEE, pp. 2642–2647, https://doi.org/10.23919/ACC.2017.7963351.

[59] C. D. Heise, *Survivable Flight Control with Guaranteed Stability and Performance Characteristics*, PhD thesis, Technische Universität München, Garching, 28.06.2017, http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20170628-1345015-1-3.

[60] L. Höhndorf, C. Wang, P. Koppitz, S. Weiss, S. Olmos, C. Czado, and F. Holzapfel, Integration of Vine Copula Dependence Structures into Subset Simulation for Accident Probability Quantifications, in ICAS 2018 - CD-ROM proceedings, Bonn, 2018, The International Council of the Aeronautical Sciences c/o Deutsche Gesellschaft für Luft- und Raumfahrt.

[61] F. Holzapfel, Flugsystemdynamik I: Skript zur Vorlesung (in German).

[62] F. Holzapfel, Flugsystemdynamik II: Skript zur Vorlesung (in German).

[63] F. Holzapfel, Model Reference Adaptive Control: Lecture Slides.

[64] F. Holzapfel, Praktikum Optimalsteuerung: Practical Course Slides (in English).

[65] F. Holzapfel, *Nichtlineare adaptive Regelung eines unbemannten Fluggerätes*, Dissertation, Technische Universität München, München, 2004.

[66] G. N. Iyengar, Robust Dynamic Programming, Mathematics of Operations Research, 30 (2005), pp. 257–280, https://doi.org/10.1287/moor.1040.0129.

[67] Y. Jiang, P. Nimmegeers, D. Telen, J. van Impe, and B. Houska, A Distributed Optimization Algorithm for Stochastic Optimal Control, IFAC-PapersOnLine, 50 (2017), pp. 11263–11268, https://doi.org/10.1016/j.ifacol.2017.08.1618.

[68] P. E. T. Jorgensen and M.-S. Song, Entropy encoding, Hilbert space, and Karhunen-Loève transforms, Journal of Mathematical Physics, 48 (2007), p. 103503, https://doi.org/10.1063/1.2793569, https://aip.scitation.org/doi/pdf/10.1063/1.2793569.

[69] I. Kaliszewski, *Quantitative Pareto Analysis by Cone Separation Technique*, Springer US, Boston, MA, 1994.

[70] R. G. Kanojiya and P. M. Meshram, Optimal tuning of PI controller for speed control of DC motor drive using particle swarm optimization, in 2012 International Conference on Advances in Power Conversion and Energy Technologies (APCET), Piscataway, NJ, 2012, IEEE, pp. 1–6, https://doi.org/10.1109/APCET.2012.6302000.

[71] L. R. Khan and K. F. Tee, Risk-Cost Optimization of Buried Pipelines Using Subset Simulation, Journal of Infrastructure Systems, 22 (2016), p. 04016001, https://doi.org/10.1061/(ASCE)IS.1943-555X.0000287.

[72] K.-K. K. Kim and R. D. Braatz, Generalized polynomial chaos expansion approaches to approximate stochastic receding horizon control with applications to probabilistic collision checking and avoidance, in IEEE International Conference on Control Applications (CCA), 2012, Piscataway, NJ, 2012, IEEE, pp. 350–355, https://doi.org/10.1109/CCA.2012.6402473.

[73] K.-K. K. Kim and R. D. Braatz, Generalised polynomial chaos expansion approaches to approximate stochastic model predictive control, International Journal of Control, 86 (2013), pp. 1324–1337, https://doi.org/10.1080/00207179.2013.801082.

[74] D. E. Kirk, *Optimal Control Theory: An Introduction*, Dover Books on Electrical Engineering, Dover Publications, Newburyport, 2012, http://gbv.eblib.com/patron/FullRecord.aspx?p=1894759.

[75] G. Lei, J. Zhu, and Y. Guo, *Multidisciplinary Design Optimization Methods for Electrical Machines and Drive Systems*, Power Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 1st ed. 2016 ed., 2016, https://doi.org/10.1007/978-3-662-49271-0, http://gbv.eblib.com/patron/FullRecord.aspx?p=4391627.

[76] H.-S. Li, Subset simulation for unconstrained global optimization, Applied Mathematical Modelling, 35 (2011), pp. 5108–5120, https://doi.org/10.1016/j.apm.2011.04.023.

[77] H.-S. Li and S.-K. Au, Design optimization using Subset Simulation algorithm, Structural Safety, 32 (2010), pp. 384–392, https://doi.org/10.1016/j.strusafe.2010.03.001.

[78] M. Li, Generalized Lagrange Multiplier Method and KKT Conditions with Application to Distributed Optimization, IEEE Transactions on Circuits and Systems II: Express Briefs, (2018), p. 1, https://doi.org/10.1109/TCSII.2018.2842085.

[79] P. Li, M. Wendt, and G. Wozny, Robust model predictive control under chance constraints, Computers & Chemical Engineering, 24 (2000), pp. 829–834, https://doi.org/10.1016/S0098-1354(00)00398-7.

[80] X. Li, P. B. Nair, Z. Zhang, L. Gao, and C. Gao, Aircraft Robust Trajectory Optimization Using Nonintrusive Polynomial Chaos, Journal of Aircraft, 51 (2014), pp. 1592–1603, https://doi.org/10.2514/1.C032474.

[81] D. Löbl, *A Total Capability Approach for the Development of Safety-Critical Functions*, Dissertation, Technische Universität München, München, 2018, http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20180627-1423023-1-6 (accessed 13.11.2018).

[82] W.-L. Loh, On Latin hypercube sampling, The Annals of Statistics, 24 (1996), pp. 2058–2080, https://doi.org/10.1214/aos/1069362310.

[83] Y. Lu and Y. Arkun, Quasi-Min-Max MPC algorithms for LPV systems, Automatica, 36 (2000), pp. 527–540, https://doi.org/10.1016/S0005-1098(99)00176-4.

[84] J. Lunze, *Regelungstechnik*, Springer-Lehrbuch, Springer, Berlin, 8., neu bearb. aufl. ed., 2010.

[85] J. Lunze, *Regelungstechnik 2: Mehrgrößensysteme, Digitale Regelung*, Springer Berlin Heidelberg, Berlin, Heidelberg, 9., überarb. aufl. 2016 ed., 2016, http://dx.doi.org/10.1007/978-3-662-52676-7.

[86] Y. Matsuno and T. Tsuchiya, Stochastic 4D trajectory optimization for aircraft conflict resolution, in IEEE Aerospace Conference, 2014, Piscataway, NJ, 2014, IEEE, pp. 1–10, https://doi.org/10.1109/AERO.2014.6836275.

[87] A. Mesbah, Stochastic Model Predictive Control: An Overview and Perspectives for Future Research, IEEE Control Systems, 36 (2016), pp. 30–44, https://doi.org/10.1109/MCS.2016.2602087.

[88] M. A. Müller and F. Allgöwer, Economic and Distributed Model Predictive Control: Recent Developments in Optimization-Based Control, SICE Journal of Control, Measurement, and System Integration, 10 (2017), pp. 39–52, https://doi.org/10.9746/jcmsi.10.39.

[89] T. Müller-Gronbach, E. Novak, and K. Ritter, *Monte Carlo-Algorithmen*, Springer-Lehrbuch, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, https://doi.org/10.1007/978-3-540-89141-3, http://dx.doi.org/10.1007/978-3-540-89141-3.

[90] H. Nahrstedt, *Die Monte-Carlo-Methode: Beispiele unter Excel VBA*, essentials, Springer Vieweg, Wiesbaden, 2015, https://doi.org/10.1007/978-3-658-10149-7, http://dx.doi.org/10.1007/978-3-658-10149-7.

[91] H. Niederreiter, ed., *Monte Carlo and Quasi-Monte Carlo Methods 2002: Proceedings of a Conference held at the National University of Singapore, Republic of Singapore, November 25-28, 2002*, Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 2004, https://doi.org/10.1007/978-3-642-18743-8.

[92] Normenstelle Luftfahrt im DIN Deutsches Institut für Normung e.V., Begriffe, Größen und Formelzeichen der Flugmechanik - Bewegung des Luftfahrzeugs gegenüber der Luft, Oktober 1990, https://www.beuth.de/de/norm/din-9300-1/1608582 (accessed 04.03.2017).

[93] H. N. Nounou and K. M. Passino, Stable auto-tuning of the adaptation gain for direct adaptive control, in American Control Conference, 2000, Piscataway, July 2000, I E E E, pp. 2154–2158 vol.3, https://doi.org/10.1109/ACC.2000.879582.

[94] K. Okamoto and T. Tsuchiya, Optimal Aircraft Control in Stochastic Severe Weather Conditions, Journal of Guidance, Control, and Dynamics, 39 (2016), pp. 77–85, https://doi.org/10.2514/1.G001105.

[95] M. Ono, Y. Kuwata, and J. Balaram, Mixed-strategy chance constrained optimal control, in American Control Conference (ACC), 2013, Piscataway, NJ, 2013, IEEE, pp. 4666–4673, https://doi.org/10.1109/ACC.2013.6580559.

[96] A. B. Owen, A Central Limit Theorem for Latin Hypercube Sampling, Journal of the Royal Statistical Society. Series B (Methodological), 54 (1992), pp. 541–551.

[97] I. Papaioannou, W. Betz, K. Zwirglmaier, and D. Straub, MCMC algorithms for Subset Simulation, Probabilistic Engineering Mechanics, 41 (2015), pp. 89–103, https://doi.org/10.1016/j.probengmech.2015.06.006.

[98] J. A. Paulson, E. A. Buehler, and A. Mesbah, Arbitrary Polynomial Chaos for Uncertainty Propagation of Correlated Random Variables in Dynamic Systems, IFAC-PapersOnLine, 50 (2017), pp. 3548–3553, https://doi.org/10.1016/j.ifacol.2017.08.954.

[99] J. A. Paulson and A. Mesbah, Shaping the Closed-Loop Behavior of Nonlinear Systems Under Probabilistic Uncertainty Using Arbitrary Polynomial Chaos, in 2018 IEEE Conference on Decision and Control (CDC), [Piscataway, New Jersey], 2018, IEEE, pp. 6307–6313, https://doi.org/10.1109/CDC.2018.8619328.

[100] P. Piprek, *Classical and Nonlinear Adaptive Flight Control Development for a Quadcopter Testbed Close to Hover Flight: Klassische und nichtlineare adaptive Regelung eines Quadrocopters in der Nähe des Schwebefluges*, Sa, Technische Universität München, Garching, 29.09.2015.

[101] P. Piprek, S. Gros, and F. Holzapfel, A Distributed Robust Optimal Control Framework Based on Polynomial Chaos, in 5th CEAS Specialist Conference on Guidance, Navigation & Control, Council of European Aerospace Societies, 2019.

[102] P. Piprek, S. Gros, and F. Holzapfel, Rare Event Chance-Constrained Optimal Control Using Polynomial Chaos and Subset Simulation, Processes, 7 (2019), p. 185, https://doi.org/10.3390/pr7040185, https://www.mdpi.com/2227-9717/7/4/185/pdf (accessed 30.03.2019).

[103] P. Piprek and F. Holzapfel, Robust Trajectory Optimization combining Gaussian Mixture Models with Stochastic Collocation, in IEEE Conference on Control Technology and Applications (CCTA), 2017, pp. 1751–1756.

[104] P. Piprek and F. Holzapfel, BI-LEVEL TRAJECTORY OPTIMIZATION BY STOCHASTIC COLLOCATION FOR UNCERTAINTY INTERVAL CALCULATION, in ICAS 2018 - CD-ROM proceedings, Bonn, 2018, The International Council of the Aeronautical Sciences c/o Deutsche Gesellschaft für Luft- und Raumfahrt.

[105] P. Piprek and F. Holzapfel, ROBUST TRAJECTORY OPTIMIZATION OF VTOL TRANSITION MANEUVER USING BI-LEVEL OPTIMAL CONTROL, in ICAS 2018 - CD-ROM proceedings, Bonn, 2018, The International Council of the Aeronautical Sciences c/o Deutsche Gesellschaft für Luft- und Raumfahrt.

[106] H. Pirnay, R. López-Negrete, and L. T. Biegler, Optimal sensitivity based on IPOPT, Mathematical Programming Computation, 4 (2012), pp. 307–331, https://doi.org/10.1007/s12532-012-0043-2.

[107] E. Platteau, 17 airlines, airports, air navigation service providers and manufacturers to test low-CO2-emission flight procedures in real conditions, 18.11.08, https://www.sesarju.eu/news-press/press-releases/17-airlines-airports-air-navigation-service-providers-and-manufacturers-te (accessed 31.10.18).

[108] E. Platteau and E. Schoeffmann, SESAR: 400t of CO2 saved on AIRE flights, 09.03.10, https://www.sesarju.eu/news-press/press-releases/sesar-400t-co2-saved-aire-flights--509 (accessed 31.10.18).

[109] M. Richter and F. Holzapfel, Robust Noise Optimal Approach Trajectories, AIAA Guidance, Navigation, and Control (GNC) Conference, Boston, (2013), https://doi.org/10.2514/6.2013-4556.

[110] M. Rieck, *Discrete Controls and Constraints in Optimal Control Problems*, Dissertation, Technische Universität München, Garching, 15.02.2017.

[111] M. Rieck, M. Bittner, B. Grüter, J. Diepolder, and P. Piprek, FALCON.m User Guide, www.falcon-m.com (accessed 29.03.2019).

[112] G. O. Roberts and J. S. Rosenthal, Optimal scaling for various Metropolis-Hastings algorithms, Statistical Science, 16 (2001), pp. 351–367, https://doi.org/10.1214/ss/1015346320, https://projecteuclid.org/download/pdf_1/euclid.ss/1015346320.

[113] M. Rockinger and E. Jondeau, Entropy densities with an application to autoregressive conditional skewness and kurtosis, Journal of Econometrics, 106 (2002), pp. 119–142, https://doi.org/10.1016/S0304-4076(01)00092-6.

[114] C. Runge, Über emprirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten, Zeitung für Mathematik und Physik, 46 (1901), pp. 224–243, https://www.math.technion.ac.il/hat/fpapers/run3.pdf (accessed 17.05.2019).

[115] L. Rüschendorf, *Wahrscheinlichkeitstheorie*, Springer-Lehrbuch Masterclass, Springer Spektrum, Berlin, 2016.

[116] R. M. Schaich and M. Cannon, Maximising the guaranteed feasible set for stochastic MPC with chance constraints, IFAC-PapersOnLine, 50 (2017), pp. 8220–8225, https://doi.org/10.1016/j.ifacol.2017.08.1388.

[117] J. Sheng and J. Q. Sun, Feedback Controls and Optimal Gain Design of Delayed Periodic Linear Systems, Modal Analysis, 11 (2016), pp. 277–294, https://doi.org/10.1177/107754605040947.

[118] C. Sherlock, P. Fearnhead, and G. O. Roberts, The Random Walk Metropolis: Linking Theory and Practice Through a Case Study, Statistical Science, 25 (2010), pp. 172–190, https://doi.org/10.1214/10-STS327.

[119] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, Chapman & Hall/CRC, Boca Raton, Fla., 3. ed. ed., 2004, http://www.loc.gov/catdir/enhancements/fy0668/2006029391-d.html.

[120] D. Shi and F. Holzapfel, Control Design with Guaranteed Statistical Performance and Application to Flight Control, in AIAA Scitech 2019 Forum, 2019, https://arc.aiaa.org/doi/abs/10.2514/6.2019-1729 (accessed 29.07.2019).

[121] J.-J. E. Slotine and W. Li, *Applied nonlinear control*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[122] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: Dynamics, controls design, and autonomous systems*, Wiley, Hoboken, N.J., 3rd ed., [elektronische ressource] ed., 2016, https://doi.org/10.1002/9781119174882, http://gbv.eblib.com/patron/FullRecord.aspx?p=4039442.

[123] X.-S. Suo, X.-Q. Yu, and H.-S. Li, Subset simulation for multi-objective optimization, Applied Mathematical Modelling, 44 (2017), pp. 425–445, https://doi.org/10.1016/j.apm.2017.02.005.

[124] P. Swarnkar, S. Jain, and R. K. Nema, Effect of Adaptation Gain in Model Reference Adaptive Controlled Second Order System, Engineering, Technology & Applied Science Research, 1 (2011), pp. 70–75, https://www.etasr.com/index.php/ETASR/article/download/11/98.

[125] N. T. Thomopoulos, *Essentials of Monte Carlo simulation: Statistical methods for building simulation models*, Springer, New York, NY [u.a.], 2013.

[126] D. Treschev and O. Zubelevich, *Introduction to the perturbation theory of Hamiltonian systems*, Springer Monographs in Mathematics, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2010, https://doi.org/10.1007/978-3-642-03028-4, http://sub-hh.ciando.com/book/?bok_id=232879.

[127] M. E. Villanueva, J. C. Li, X. Feng, B. Chachuat, and B. Houska, Computing ellipsoidal robust forward invariant tubes for nonlinear MPC, in In Proceedings of the 20th IFAC World Congress, Toulouse, France, 2017, pp. 7436–7441.

[128] M. E. Villanueva, R. Quirynen, M. Diehl, B. Chachuat, and B. Houska, Robust MPC via min–max differential inequalities, Automatica, 77 (2017), pp. 311–321, https://doi.org/10.1016/j.automatica.2016.11.022.

[129] V. Vittaldev, R. P. Russell, and R. Linares, Spacecraft Uncertainty Propagation Using Gaussian Mixture Models and Polynomial Chaos Expansions, Journal of Guidance, Control, and Dynamics, 39 (2016), pp. 2615–2626, https://doi.org/10.2514/1.G001571.

[130] M. P. Vitus, *STOCHASTIC CONTROL VIA CHANCE CONSTRAINED OPTIMIZATION AND ITS APPLICATION TO UNMANNED AERIAL VEHICLES*, Dissertation, Stanford University, Stanford, 2012.

[131] A. Wächter, Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT, https://projects.coin-or.org/Ipopt/browser/stable/3.10/Ipopt/doc/documentation.pdf?format=raw (accessed 15.07.2016).

[132] A. Wächter, Short Tutorial: Getting Started With Ipopt in 90 Minutes, in Combinatorial Scientific Computing, U. Naumann, O. Schenk, H. D. Simon, and S. Toledo, eds., vol. 09061 of Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, 2009, http://drops.dagstuhl.de/opus/volltexte/2009/2089 (accessed 6.7.16).

[133] A. Wächter and L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical Programming, 106 (2006), pp. 25–57, https://doi.org/10.1007/s10107-004-0559-y.

[134] X. Wan, D. Xiu, and G. E. Karniadakis, Modeling Uncertainty In Three-dimensional Heat Transfer Problems, WIT Transactions on Engineering Sciences, 46 (2004).

[135] N. Wiener, The Homogeneous Chaos, American Journal of Mathematics, 60 (1938), p. 897, https://doi.org/10.2307/2371268.

[136] J. A. Witteveen and H. Bijl, Modeling Arbitrary Uncertainties Using Gram-Schmidt Polynomial Chaos, 44th AIAA Aerospace Sciences Meeting and Exhibit, (2006).

[137] C. B. Wu, G. H. Huang, W. Li, Y. L. Xie, and Y. Xu, Multi-stage stochastic inexact chance-constraint programming for an integrated biomass-municipal solid waste power supply management under uncertainty, Renewable and Sustainable Energy Reviews, 41 (2015), pp. 1244–1254, https://doi.org/10.1016/j.rser.2014.09.019.

[138] D. Xiu, *Generalized (Wiener-Askey) Polynomial Chaos*, Phd thesis, Brown University, Providence, 2004, https://www.brown.edu/research/projects/crunch/sites/brown.edu.research.projects.crunch/files/uploads/Dongbin%20Xiu%20Thesis.pdf (accessed 18.11.2018).

[139] D. Xiu, Fast numerical methods for robust optimal design, Engineering Optimization, 40 (2008), pp. 489–504, https://doi.org/10.1080/03052150801893631.

[140] D. Xiu, Fast Numerical Methods for Stochastic Computations: A Review, Communications in Computational Physics, 5 (2009), pp. 242–272.

[141] D. Xiu, *Numerical methods for stochastic computations: A spectral method approach*, Princeton University Press, Princeton, N.J, 2010, http://lib.myilibrary.com/detail.asp?ID=293631.

[142] D. Xiu and J. S. Hesthaven, High-Order Collocation Methods for Differential Equations with Random Inputs, SIAM Journal on Scientific Computing, 27 (2005), pp. 1118–1139, https://doi.org/10.1137/040615201.

[143] D. Xiu and G. E. Karniadakis, The Wiener-Askey Polynomial Chaos for Stochastic Differential Equations, SIAM Journal on Scientific Computing, 24 (2002), pp. 619–644, https://doi.org/10.1137/S1064827501387826.

[144] D. Xiu and G. E. Karniadakis, Modeling uncertainty in flow simulations via generalized polynomial chaos, Journal of Computational Physics, 187 (2003), pp. 137–167.

[145] Y. Yin, Genetic-Algorithms-Based Approach for Bilevel Programming Models, Journal of Transportation Engineering, 126 (2000), pp. 115–120, https://doi.org/10.1061/(ASCE)0733-947X(2000)126:2(115).

[146] X. Zhang, A. Georghiou, and J. Lygeros, Convex approximation of chance-constrained MPC through piecewise affine policies using randomized and robust optimization, in 2015 54th IEEE Conference on Decision and Control (CDC), Piscataway, NJ, 2015, IEEE, pp. 3038–3043, https://doi.org/10.1109/CDC.2015.7402675.

[147] Z. Zhao, F. Liu, M. Kumar, and A. V. Rao, A novel approach to chance constrained optimal control problems, in American Control Conference (ACC), 2015, Piscataway, NJ, 2015, IEEE, pp. 5611–5616, https://doi.org/10.1109/ACC.2015.7172218.

[148] K. M. Zuev, J. L. Beck, S.-K. Au, and L. S. Katafygiotis, Bayesian post-processor and other enhancements of Subset Simulation for estimating failure probabilities in high dimensions, Computers & Structures, 92-93 (2012), pp. 283–296, https://doi.org/10.1016/j.compstruc.2011.10.017.

[149] M. V. C. de Souza, M. J. Colaço, and A. J. K. Leiroz, Application of the generalized Polynomial Chaos expansion to the simulation of an internal combustion engine with uncertainties, Fuel, 134 (2014), pp. 358–367.

Appendix A

Coordinate Frames and Transformations

The coordinate frames used in this thesis are based on [92] as well as [61, 62, 65] and are adapted to the specific needs and nomenclature of the thesis. The transformation matrices are based on [61, 62, 65] as well. The used frames and transformations are introduced in the following sections, while Figure A.1 gives an initial, general overview of the used coordinate frames and the transformations, i.e., dependencies, in between them.

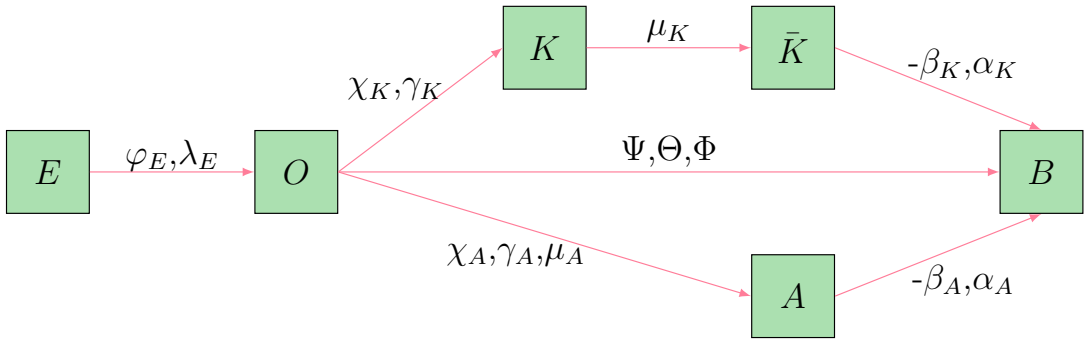

Figure A.1: *Coordinate systems, connections, and transformations in between them with respective transformation angles and order (after [61], [17, p. 14]).*

A.1 Earth-Centered, Earth-Fixed / WGS84 Frame

The earth-centered, earth-fixed (ECEF) frame is used to depict positional data on the earth. It has the index E and is located in the earth's center E. The x_E-axis points towards the GREENWICH MERIDIAN, while the z_E-axis is the earth's rotation axis. The y_E-axis forms a right-handed coordinate frame.

Because the definition of a position by the x_E, y_E, and z_E coordinate on the earth is uncommon in aviation, the World Geodetic System 1984 (WGS84) frame is commonly used as an alternative description. This frame denotes the position by geodetic latitude φ_E, geodetic longitude λ_E, and geodetic height h.

The mentioned definitions are illustrated in Figure A.2.

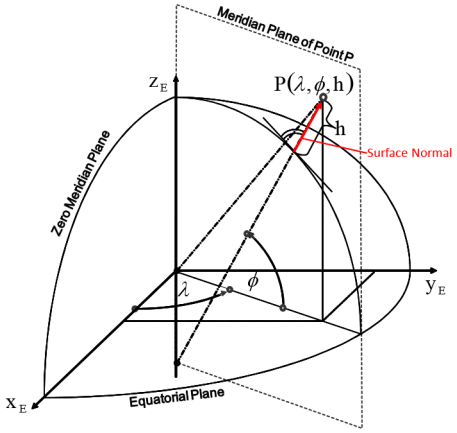

Figure A.2: *Earth-Centered, Earth-Fixed frame with WGS84 geodetic position definition [62].*

A.2 Body-fixed Frame

The body-fixed frame is a frame of reference for the aircraft that translates and rotates with the aircraft. It is generally used to note body related quantities such as forces, moments, and inertia. It is fixed in the aircraft reference point, R, and has the index B. The x_B-axis points towards the nose of the aircraft, the y_B-axis points towards the right wing, and the z_B-axis points downwards in the symmetric plane of the aircraft. These definitions are illustrated in Figure A.3.

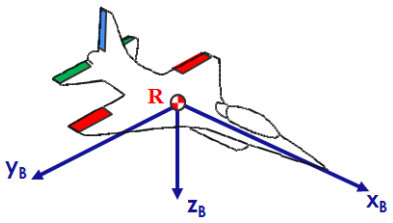

Figure A.3: *Body-Fixed Frame [62].*

A.3 Aerodynamic Frame

The aerodynamic frame, depicted in Figure A.4, is a frame of reference used to denote aerodynamic quantities such as the aerodynamic velocity or aerodynamic angle of attack. It is fixed in the aircraft's reference point R and has the index A. It translates with the aircraft and rotates with respect to the airflow around the aircraft. The x_A-axis points in the direction of the aerodynamic velocity, the z_A-axis points downwards in the symmetric plane of the aircraft, and the y_A-axis points to the right, setting up the right-handed coordinate frame.

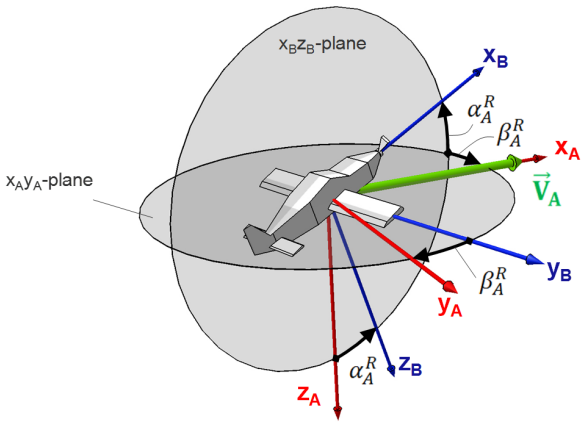

Figure A.4: *Aerodynamic Frame [62].*

A.4 Kinematic Frame

The kinematic frame, also called trajectory frame, is a frame of reference for the trajectory of the aircraft. It is visualized in Figure A.5: Again, the frame is fixed in the aircraft reference point R, while it holds the index K. It translates with the aircraft, while it rotates with respect to the kinematic velocity. The x_K-axis points in the direction of the kinematic velocity, the z_K-axis points downwards in the symmetric plane of the aircraft, and the y_K-axis points to the right, setting up the right-handed coordinate frame.

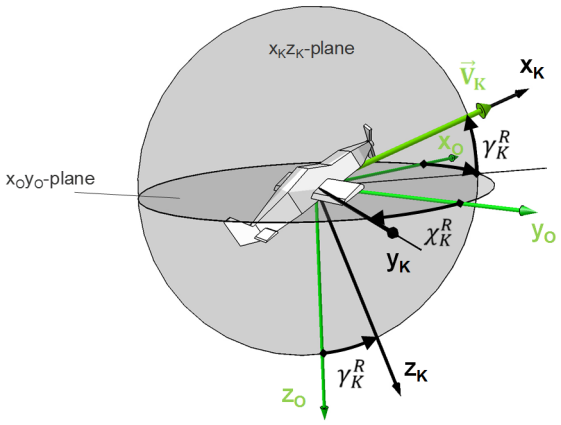

Figure A.5: *Kinematic Frame [62].*

A.5 North-East-Down Frame

The North-East-Down (NED) frame is a frame of reference used to denote the orientation of the aircraft. It is fixed in the aircraft's reference point R, translates with the aircraft, and is holding the index O. The x_O-axis points towards geographic north, the y_O-axis points towards geographic east, and the z_O-axis points downwards and is orthogonal with respect to the local geoid surface. This orthogonality relation is secured by a rotation of the NED frame by the transportation rate. The frame is visualized in Figure A.6.

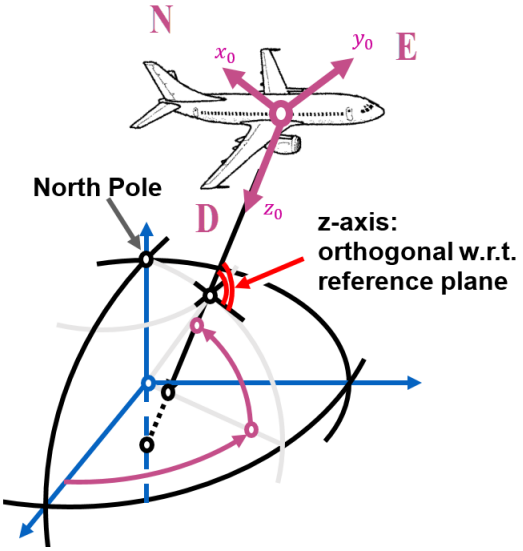

Figure A.6: *North-East-Down Frame [62].*

A.6 Geodetic/Navigation Frame

The geodetic frame, also called navigation frame, is a frame of reference used in the simplified view of a fixed-flat earth (FFE). Essentially, it is a (rotated) NED frame (Section A.5) that is fixed somewhere on the earth's surface. Thus, it only rotates with the earth's rate,

but it does not translate. It is fixed in the point O, and has the index N. The x_N-axis points to geographic north (or a rotated direction), the z_N-axis points downwards and is orthogonal with respect to the local geoid surface, and the y_N-axis points to geographic east (or the rotated direction). This frame of reference is depicted in Figure A.7.

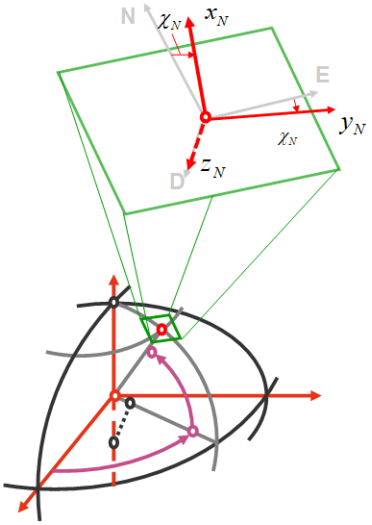

Figure A.7: *Geodetic/Navigation Frame [62].*

A.7 Transformation Matrices

As already introduced in Figure A.1 the coordinates frames introduced in the previous sections can be transformed in between using transformation angles. These transformations are introduced in the following. It should be noted that the transformation matrices in this thesis are orthonormal matrices, which makes it easy to calculate the inverse by simply transposing it [122, p. 10], i.e., $\mathbf{M}_{XY} = \mathbf{M}_{YX}^{-1} = \mathbf{M}_{YX}^{\mathsf{T}}$. In this context, it should be noted that the first index denotes the output frame while the second one is the input frame. The following formulas show the calculation of the transformation matrices [61, 62, 65]:

O frame to B frame:

$$\mathbf{M}_{BO} = \begin{bmatrix} \cos(\Theta)\cos(\Psi) & \cos(\Theta)\sin(\Psi) & -\sin(\Theta) \\ \sin(\Phi)\sin(\Theta)\cos(\Psi) - \cos(\Phi)\sin(\Psi) & \sin(\Phi)\sin(\Theta)\sin(\Psi) + \cos(\Phi)\cos(\Psi) & \sin(\Phi)\cos(\Theta) \\ \cos(\Phi)\sin(\Theta)\cos(\Psi) + \sin(\Phi)\sin(\Psi) & \cos(\Phi)\sin(\Theta)\sin(\Psi) - \sin(\Phi)\cos(\Psi) & \cos(\Phi)\cos(\Theta) \end{bmatrix} \tag{A.1}$$

O frame to K frame:

$$\mathbf{M}_{KO} = \begin{bmatrix} \cos(\chi_K)\cdot\cos(\gamma_K) & \sin(\chi_K)\cdot\cos(\gamma_K) & -\sin(\gamma_K) \\ -\sin(\chi_K) & \cos(\chi_K) & 0 \\ \cos(\chi_K)\cdot\sin(\gamma_K) & \sin(\chi_K)\cdot\sin(\gamma_K) & \cos(\gamma_K) \end{bmatrix} \tag{A.2}$$

O frame to A frame:

$$\mathbf{M}_{AO} = \begin{bmatrix} \cos(\gamma_A)\cos(\chi_A) & \cos(\gamma_A)\sin(\chi_A) & -\sin(\gamma_A) \\ \sin(\mu_A)\sin(\gamma_A)\cos(\chi_A) - \cos(\mu_A)\sin(\chi_A) & \sin(\mu_A)\sin(\gamma_A)\sin(\chi_A) + \cos(\mu_A)\cos(\chi_A) & \sin(\mu_A)\cos(\gamma_A) \\ \cos(\mu_A)\sin(\gamma_A)\cos(\chi_A) + \sin(\mu_A)\sin(\chi_A) & \cos(\mu_A)\sin(\gamma_A)\sin(\chi_A) - \sin(\mu_A)\cos(\chi_A) & \cos(\mu_A)\cos(\gamma_A) \end{bmatrix} \tag{A.3}$$

A frame to B frame:

$$\mathbf{M}_{BA} = \begin{bmatrix} \cos(\beta_A)\cdot\cos(\alpha_A) & -\sin(\beta_A)\cdot\cos(\alpha_A) & -\sin(\alpha_A) \\ \sin(\beta_A) & \cos(\beta_A) & 0 \\ \cos(\beta_A)\cdot\sin(\alpha_A) & -\sin(\beta_A)\cdot\sin(\alpha_A) & \cos(\alpha_A) \end{bmatrix} \tag{A.4}$$

$\bar{K}$ frame to B frame:

$$\mathbf{M}_{B\bar{K}} = \begin{bmatrix} \cos(\beta_K)\cdot\cos(\alpha_K) & -\sin(\beta_K)\cdot\cos(\alpha_K) & -\sin(\alpha_K) \\ \sin(\beta_K) & \cos(\beta_A) & 0 \\ \cos(\beta_K)\cdot\sin(\alpha_K) & -\sin(\beta_K)\cdot\sin(\alpha_K) & \cos(\alpha_K) \end{bmatrix} \tag{A.5}$$

K frame to $\bar{K}$ frame:

$$\mathbf{M}_{\bar{K}K} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\mu_K) & \sin(\mu_K) \\ 0 & -\sin(\mu_K) & \cos(\mu_K) \end{bmatrix} \tag{A.6}$$

A frame to K frame:

$$\mathbf{M}_{KA} = \mathbf{M}_{\bar{K}K}^{\mathsf{T}} \cdot \mathbf{M}_{B\bar{K}}^{\mathsf{T}} \cdot \mathbf{M}_{BA} \tag{A.7}$$

B frame to K frame:

$$\mathbf{M}_{KB} = \mathbf{M}_{KA} \cdot \mathbf{M}_{AB} \tag{A.8}$$

Appendix B

Continuous Probability Theory

This appendix deals with a basic introduction to continuous probability theory, as required for this thesis. More comprehensive reviews and detailed introductions are given in [141, ch. 2], [115], [12]. These are also the basis for the following review.

B.1 Random Variables

Generally speaking, random variables (RVs) are used to assign a numeric value to each random outcome of an event. More precisely, the RV is a real-valued function that assigns a numeric value to each random output of the event ω that lies within the abstract random space Ω. It should be noted that an abstract random space does not necessarily need to be numeric, but can also be defined by e.g., words or colors.

Then, all relevant subsets of the random space Ω are collected in a random class $\mathcal{F}$ that is called σ-algebra. This yields the following definition for $\mathcal{F}$ [141, p. 10]:

Definition B.1 (Random Class)**.** *A σ-field $\mathcal{F}$ on Ω is a collection of subsets of Ω satisfying the following three necessary conditions:*

- *$\mathcal{F}$ is not empty, i.e., $\emptyset \in \mathcal{F}$ and $\Omega \in \mathcal{F}$*
- *$\mathcal{F}$ is closed under complementation, i.e., if $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$*
- *$\mathcal{F}$ is closed under countable unions, i.e., $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$*

With the De-Morgan theorem *[5, p. 41] it follows that the last condition is also valid for intersections, i.e., $\bigcap_{i=1}^{\infty} A_i \in \mathcal{F}$.*

B.2 Probability

Generally speaking, probability is a concept used to quantify the likelihood of occurrence for particular events. It is defined by a probability space as follows [141, p. 11]:

Definition B.2 (Probability Space)**.** *A probability space is a triplet* $(\Omega, \mathcal{F}, \mathbb{P})$*, where* Ω *is the countable event space (also: random space),* $\mathcal{F} \subset 2^{\Omega}$ *("power set") is a* σ*-field of* Ω*, and* $\mathbb{P}$ *is a probability measure such that:*

1. $0 \leq \mathbb{P}[A] \leq 1, \qquad \forall A \in \mathcal{F}$
2. $\mathbb{P}[\Omega] = 1$
3. *For* $A_1, A_2, \ldots \in \mathcal{F}$ *and* $A_i \cap A_j = \emptyset, \forall i \neq j$

$$\mathbb{P}\left[\bigcup_{i=1}^{\infty} A_i\right] = \sum_{i=1}^{\infty} \mathbb{P}[A_i]$$

We can use the probability space in Definition B.2, to define the collection of all probabilities as the distribution function [141, p. 11]:

Definition B.3 (Distribution Function)**.** *The collection of probabilities*

$$F_{\Theta}(\theta) = \mathbb{P}[\Theta \leq \theta] = \mathbb{P}[\omega : \Theta(\omega) \leq \theta], \quad \theta \in \mathbb{R}$$

is the distribution function F_{Θ} *of* Θ*.*

Now, the term DISTRIBUTION can be defined [141, p. 11]:

Definition B.4 (Distribution)**.** *The collection of the probabilities*

$$\mathbb{P}_{\Theta}(\mathcal{B}) = \mathbb{P}[\Theta \in \mathcal{B}] = \mathbb{P}[\{\omega : \Theta(\omega) \in \mathcal{B}\}]$$

for suitable subsets $\mathcal{B} \subset \mathbb{R}$ *(called* BOREL SETS *[5, p. 1297]) is the distribution of* Θ*.*

B.2.1 Continuous Distributions

In general, the most important property of a continuous RV is the fact that it is not exhibiting jumps. As a consequence, the probability of a particular value of the RV is always zero [141, p. 12]:

$$\mathbb{P}\left[\Theta = \theta\right] \equiv 0, \qquad \forall \theta \in \mathbb{R} \tag{B.1}$$

If possible, the RV of a continuous distribution is defined by its PDF $\rho_\Theta(\theta)$ as follows [141, p. 12]:

$$F_\Theta(\theta) = \int_{-\infty}^{x} \rho_\Theta(\theta)\, \mathrm{d}\theta, \qquad \theta \in \mathbb{R} \tag{B.2}$$

This is also known as the cumulative distribution function.

The PDF then has the helpful property that the integral with respect to the real numbers is *one* [141, p. 12]:

$$\int_{-\infty}^{\infty} \rho_\Theta(\theta)\, \mathrm{d}\theta \equiv 1, \quad \rho_\Theta(\theta) \geq 0, \quad \forall \theta \in \mathbb{R} \tag{B.3}$$

B.2.2 Statistical Moments

Most of the important characteristics of a RV can be described in terms of statistical moments. Generally, the m-th statistical moment of a continuous RV, Θ, is defined as follows [141, p. 13f.]:

$$\mathbb{E}\left[\Theta^m\right] = \int_{-\infty}^{\infty} \theta^m \rho_\Theta(\theta)\, \mathrm{d}\theta, \qquad m \in \mathbb{N} = \{1, 2, \ldots, \infty\} \tag{B.4}$$

The first moment is called the mean value and defined by:

$$\mu_\Theta = \mu\left[\Theta\right] = \mathbb{E}\left[\Theta\right] = \int_{-\infty}^{\infty} \theta \rho_\Theta(\theta)\, \mathrm{d}\theta \tag{B.5}$$

The second moment is generally normalized (also called: centered) with respect to the mean value and called variance:

$$\sigma_\Theta^2 = \sigma^2\left[\Theta\right] = \mathrm{var}\left[\Theta\right] = \int_{-\infty}^{\infty} \left(\theta - \mu_\Theta\right)^2 \rho_\Theta(\theta)\, \mathrm{d}\theta \tag{B.6}$$

The square root of the variance is called standard deviation $\sigma_\Theta = \sigma\left[\Theta\right]$ and is normally used to describe a RV instead of the variance. The standard deviation describes the spreading of the data around the center given by the mean value.

The skewness describes the third centered moment and is a measure of the symmetry of the PDF. It is defined as follows [2, p. 928]:

$$\mathrm{skew}\left[\Theta\right] = \mathbb{E}\left[\left(\frac{\Theta - \mu_\Theta}{\sigma_\Theta}\right)^3\right] = \frac{\mathbb{E}\left[\Theta^3\right] - 3\mu_\Theta\sigma_\Theta^2 - \mu_\Theta^3}{\sigma_\Theta^3} \tag{B.7}$$

Here, (B.4) can be used to calculate the third moment of the RV.

Finally, another commonly used statistical moment is the kurtosis that is defined to be the fourth centered moment and is a measure of the tailedness of the PDF [2, p. 928]:

$$\operatorname{kurt}[\Theta] = \mathbb{E}\left[\left(\frac{\Theta - \mu_\Theta}{\sigma_\Theta}\right)^4\right] = \frac{\mathbb{E}[\Theta^4] - 4\mu_\Theta \mathbb{E}[\Theta^3] + 6\mu_\Theta^2\sigma_\Theta^2 + 3\mu_\Theta^4}{\sigma_\Theta^4} \tag{B.8}$$

Once more, (B.4) can be used to calculate the fourth moment of the RV.

In addition to the moments related to a single continuous RV, one can also define the covariance of two RVs, Θ_1 and Θ_2, as follows [5, p. 1318]:

$$\begin{aligned}\operatorname{cov}[\Theta_1, \Theta_2] &= \mathbb{E}[(\Theta_1 - \mathbb{E}[\Theta_1]) \cdot (\Theta_2 - \mathbb{E}[\Theta_2])] \\ &= \mathbb{E}[\Theta_1 \cdot \Theta_2] - \mathbb{E}[\Theta_1]\,\mathbb{E}[\Theta_2]\end{aligned} \tag{B.9}$$

Generally, the covariance can be regarded as a measure of how Θ_1 behaves/changes when Θ_2 changes and vice versa. This can also be viewed as a measure of correlation.

B.3 Random Vectors

A n_θ-dimensional vector $\boldsymbol{\Theta} = \begin{bmatrix}\Theta_1 & \dots & \Theta_{n_\theta}\end{bmatrix}^\mathsf{T}$ is called random vector (i.e., a vector of one-dimensional random variables) (RVec), if the components $\Theta_1, \dots, \Theta_{n_\theta}$ are one-dimensional and real-valued RVs. Therefore, a RVec is merely a collection of RVs and the concepts such as distribution functions (Definition B.3) can be transformed from the scalar RV as follows [141, p. 17]:

Definition B.5 (Multi-variate Distribution Function)**.** *The collection of the probabilities*

$$F_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \mathbb{P}[\Theta_1 \leq \theta_1, \dots, \Theta_{n_\theta} \leq \theta_{n_\theta}], \qquad \boldsymbol{\theta} = \begin{bmatrix}\theta_1 & \dots & \theta_{n_\theta}\end{bmatrix}^\mathsf{T} \in \mathbb{R}^{n_\theta},$$

is the distribution function $F_{\boldsymbol{\Theta}}$ of $\boldsymbol{\Theta}$.

With Definition B.5, the distribution function can also be represented by means of PDFs:

$$F_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \int_{-\infty}^{\theta_1} \dots \int_{-\infty}^{\theta_{n_\theta}} \rho_{\boldsymbol{\Theta}}(\theta_1, \dots, \theta_{n_\theta})\, \mathrm{d}\theta_1 \dots \mathrm{d}\theta_{n_\theta}, \qquad \rho_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) \geq 0, \quad \forall \boldsymbol{\theta} \in \mathbb{R}^{n_\theta} \tag{B.10}$$

B.4 Stochastic Process

Stochastic processes are used to describe the development of a RV, and therefore the randomness of the system, as a function of space and/or time. A stochastic process is defined by [141, p. 20f.]:

Definition B.6 (Stochastic Process)**.** *A stochastic process is a collection of RVs*

$$(\Theta, t \in T) = (\Theta_t(\omega), t \in T, \omega \in \Omega)$$

defined in a random space, Ω*, with* t *the time/spatial index of the RV. The time interval,* T*, is some continuous interval, defined by* $[a, b]$ *with* $a < b$*.*

B.5 Dependence and Conditional Expectation

Generally speaking, independent random events are present, if the outcome of one event does not influence the outcome of the other one. RVs are independent, if the following definition holds [141, p. 18]:

Definition B.7 (Independent Random Variables)**.** *Two RVs* Θ_1 *and* Θ_2 *are independent if*

$$\mathbb{P}[\Theta_1 \in \mathcal{B}_1, \Theta_2 \in \mathcal{B}_2] = \mathbb{P}[\Theta_1 \in \mathcal{B}_1]\,\mathbb{P}[\Theta_2 \in \mathcal{B}_2]$$

for all suitable subsets $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *in* $\mathbb{R}$*. This means that the events* $\{\Theta_1 \in \mathcal{B}_1\}$ *and* $\{\Theta_2 \in \mathcal{B}_2\}$ *are independent.*

A more useful definition for this work is given by using the PDFs [141, p. 18f.]:

Definition B.8 (Independent Random Variables via Probability Density Function)**.** *The RVs* $\Theta_1, \ldots, \Theta_n$ *are independent if, and only if, their densities can be written as:*

$$\rho_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \rho_{\Theta_1,\ldots,\Theta_{n_\theta}}(\theta_1, \ldots, \theta_{n_\theta}) = \rho_{\Theta_1}(\theta_1) \cdot \ldots \cdot \rho_{\Theta_{n_\theta}}(\theta_{n_\theta})$$

Definition B.8 directly yields that the expectations are independent as well, which in turn yields the fact that the covariance of two RVs is *zero* (see (B.9)). The consequence of this is that two independent RVs are also uncorrelated. Take into account that the opposite statement is, in general, not true.

B.6 Modes of Convergence

Modes of convergence are an important property of a statistical method, as they define how fast the method approaches the desired statistical distribution. Therefore, at first the convergence in distribution and probability is defined [141, p. 22]:

Definition B.9 (Convergence in Distribution)**.** *The sequence* $\{\Theta_n\}$ *converges in distribution (or converges weakly) to the RV* Θ*, written as* $\Theta_n \xrightarrow{d} \Theta$*, if for all bounded and continuous functions* f*,*

$$\mathbb{E}\left[f\left(\Theta_n\right)\right] \to \mathbb{E}\left[f\left(\Theta\right)\right], \qquad n \to \infty$$

Furthermore, convergence in probability is defined as follows:

Definition B.10 (Convergence in Probability)**.** *The sequence* $\{\Theta_n\}$ *converges in probability to the RV* Θ*, written as* $\Theta_n \xrightarrow{\mathbb{P}} \Theta$*, if for all positive* ϵ*,*

$$\mathbb{P}\left[|\Theta_n - \Theta| > \epsilon\right] \to 0, \qquad n \to \infty$$

It should be noted that convergence in probability implies convergence in distribution [141, p. 22].

Then, the concept of almost sure convergence can be defined [141, p. 22]:

Definition B.11 (Almost Sure Convergence)**.** *The sequence* $\{\Theta_n\}$ *converges almost surely (a.s.), or with probability* 1*, to the RV* Θ*, written as* $\Theta_n \xrightarrow{a.s.} \Theta$*, if the set of* ω *with*

$$\Theta_n\left(\omega\right) \to \Theta\left(\omega\right), \qquad n \to \infty$$

has probability 1*.*

It should be noted that almost sure convergence implies convergence in probability and convergence in distribution [141, p. 22].

Finally, L^p convergence, which is the type of convergence the gPC method adheres, is defined as follows [141, p. 23]:

Definition B.12 (L^p Convergence)**.** *Let* $p > 0$*. Then, the sequence* $\{\Theta_n\}$ *converges in* L^p*, or in the p-th mean, to* Θ*, written as* $\Theta_n \xrightarrow{L^p} \Theta$ *if*

$$\mathbb{E}\left[|\Theta_n|^p + |\Theta|^p\right] < \infty, \qquad n \to \infty$$

and

$$\mathbb{E}\left[|\Theta_n - \Theta|^p\right] \to 0, \qquad n \to \infty$$

Take into account that L^P convergence is considered a form of *strong* convergence [141, p. 23].

B.7 Central Limit Theorem

The CLT is one of the most important theorems in probability analysis. Here, the definition after [141, p. 23] is used:

Theorem B.13 (Central Limit Theorem). *Let $\Theta_1, \ldots, \Theta_{n_\theta}$ be independent and identically distributed RVs with $\mathbb{E}[\Theta_i] = \mu_\Theta$ and $\mathrm{var}[\Theta_i] = \sigma_\Theta^2 < \infty$. Let*

$$\overline{\Theta} = \frac{1}{n_\theta} \sum_{i=1}^{n_\theta} \Theta_i$$

and

$$U_n = \sqrt{n} \left(\frac{\overline{\Theta} - \mu_\Theta}{\sigma_\Theta} \right)$$

Then, the PDF of U_n converges to a $\mathcal{N}(0,1)$, i.e., the standard Gaussian *PDF, as $n \to \infty$.*

It should be noted that Theorem B.13 immediately implies that any set of independent and identically distributed RVs converges to a Gaussian PDF defined by $\mathcal{N}(\mu_\Theta, \sigma_\Theta^2 \mu_\Theta)$. Here, μ_Θ and σ_Θ^2 are then representing the mean and variance of the independent and identically distributed RVs.

B.8 Probability Inequalities

The Chebyshev's inequality is a very general and powerful tool in statistical analysis hinting at the number of samples that are in a certain distance to the mean value. The only knowledge that must be applied is the mean value and the standard deviation. Take into account that this generally requires no additional statement of the shape of the PDF. Thus, these bounds cannot be improved on for arbitrary PDFs. Thus, it returns a conservative bound and is therefore very useful to calculate ROC solutions, as unmodeled uncertainties are also partially covered. The inequality is defined as follows [5, p. 1312]:

Theorem B.14 (Chebyshev's Inequality). *Let Θ be an integrable RV with finite mean value and non-zero standard deviation. Then for any real number $k > 1$ it holds that:*

$$\mathbb{P}[|\Theta - \mu_\Theta| \geq k\sigma_\Theta] \leq \frac{1}{k^2}$$

Thus, Theorem B.14 states that a worst-case probability of exceeding the interval $k\sigma_\Theta$ around the mean value can be defined, without requiring any knowledge on how the PDF looks like. This is specifically powerful in combination with the gPC formulation, as it provides a very good approximation of mean and standard deviation rather cheap, but the description of the PDF that results is rather difficult to obtain. It should be noted that Theorem B.14 also holds for values of $0 < k \leq 1$, but the results become trivial for this case as the probability is then $\mathbb{P} \geq 1$.

Take into account that the approach in Theorem B.14 is always conservative and therefore, also beneficial for ROC. This can be shown by looking at the following example: It is known that approximately 99.7% of the samples of a standard Gaussian PDF are within a 3σ interval around the mean value. Evaluating Theorem B.14 with $k = 3$ gives a value of $\approx 89\%$. This shows that Theorem B.14 is indeed fairly conservative, which is beneficial in order to calculate robust trajectories as there are unmodeled dynamics and additional uncertainties in reality that need to be overcome.
As a refinement of the Chebyshev's inequality in Theorem B.14, the Vysochanskij–Petunin inequality is available: Here, an additional assumption is required, which is that the PDF must be unimodal, i.e., it changes at only exactly one point in the random parameter space from convex to concave or vice versa. The inequality is given as follows [2, p. 931]:

Theorem B.15 (Vysochanskij–Petunin inequality)**.** *Let Θ be an integrable RV with finite mean value and non-zero standard deviation. Furthermore, let the RV be distributed unimodal. Then for any real number $k > \sqrt{\frac{8}{3}}$ it holds that:*

$$\mathbb{P}\left[|\Theta - \mu_\Theta| \geq k\sigma_\Theta\right] \leq \frac{4}{9k^2}$$

Theorem B.15 gives a tighter bound on the probability, but it also adds a further assumption (unimodality) that might not always be fulfilled. For instance, using $k = 3$ in Theorem B.15 with the standard Gaussian definition for mean and standard deviation gives a probability of $\approx 95.1\%$ to be in the interval. This is in the middle between the Chebyshev's inequality approximation and the exact Gaussian PDF. Take into account that the constraint $k > \sqrt{\frac{8}{3}}$ is required as some non-symmetric PDFs can be shown to exceed the probability given by Theorem B.15 for smaller values. As this bound corresponds to a probability of around 94% this is normally not problematic for the implementation of e.g., CCs as a higher probability should be achieved anyways.

Appendix C

Orthogonal Polynomials

This appendix gives an overview on important properties of orthogonal polynomials that are required within the gPC expansion. More comprehensive reviews are also given in [141, ch. 3], [2, ch. 22], or [27].

At first, the term *orthogonal polynomial* is defined [2, p. 773f.]:

Theorem C.1 (Favard's Theorem)**.** *Let $A^{(m)}$, $B^{(m)}$, and $C^{(m)}$ be arbitrary sequences of real numbers and let the set $\left\{\phi^{(m)}(\theta)\right\}$ be defined by the recurrence relation*

$$\phi^{(m+1)}(\theta) = \left[A^{(m)}\theta + B^{(m)}\right]\phi^{(m)}(\theta) - C^{(m)}\phi^{(m-1)}(\theta), \qquad m \geq 0$$

together with $\phi^{(0)}(\theta) = 1$ and $\phi^{(-1)}(\theta) = 0$.
Then, the polynomials $\left\{\phi^{(m)}(\theta)\right\}$, defined by the recurrence algorithm, are a system of orthogonal polynomials if, and only if, $A^{(m)} \neq 0$, $C^{(m)} \neq 0$, and $C^{(m)}A^{(m)}A^{(m-1)} > 0$ for all m.

Now, some examples of orthogonal polynomials, which are outstanding for this thesis are looked at. Therefore, at first the Legendre polynomials are defined [5, p.1182]:

Definition C.2 (Legendre Polynomials)**.** *Legendre polynomials satisfy the recurrence sequence*

$$\phi^{(m+1)}(\theta) = \frac{2m+1}{m+1}\theta\phi^{(m)}(\theta) - \frac{m}{n+1}\phi^{(m-1)}(\theta), \qquad m \geq 0$$

and the orthogonality relation

$$\int_{-1}^{1} \phi^{(n)}(\theta)\,\phi^{(m)}(\theta)\,1\,\mathrm{d}\theta = \frac{2}{2m+1}\delta_{mn}.$$

Thus, they are orthogonal with respect to the PDF $\rho_\Theta(\theta) = 1$ and have the normalizing constant $\left[h^{(m)}\right]^2 = \frac{2}{2m+1}$.

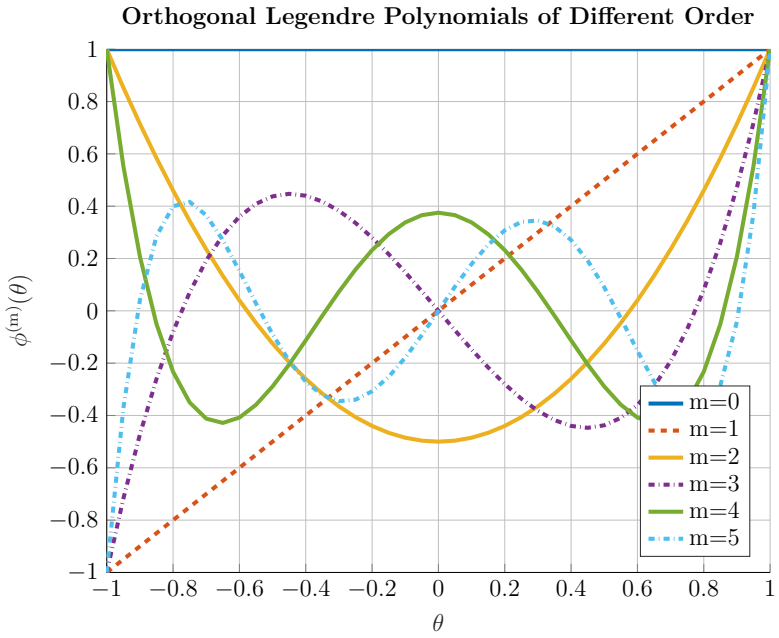

Figure C.1: *First six non-normalized orthogonal Legendre polynomials as defined in the thesis.*

Definition C.2 shows that Legendre polynomials are orthogonal with respect to the PDF of the UNIFORM distribution.

The first four non-normalized Legendre polynomials are:

$$\phi^{(0)}(\theta) = 1, \quad \phi^{(1)}(\theta) = \theta, \quad \phi^{(2)}(\theta) = \frac{3}{2}\theta^2 - \frac{1}{2}, \quad \phi^{(3)}(\theta) = \frac{5}{2}\theta^3 - \frac{3}{2}\theta \tag{C.1}$$

A visualization of the first six non-normalized Legendre polynomials is also given in Figure C.1.

Then, Hermite polynomials that are orthogonal with respect to the PDF of the GAUSSIAN distribution are defined [5, p.1186]:

Definition C.3 (Hermite Polynomials)**.** *Hermite polynomials satisfy the recurrence sequence*

$$\phi^{(m+1)}(\theta) = \theta\phi^{(m)}(\theta) - m\phi^{(m-1)}(\theta), \qquad m \geq 0$$

and the orthogonality relation

$$\int_{-\infty}^{\infty} \phi^{(n)}(\theta)\,\phi^{(m)}(\theta)\,\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\theta^2}{2}\right)\mathrm{d}\theta = m!\delta_{mn}\,.$$

Thus, they are orthogonal with respect to the PDF $\rho_\Theta(\theta) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\theta^2}{2}\right)$ *and have the normalizing constant* $\left[h^{(m)}\right]^2 = m!$.

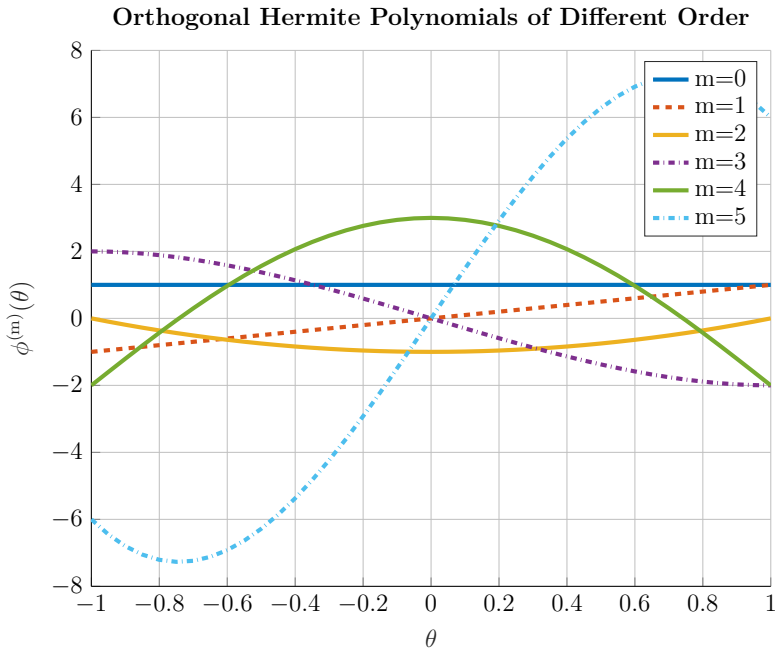

Figure C.2: *First six non-normalized orthogonal Hermite polynomials as defined in the thesis.*

The first four non-normalized Hermite polynomials in their functional form are:

$$\phi^{(0)}(\theta) = 1, \quad \phi^{(1)}(\theta) = \theta, \quad \phi^{(2)}(\theta) = \theta^2 - 1, \quad \phi^{(3)}(\theta) = \theta^3 - 3\theta \tag{C.2}$$

A visualization of the first six non-normalized Hermite polynomials is given in Figure C.2.

Another important group of orthogonal polynomials for the gPC expansion are the Laguerre polynomials [5, p. 1186]:

Definition C.4 (Laguerre Polynomials)**.** *Laguerre polynomials satisfy the recurrence sequence*

$$\begin{aligned}(m+1)\,\phi^{(m+1)}(\theta,\alpha) &= (-\theta + 2m + \alpha + 1)\,\phi^{(m)}(\theta,\alpha) \\ &\quad - (m+\alpha)\,\phi^{(m-1)}(\theta,\alpha)\,, \qquad m \geq 0 \text{ and } \alpha > -1\end{aligned}$$

and the orthogonality relation

$$\int_0^\infty \phi^{(n)}(\theta,\alpha)\,\phi^{(m)}(\theta,\alpha)\exp(-\theta)\,\theta^\alpha\,\mathrm{d}\theta = \frac{\Gamma(m+\alpha+1)}{m!}\delta_{mn}\,, \qquad \alpha > -1.$$

Thus, they are orthogonal with respect to the PDF $\rho_\Theta(\theta) = \exp(-\theta)\,\theta^\alpha$ *and have the normalizing constant* $\left[h^{(m)}\right]^2 = \frac{\Gamma(m+\alpha+1)}{m!}$.

It is imminent that Definition C.4 suggests that the Laguerre polynomials are orthogonal with respect to the PDF of the Gamma distribution.

The final orthogonal polynomial example, to be looked at in this work, are the Jacobi polynomials. These are orthogonal with respect to the PDF of the Beta distribution [2, p. 774]:

Definition C.5 (Jacobi Polynomials)**.** *Jacobi polynomials satisfy the recurrence sequence*

$$\begin{aligned}\theta\phi^{(m)}(\theta,\alpha,\beta) = \\ \frac{2(m+1)(m+\alpha+\beta+1)}{(2m+\alpha+\beta+1)(2m+\alpha+\beta+2)}\phi^{(m+1)}(\theta,\alpha,\beta) \\ \frac{\beta^2-\alpha^2}{(2m+\alpha+\beta)(2m+\alpha+\beta+2)}\phi^{(m)}(\theta,\alpha,\beta) \\ \frac{2(m+\alpha)(m+\beta)}{(2m+\alpha+\beta)(2m+\alpha+\beta+1)}\phi^{(m-1)}(\theta,\alpha,\beta), \\ m\geq 0 \text{ and } \alpha,\beta>-1\end{aligned}$$

and the orthogonality relation

$$\begin{aligned}\int_{-1}^{1}\phi^{(n)}(\theta,\alpha,\beta)\,\phi^{(m)}(\theta,\alpha,\beta)\,\frac{\Gamma(\alpha+\beta+2)}{2^{\alpha+\beta+1}\Gamma(\alpha+1)\Gamma(\beta+1)}(1-\theta)^{\alpha}(1+\theta)^{\beta}\,\mathrm{d}\theta \\ =\frac{(\alpha+1)_m(\beta+1)_m}{m!(2m+\alpha+\beta+1)(\alpha+\beta+2)_{m-1}}\delta_{mn}, \qquad \alpha,\beta>-1.\end{aligned}$$

Thus, they are orthogonal with respect to the PDF $\rho_\Theta(\theta) = \frac{\Gamma(\alpha+\beta+2)}{2^{\alpha+\beta+1}\Gamma(\alpha+1)\Gamma(\beta+1)}(1-\theta)^{\alpha}(1+\theta)^{\beta}$ *and have the normalizing constant* $\left[h^{(m)}\right]^2 = \frac{(\alpha+1)_m(\beta+1)_m}{m!(2m+\alpha+\beta+1)(\alpha+\beta+2)_{m-1}}$.

In general, polynomials (or in this work: orthogonal polynomials) are able to uniformly approximate any continuous function on a bounded interval. This result is summarized within the following theorem [5, p. 1071]:

Theorem C.6 (Weierstrass Approximation Theorem)**.** *Let f be a continuous real-valued function defined on the real interval $[a,b]$. Now, for every $\epsilon>0$, there exist a polynomial ϕ such that for all θ in $[a,b]$, it is*

$$|f(\theta)-\phi(\theta)|<\epsilon$$

Appendix D

Derivation of Statistical Moments for Generalized Polynomial Chaos Expansion

This appendix summarizes the derivation of statistical information for the gPC expansion (Subsection 2.3.4). It is assumed that the expansion coefficients are known, i.e., the discrete expansion of gPC is used (Subsection 2.3.3).

The mean value is calculated as follows:

$$\begin{aligned}
\mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] \approx \mathbb{E}\left[\mathbf{y}_N^{(D)}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] &= \int_{\boldsymbol{\Omega}} \left(\sum_{m=0}^{M-1} \hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right) \Phi^{(m)}\left(\boldsymbol{\theta}\right) \right) \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right) \, \mathrm{d}\boldsymbol{\theta} = \\
&= \sum_{m=0}^{M-1} \hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right) \int_{\boldsymbol{\Omega}} \Phi^{(m)}\left(\boldsymbol{\theta}\right) \underbrace{\Phi^{(0)}\left(\boldsymbol{\theta}\right)}_{\equiv 1} \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right) \, \mathrm{d}\boldsymbol{\theta} = \\
&= \hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right) \int_{\boldsymbol{\Omega}} \Phi^{(0)}\left(\boldsymbol{\theta}\right) \Phi^{(0)}\left(\boldsymbol{\theta}\right) \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right) \, \mathrm{d}\boldsymbol{\theta} = \\
&= \hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right)
\end{aligned} \tag{D.1}$$

The variance is calculated in a similar manner:

$$\begin{aligned}
\operatorname{var}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] &= \mathbb{E}\left[\left(\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right) - \mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\right)^2\right] = \\
&= \mathbb{E}\left[\left(\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right)^2\right] - \mathbb{E}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right]^2 \approx \\
&\approx \int_{\boldsymbol{\Omega}} \left(\sum_{m=0}^{M-1} \hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right) \Phi^{(m)}\left(\boldsymbol{\theta}\right)\right)^2 \boldsymbol{\rho}\left(\boldsymbol{\theta}\right)\, \mathrm{d}\boldsymbol{\theta} - \left[\hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right)\right]^2 = \\
&= \int_{\boldsymbol{\Omega}} \sum_{m=0}^{M-1} \left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2 \left[\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right]^2 \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\, \mathrm{d}\boldsymbol{\theta} + \\
&+ \underbrace{\sum_{m\neq n} \left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2 \int_{\boldsymbol{\Omega}} \Phi^{(m)}\left(\boldsymbol{\theta}\right) \Phi^{(n)}\left(\boldsymbol{\theta}\right) \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\, \mathrm{d}\boldsymbol{\theta}}_{0,\, m\neq n} - \left[\hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right)\right]^2 = \\
&= \sum_{m=0}^{M-1} \left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2 \underbrace{\int_{\boldsymbol{\Omega}} \left[\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right]^2 \rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\, \mathrm{d}\boldsymbol{\theta}}_{\text{orthonormal, i.e., } =1} - \left[\hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right)\right]^2 = \\
&= \sum_{m=1}^{M-1} \left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2 + \left[\hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right)\right]^2 - \left[\hat{\mathbf{y}}^{(0)}\left(\mathbf{z}\right)\right]^2 = \\
&= \sum_{m=1}^{M-1} \left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2
\end{aligned} \tag{D.2}$$

The derivation of the formulas for covariance, skewness, and kurtosis is equally possible with the properties of the orthogonal polynomials and the mathematical expansion formulas for the corresponding definitions for the quadrature of summation formulas.

At first, the calculation of the covariance of two measurements, i.e., a measure for their correlation, is given as follows [141, p. 254]:

$$\begin{aligned}
\operatorname{cov}\left[y_1, y_2\right] &= \mathbb{E}\left[\left(y_1\left(\mathbf{z};\boldsymbol{\theta}\right) - \mathbb{E}\left[y_1\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\right)\left(y_2\left(\mathbf{z};\boldsymbol{\theta}\right) - \mathbb{E}\left[y_2\left(\mathbf{z};\boldsymbol{\theta}\right)\right]\right)\right] = \\
&= \mathbb{E}\left[y_1\left(\mathbf{z};\boldsymbol{\theta}\right) y_2\left(\mathbf{z};\boldsymbol{\theta}\right)\right] - \mathbb{E}\left[y_1\left(\mathbf{z};\boldsymbol{\theta}\right)\right] \mathbb{E}\left[y_2\left(\mathbf{z};\boldsymbol{\theta}\right)\right] = \\
&\approx \sum_{m=1}^{M-1} \hat{\mathbf{y}}_1^{(m)}\left(\mathbf{z}\right) \hat{\mathbf{y}}_2^{(m)}\left(\mathbf{z}\right)
\end{aligned} \tag{D.3}$$

Furthermore, the skewness and kurtosis of the results can be calculated. For this, we need the third and fourth moment of the gPC expansion given by (2.61) are required. The third order can be derived as follows:

$$\begin{aligned}
\mathbb{E}\left[\left(\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right)^3\right] &\approx \mathbb{E}\left[\left(\sum_{m=0}^{M-1}\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right)^3\right] = \\
&= \sum_{m=0}^{M-1}\left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^3\int_{\boldsymbol{\Omega}}\left[\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right]^3\rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\,\mathrm{d}\boldsymbol{\theta} = \\
&+ 2\sum_{m\neq n}\left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2\hat{\mathbf{y}}^{(n)}\left(\mathbf{z}\right)\int_{\boldsymbol{\Omega}}\left[\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right]^2\Phi^{(n)}\left(\boldsymbol{\theta}\right)\rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\,\mathrm{d}\boldsymbol{\theta} = \\
&+ \sum_{\substack{m\neq n\\ m\neq k}}\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\hat{\mathbf{y}}^{(n)}\left(\mathbf{z}\right)\hat{\mathbf{y}}^{(k)}\left(\mathbf{z}\right)\int_{\boldsymbol{\Omega}}\Phi^{(m)}\left(\boldsymbol{\theta}\right)\Phi^{(n)}\left(\boldsymbol{\theta}\right)\Phi^{(k)}\left(\boldsymbol{\theta}\right)\rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\,\mathrm{d}\boldsymbol{\theta}
\end{aligned} \tag{D.4}$$

Similarly, the fourth order statistical moment is given by:

$$\begin{aligned}
\mathbb{E}\left[\left(\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right)^4\right] &\approx \mathbb{E}\left[\left(\sum_{m=0}^{M-1}\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right)^4\right] = \\
&= \sum_{m=0}^{M-1}\left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^4\int_{\boldsymbol{\Omega}}\left[\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right]^4\rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\,\mathrm{d}\boldsymbol{\theta} + \\
&+ 3\sum_{m\neq n}\left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^3\hat{\mathbf{y}}^{(n)}\left(\mathbf{z}\right)\int_{\boldsymbol{\Omega}}\left[\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right]^3\Phi^{(m)}\left(\boldsymbol{\theta}\right)\rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\,\mathrm{d}\boldsymbol{\theta} + \\
&+ 3\sum_{\substack{m\neq n\\ m\neq k}}\left[\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\right]^2\hat{\mathbf{y}}^{(n)}\left(\mathbf{z}\right)\hat{\mathbf{y}}^{(k)}\left(\mathbf{z}\right)\cdot \\
&\cdot\int_{\boldsymbol{\Omega}}\left[\Phi^{(m)}\left(\boldsymbol{\theta}\right)\right]^2\Phi^{(n)}\left(\boldsymbol{\theta}\right)\Phi^{(k)}\left(\boldsymbol{\theta}\right)\rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\,\mathrm{d}\boldsymbol{\theta} + \\
&+ \sum_{\substack{m\neq n\\ m\neq k\\ m\neq l}}\hat{\mathbf{y}}^{(m)}\left(\mathbf{z}\right)\hat{\mathbf{y}}^{(n)}\left(\mathbf{z}\right)\hat{\mathbf{y}}^{(k)}\left(\mathbf{z}\right)\hat{\mathbf{y}}^{(l)}\left(\mathbf{z}\right)\cdot \\
&\cdot\int_{\boldsymbol{\Omega}}\Phi^{(m)}\left(\boldsymbol{\theta}\right)\Phi^{(n)}\left(\boldsymbol{\theta}\right)\Phi^{(k)}\left(\boldsymbol{\theta}\right)\Phi^{(l)}\left(\boldsymbol{\theta}\right)\rho_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\,\mathrm{d}\boldsymbol{\theta}
\end{aligned} \tag{D.5}$$

With (D.4) and (D.5), the skewness and kurtosis cab then be calculated by their well-known formulas as follows (Subsection B.2.2):

$$\operatorname{skew}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] \approx \mathbb{E}\left[\left(\frac{\mathbf{y}_N^{(D)}\left(\mathbf{z};\boldsymbol{\theta}\right)-\mu_\Theta}{\sigma_\Theta}\right)^3\right] = \frac{\mathbb{E}\left[\left(\mathbf{y}_N^{(D)}\left(\mathbf{z};\boldsymbol{\theta}\right)\right)^3\right] - 3\mu_\Theta\sigma_\Theta^3 - \mu_\Theta^3}{\sigma_\Theta^3} \tag{D.6}$$

$$\begin{aligned}
\operatorname{kurt}\left[\mathbf{y}\left(\mathbf{z};\boldsymbol{\theta}\right)\right] &\approx \mathbb{E}\left[\left(\frac{\mathbf{y}_N^{(D)}\left(\mathbf{z};\boldsymbol{\theta}\right)-\mu_\Theta}{\sigma_\Theta}\right)^4\right] = \\
&= \frac{\mathbb{E}\left[\left(\mathbf{y}_N^{(D)}\left(\mathbf{z};\boldsymbol{\theta}\right)\right)^4\right] - 4\mu_\Theta\mathbb{E}\left[\left(\mathbf{y}_N^{(D)}\left(\mathbf{z};\boldsymbol{\theta}\right)\right)^3\right] + 6\mu_\Theta^2\sigma_\Theta^2 + 3\mu_\Theta^4}{\sigma_\Theta^4}
\end{aligned} \tag{D.7}$$

Appendix E

Statistical Moments of PT1 Step Response

This appendix summarizes the calculation of the semi-analytic solution of the variance for the step response of a first order lag (PT1 element) with an uncertain time constant T (Subsection 2.3.6).

For this, the general definition of the exponential integral allowing also complex values is required [2, p. 228]:

$$\mathrm{E}_1(t) = \int_{\tau=t}^{\infty} \frac{\exp(-\tau)}{\tau}\,\mathrm{d}\tau, \qquad |\arg(t)| < \pi \tag{E.1}$$

The standard exponential integral is defined as given in (2.76).

As a reminder the mean value is calculated as follows:

$$\mathbb{E}[y] = \int_{\Omega} \left[1 - \exp\left(\frac{t}{T}\right)\right] \cdot \underbrace{\rho_{\Theta}(\theta)}_{1}\,\mathrm{d}\theta \tag{E.2}$$

Evaluating the integral results in (2.75):

$$\mathbb{E}[y] = \left[\mathrm{Ei}(-2t) - \mathrm{Ei}\left(-\frac{2t}{3}\right)\right] t + \frac{\exp(-2t)}{2} - \left[1.5 \cdot \exp\left(-\frac{2t}{3}\right)\right] + 1 \tag{E.3}$$

Then, a similar evaluation can be done for the variance using the following formula:

$$\mathrm{var}[y] = \int_{\Omega} \left[1 - \exp\left(\frac{t}{T}\right)\right]^2 \cdot \underbrace{\rho_{\Theta}(\theta)}_{1}\,\mathrm{d}\theta - \mathbb{E}[y]^2 \tag{E.4}$$

This results in the following semi-analytic solution, using the two introduced definitions of the exponential integral as follows:

$$
\begin{aligned}
\operatorname{var}[y] = & \left\{ -\left[\operatorname{Ei}(-2t) - \operatorname{Ei}\left(-\frac{2t}{3}\right) \right]^2 \right\} t^2 \\
& + \left\{ 2\mathrm{E}_1(4t) - 2\mathrm{E}_1(2t) + 2\mathrm{E}_1\left(\frac{2t}{3}\right) - 2\mathrm{E}_1\left(\frac{4t}{3}\right) \right. \\
& \left. -2\left[\operatorname{Ei}(-2t) - \operatorname{Ei}\left(-\frac{2t}{3}\right) \right] \left[\frac{\exp(-2t)}{2} - \frac{3\exp\left(-\frac{2t}{3}\right)}{2} + 1 \right] \right\} t \\
& + \exp(-2t) - \frac{\exp(-4t)}{2} - 3\exp\left(-\frac{2t}{3}\right) \\
& + \frac{3\exp\left(-\frac{4t}{3}\right)}{2} - \left[\frac{\exp(-2t)}{2} - \frac{3\exp\left(-\frac{2t}{3}\right)}{2} + 1 \right]^2 + 1
\end{aligned}
\tag{E.5}
$$

The standard deviation can then be calculated by taking the square root of (E.5).

Thus, (E.2) and (E.5) provide the desired semi-analytic solution of the PT1 step response that are used for the accuracy and convergence estimation of the gPC expansion.