BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NONLINEAR OPTIMAL CONTROL SYSTEM SYNTHESIS FOR DYNAMICALLY COUPLED AUTONOMOUS AERIAL VEHICLES

Ph.D.THESIS

AUTHOR Ing. JIŘÍ NOVÁK

SUPERVISOR doc. Ing. PETER CHUDÝ, Ph.D., MBA

BRNO 2024

Project 8A19003, 826610 "Framework of key enabling technologies for safe and autonomous drones´ applications" has been supported by ECSEL Joint Undertaking.

Project TN 01000029"Artificial Intelligence Driven Autonomy" has been supported by the Technology Agency of the Czech Republic.

ABSTRACT

The popularity of state-of-the-art machine learning and optimization techniques is currently accelerated by the advancements in computational power and miniaturization of embedded computers. In many industrial applications, the use of machine learning and optimization techniques now becomes a viable and preferred option. Guidance, Navigation, and Control (GNC) techniques are critical for enabling both aerial and ground vehicle autonomy. The development of GNC algorithms primarily relies on a model-based design using the first principle approach, which leads to explainable and verifiable solutions. The increasing requirements and dynamically changing environments challenge the current state-of-the-art GNC approaches and promote the use of data-driven and adaptive systems. An associated challenge is the integration into safety-critical applications, which requires the development of protecting mechanisms as well as conducting verification and validation procedures. The goal of this thesis is to design, implement, and test a GNC system for an Unmanned Aerial Vehicle modified for aerial grasping missions while incorporating optimal control and machine learning components, enabling the creation of a robust, explainable, and safe control. A statistics-based approach utilizing surrogate modeling techniques is proposed for the system's safety validation, treating the system as a black-box. The subsequent evaluation of the system's failure probability is paramount for real-world integration endeavors requiring certification by respective regulation authorities.

KEYWORDS

Adaptive Control; Aerial Grasping; Convex Optimization; Direct Collocation Method; Model Predictive Control; Nonlinear Programming; Optimal Control; Polynomial Chaos Expansion; Surrogate Modeling; Trajectory Optimization; Unmanned Aerial Vehicles; Visual Servoing

ABSTRAKT

Popularita moderních technik strojového učení a optimalizace je v současnosti podpořena pokrokem z pohledu výkonu a miniaturizace vestavěných počítačových platforem. V řadě průmyslových odvětví lze proto nyní považovat implementaci technik strojového učení a optimalizace za možnou a žádoucí. Techniky navádění, navigace a řízení (GNC) jsou kritické pro autonomii vzdušných i pozemních prostředků. Vývoj algoritmů GNC se primárně spoléhá na návrh založeném na modelu, který vychází z principů fyziky a vede k vysvětlitelnému a verifikovatelnému řešení. Zvýšené požadavky spolu s dynamicky se měnícím prostředím představují výzvu pro současné algoritmy GNC a motivují využití mechanismů založených na učení a adaptaci. Související výzvou pro využití takovýchto systému v kritických aplikacích je nutnost integrace vestavěných obranných mechanismů zabezpečujících jejich stabilitu a provedení verifikace a validace. Cílem práce je navrhnout systém GNC pro bezpilotní letoun modifikovaný pro integrovanou robotickou manipulaci s využitím optimálního řízení a strojového učení pro dosažení robustního, vysvětlitelného a bezpečného řízení. Navržený statistický přístup založen na náhradním modelu je použit v rámci verifikace systému. Následná evaluace pravděpodobnosti selhání systému je důležitá v rámci jeho integrace do reálného světa, která vyžaduje certikaci příslušnými úřady.

KLÍČOVÁ SLOVA

adaptivní řízení; bezpilotní drony; konvexní optimalizace; náhradní modelování; nelineární programování; optimalizace trajektorie; optimální řízení; polynomiální chaos; prediktivní řízení; přímá kolokační metoda; vzdušná manipulace; vizuální řízení

ROZŠÍŘENÝ ABSTRAKT

Popularita moderních technik strojového učení a optimalizace je v současnosti podpořena pokrokem z pohledu výkonu a miniaturizace vestavěných počítačových platforem. V řadě průmyslových odvětví lze proto nyní považovat implementaci technik strojového učení a optimalizace za možnou a žádoucí. Techniky navádění, navigace a řízení (GNC) jsou kritické pro autonomii vzdušných i pozemních prostředků. Vývoj algoritmů GNC se primárně spoléhá na návrh založeném na modelu, který vychází z principů fyziky a vede k vysvětlitelnému a verifikovatelnému řešení.

Zvýšené požadavky na autonomní mise spolu s dynamicky se měnícím prostředím představují výzvu pro současné algoritmy GNC a motivují využití mechanismů založených na učení a adaptaci. Související výzvou pro využití takovýchto systému v kritických aplikacích je nutnost integrace vestavěných obranných mechanismů zabezpečujících jejich stabilitu a provedení verifikace a validace. Cílem práce je navrhnout systém GNC pro bezpilotní dron modifikovaný pro integrovanou robotickou manipulaci s využitím optimálního řízení a strojového učení pro dosažení robustního, vysvětlitelného a bezpečného řízení. Navržený statistický přístup založen na náhradním modelu je použit v rámci verifikace systému. Následná evaluace pravděpodobnosti selhání systému je důležitá pro integraci do reálného autonmního prostředku, která vyžaduje certikaci příslušnými úřady. V současnosti je očekáván další vývoj a využití statistických přístupů v rámci certifikace autonomních platforem s ohledem na vývoj systémů strojového učení, které nejsou klasicky programované, tudíž vyžadují jíný přístup v rámci prokázání jejich funkčnosti a vlastností.

Práce je členěna do šesti hlavních kapitol, které adresují aspekty vývoje, implementace a validace systémů GNC s důrazem na využití vhodných technik strojového učení, které vyžadují malé množství provedených experimentů. První kapitola shrnuje teoretický základ matematické optimalizace a jeho využití pro řešení problémů optimálního řízení. Tato kapitola také uvádí několik numerických příkladů, které jsou součástí publikací autora a které představují možná využití těchto metod v doméně bezpilotního letectví. Důraz je kladen zejména na výpočetně efektivní metody pro řešení nelineárních optimalizačních úloh a vyhodnocení výpočetních vlastností na dostupných vestavěných platformách.

Druhá kapitola uvádí koncept modelování vícerotorových bezpilotních dronů za použití základních fyzikálních zákonů a následné rozšíření o kinematický a dynamický model integrovaného robotického ramene. Tato kapitola také představuje vyvinutý softwarový nástroj pro návrh konfigurace bezpilotního dronu a zpřesněné kvazi-fyzikální modely bezpilotního dronu, robotického ramene a propojeného modelu.

Třetí kapitola se zabývá návrhem systému GNC pro bezpilotní dron s integrovaným robotickým ramenem. Systém založen na metodě prediktivního řízení je rozšířen o adaptivní zpětnovazební smyčku, která zajistí kontrolu i za předpokladu působení nejistot vlivem prostředí či rozdílem mezi skutečným a virtuálním dynamickým modelem. Důraz je také kladen na zavedení provozních omezení pro tento integrovaný systém, čímž je zajištěna bezpečnost provozu platformy.

Ve čtvrté kapitole jsou představeny základy křížové entropie vhodné jak pro kombinatorickou tak spojitou optimalizaci, jejíž využití je vhodné zejména pro black-box modely, kde není uvažován výpočet parciálních derivací v bodech prohledávaných řešení. Tato kapitola následně rozšíří základní metodu křížové entropie o simultánní validaci systému za použití náhradního modelu, který aproximuje pravděpodobnost selhání systému v návrhovém bodě.

Následující pátá kapitola dále uvadí provedené numerické experimenty a diskutuje hypotézy stanovené v úvodní kapitole. Nejprve je testován samotný bezpilotní dron nejdříve za použití prediktivního řízení a dále s rošířenou adaptivní smyčkou. Dále je testováno prediktivní řízení pro robotické rameno a nakonec je provedeno testování bezpilotního dronu s integrovaným robotickým ramenem. Systém řízení je navržen jako plně propojený, tedy bezpilotní letoun a robotické rameno nejsou řízeny samostatně, jak bývá běžné u podobných propojených pohybových platforem, ale sdruženě. Tento přístup má za následek rozšířené schopnosti vzhledem k úkolům vzdušné robotické manipulace, které mohou přesahovat běžné provozní režimy a využít predikční horizont uvažující kombinovaný dynamický systém. Kapitola dále obsahuje popis a návrh systému implementovaného v rámci letového experimentu, který integruje bezpilotní dron a pohyblivé rameno s jedním stupněm volnosti. Tento bezpilotní dron je vybaven nádrží a malým čerpadlem, tudíž lze za letu přečerpávat kapalinu z nebo do palubní nádrže skrze pohyblivé rameno.

V poslední kapitole jsou shrnuty dosažené výsledky a výzvy, které motivují další výzkum v oblastech prediktivního řízení pro vzdušnou robotickou manipulaci a náhradních modelů určených pro validaci a verifikaci systémů GNC. V této kapitole jsou také shrnuty publikační výsledky a projektová praxe autora související s tématem dizertační práce. Nakonec jsou uvedeny přílohy, které uvádí důležité koncepty, na kterých práce staví a seznam hardwarových komponnet využitých pro sestavení bezpilotního dronu použitého pro letový experiment.

BIBLIOGRAPHIC CITATION

Ing. Jiří Novák: *Nonlinear Optimal Control System Synthesis for Dynamically Coupled Autonomous Aerial Vehicles*, doctoral thesis Brno, Brno University of Technology, Faculty of Information Technology, 2024.

DECLARATION

I declare that this dissertation thesis is my original work and that I have written it under the guidance of Associate Professor Peter Chudý, Ph.D., MBA. All sources and literature that I used during my work on the thesis are correctly cited with a complete reference to the respective sources.

Brno, 2024

Ing. Jiří Novák, July 31, 2024

© Ing. Jiří Novák, 2024.

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, Associate Professor Peter Chudý, Ph.D., MBA., for his support, advice and feedback he had devoted to me and my research, and for teaching me to never give up.

Thank you to my colleagues for their friendship, hard work, and moral support. Special thanks to Ing. Jiří Hanák for his technical support.

I owe a debt of gratitude to my family and closest friends for their support over the past years.

Special thanks belongs to my wife for her love, understanding, and patience. I would not have gained so much resilience and determination without her.

CONTENTS

1 INTRODUCTION 1
- 1.1 Motivation and Problem Statement . . . 1
- 1.2 State of the Art . . . 5
- 1.3 Contributions of the Thesis . . . 8
- 1.4 Structure of the Thesis . . . 10

2 THEORETICAL BACKGROUND 11
- 2.1 Mathematical Optimization . . . 11
- 2.2 Nonlinear Optimization . . . 11
- 2.3 Convex Optimization . . . 12
- 2.4 Interior-Point Methods . . . 13
- 2.5 Proximal-Averaged Methods . . . 15
- 2.6 Optimal Control Theory Foundations . . . 16
- 2.7 Applications of Optimal Control Theory . . . 23
- 2.8 Summary and Solver Overview . . . 35

3 FIRST PRINCIPLE BASED MODELING 37
- 3.1 Multirotor Unmanned Aerial Vehicle Mathematical Modeling . . . 38
- 3.2 Multirotor Unmanned Aerial Vehicle Quasi-Physical Modeling . . . 45
- 3.3 Serial Robotic Manipulator Mathematical Modeling . . . 46
- 3.4 Serial Robotic Manipulator Quasi-Physical Modeling . . . 48
- 3.5 Disturbance Modeling . . . 49

4 SYSTEM ARCHITECTURE FOR THE COUPLED PROBLEM 55
- 4.1 Nonlinear Controller Design . . . 56
- 4.2 Adaptive Augmentation Loop Design . . . 60
- 4.3 Weight Adaptation . . . 62
- 4.4 Maneuver Planning . . . 64

5 RELIABILITY-ASSISTED SYSTEM OPTIMIZATION 65
- 5.1 Cross-Entropy Method . . . 65
- 5.2 Polynomial Chaos Expansion . . . 68
- 5.3 Reliability-Based System Optimization Algorithm . . . 70

6 NUMERICAL RESULTS AND EXPERIMENTS 73
- 6.1 Operational Design Domain Specification . . . 73
- 6.2 Testing Trajectory Specification . . . 74
- 6.3 Evaluation Metrics . . . 76
- 6.4 Multirotor Unmanned Aerial Vehicle . . . 78
- 6.5 Serial Robotic Manipulator . . . 89
- 6.6 Coupled System for Aerial Grasping . . . 93
- 6.7 Integration to Physical System . . . 113

7 CONCLUSION AND PERSPECTIVE 117

A REFERENCE FRAMES AND TRANSFORMATIONS 123
B NUMERICAL AND ANALYTICAL DIFFERENTIATION 127
C ORTHOGONAL POLYNOMIALS 129
D LIST OF HARDWARE COMPONENTS 131

LIST OF FIGURES

Figure 1.1 Extending a classical quadcopter airframe by robotic arm. 1
Figure 1.2 Range of trajectory optimization methods. 5
Figure 1.3 Range of state estimation methods. 6
Figure 1.4 Range of control system methods. 7
Figure 1.5 Diagram of model types and their relationship. 8
Figure 1.6 Modified V/W-diagram of the system engineering process. 9

Figure 2.1 Convex and non-convex set examples. 12
Figure 2.2 Convex and non-convex function examples. 12
Figure 2.3 Defect visualization over a short time horizon. 18
Figure 2.4 Obtained trajectories in a 3D plot. 22
Figure 2.5 Trajectory projection. 22
Figure 2.6 The velocity and SoC profiles. 22
Figure 2.7 Visualization of MPC prediction horizon. 29
Figure 2.8 Hover stabilization by MPC and SINDy augmented MPC. 30
Figure 2.9 UAV trajectory avoiding both static and moving obstacles. 34
Figure 2.10 Convergence time comparison of three selected solvers. 36
Figure 2.11 Convergence time comparison on three selected platforms. 36

Figure 3.1 Range of multirotor UAV configurations. 37
Figure 3.2 Multirotor UAV modeling tool and a terrain model. 42
Figure 3.3 Section cut of propeller's blade. 42
Figure 3.4 Induced velocity grid map at fixed vertical airflow magnitudes. 43
Figure 3.5 Rotor disk from side and top view. 43
Figure 3.6 Quadcopter model visualization shown in mechanical explorer. 45
Figure 3.7 Quasi-physical model components of a quadcopter drone. 46
Figure 3.8 Robotic arm model visualization shown in mechanical explorer. 48
Figure 3.9 Implementation of a single manipulator joint with friction. 48
Figure 3.10 Generated von Kármán translation and rotation velocities. 50
Figure 3.11 Example CFD simulation of ground effect. Source: [53] 51
Figure 3.12 Vortex magnitudes based on intensity and fly-by distance. 52
Figure 3.13 Vortex direction based on intensity and fly-by distance. 52
Figure 3.14 Varying wind gust profiles with symmetric shape. 53

Figure 4.1 Autonomous grasp maneuver with three distinct phases. 55
Figure 4.2 Control system architecture of UAV coupled with a manipulator. 56
Figure 4.3 Visualization of a coupled system with respective reference frames. 58
Figure 4.4 Schematics of soft perception constraints. 59

Figure 4.5 Visualization of a second-order cone safety constraint. 60
Figure 4.6 Schematics of weight adaptation loop trained by DRL approach. . . . 63
Figure 4.7 Varying approach trajectories to the target payload. 64

Figure 5.1 Schematics of reliability-assisted system optimization. 65
Figure 5.2 Example of the CEM update through the elite set of samples. 67
Figure 5.3 Schematics of PCE computational model approximation. 68
Figure 5.4 General approach for PCE surrogate model computation. 70
Figure 5.5 PCE-Kriging adaptive resampling for failure boundary refinement. . . 71

Figure 6.1 Count of measured wind speed fitted by Weibull distribution. 73
Figure 6.2 Count of target/obstacle detection based on slant range. 74
Figure 6.3 Considered testing trajectories projected to two planes. 74
Figure 6.4 Lemniscate trajectory following by a quadcopter UAV. 75
Figure 6.5 Aerial grasping observed in nature. Source: [88] 77
Figure 6.6 Trajectory following using quasi-physical model of a quadcopter. . . . 78
Figure 6.7 Geometric and static parameters of the selected propeller. 79
Figure 6.8 Aerodynamic characteristics of the selected propeller. 80
Figure 6.9 Convergence time statistics with respect to varying cost function gains. 80
Figure 6.10 Response of the quadcopter dynamics to reference step change. 81
Figure 6.11 Angular velocity control inputs with lower and upper limits. 81
Figure 6.12 Quasi-physical model control using different weighting matrices. . . . 82
Figure 6.13 Position and heading RMSE with respect to changing MPC update rate. 82
Figure 6.14 Adaptive augmentation loop inputs based on mass errors. 83
Figure 6.15 Adaptive augmentation loop inputs based on inertia errors. 83
Figure 6.16 Nominal MPC evaluation using PCE-Kriging surrogate model. 84
Figure 6.17 MPC with $\mathcal{L}_1$ evaluation using PCE-Kriging surrogate model. 84
Figure 6.18 Position deviation with respect to time-varying reference. 85
Figure 6.19 Spiral trajectory tracking optimization. 87
Figure 6.20 Optimizing MPC parameters for two types of models. 87
Figure 6.21 Optimized MPC evaluation using PCE kriging surrogate model. . . . 88
Figure 6.22 Box pick and place simulation of manipulator quasi-physical model. . 89
Figure 6.23 End-effector position control with reference target values. 90
Figure 6.24 End-effector orientation reference tracking. 91
Figure 6.25 Statistical evaluation of solver convergence on robotic arm MPC. . . . 91
Figure 6.26 Joint angle acceleration control inputs generated by MPC. 92
Figure 6.27 Joint angle position and angular velocities in demonstration case. . . . 92
Figure 6.28 Effect of coupled problem objective weighting on the simulation result. 93
Figure 6.29 Quadcopter state evolution using initial settings. 94
Figure 6.30 Joint angle position and angular velocities in demonstration case. . . . 94
Figure 6.31 Rotor angular velocities in initial testing of coupled MPC. 95
Figure 6.32 Joint angle acceleration inputs in initial testing of coupled MPC. . . . 95
Figure 6.33 Solver convergence for different problem structures. 96

Figure 6.34 Evolving states of a UAV performing a grasping maneuver. 97
Figure 6.35 Evolving control inputs of a UAV performing a grasping maneuver. . 97
Figure 6.36 Evolving joint states of an arm performing a grasping maneuver. . . . 98
Figure 6.37 Evolving control inputs of an arm performing a grasping maneuver. . 98
Figure 6.38 Effect of imposing safety operational constraints. 99
Figure 6.39 Visualization of the aerial grasping maneuver. 99
Figure 6.40 End-effector position with respect to commanded reference. 99
Figure 6.41 End-effector orientation in the inertial frame. 100
Figure 6.42 End-effector acceleration in the inertial frame. 100
Figure 6.43 End-effector angular velocity in the inertial frame. 100
Figure 6.44 Quadcopter states in the considered scenarios. 101
Figure 6.45 Quadcopter motor angular velocities in the considered scenarios. . . . 101
Figure 6.46 Convergence of CEM in fixed-weight problem configuration. 102
Figure 6.47 Convergence of CEM in weight-varying problem configuration. 102
Figure 6.48 Convergence of CEM in fixed-weight problem configuration. 103
Figure 6.49 Convergence of CEM with weight-varying problem configuration. . . 103
Figure 6.50 End-effector position considering the optimized controllers. 104
Figure 6.51 End-effector position considering the optimized controllers. 104
Figure 6.52 End-effector acceleration considering the optimized controllers. 104
Figure 6.53 End-effector angular velocities considering the optimized controllers. . 105
Figure 6.54 Evolving UAV states considering the optimized controllers. 105
Figure 6.55 UAV control inputs considering the optimized controllers. 105
Figure 6.56 Evolving joint states considering the optimized controllers. 106
Figure 6.57 Robotic arm control inputs considering the optimized controllers. . . . 106
Figure 6.58 Coupled UAV performing shallow descent to target. 107
Figure 6.59 Coupled UAV performing medium rate descent to target. 107
Figure 6.60 Coupled UAV performing sharp descent to target. 107
Figure 6.61 Quasi-physical model state trajectories in considered maneuvers. . . . 108
Figure 6.62 Manipulator joint angles and angular velocities. 108
Figure 6.63 UAV motor's angular rates in the considered maneuvers. 108
Figure 6.64 Autonomous aerial grasping simulations with varying MPC weights. . 109
Figure 6.65 Evolving UAV control inputs in the considered scenarios. 109
Figure 6.66 Evolving UAV states in considered scenarios. 110
Figure 6.67 Evolving robotic arm states in the considered scenarios. 110
Figure 6.68 Evolving robotic arm control inputs in the considered scenarios. . . . 110
Figure 6.69 Failure boundary before optimization. 111
Figure 6.70 Failure region RBCE iteration 1. 111
Figure 6.71 Failure region RBCE iteration 6. 111
Figure 6.72 Optimized aerial grasping maneuver robust to uncertainties. 112
Figure 6.73 Assembled drone platform with an integrated movable arm. 113
Figure 6.74 Block diagram of the interconnected software components. 114
Figure 6.75 Block diagram of the flight controller used in the modified firmware. . 114

Figure 6.76 AprilTag detection using the downfacing camera. 115

Figure A.1 Schematics of coordinate systems and their transformations. 123
Figure A.2 Body-Fixed frame visualization. 124
Figure A.3 Aerodynamic frame visualization. 124
Figure A.4 Kinematic frame visualization. 125
Figure A.5 North-East-Down frame visualization. 125

Figure B.1 Automatic and symbolic differentiation. 128

Figure C.1 Hermite polynomials for varying degrees on a unit domain. 129
Figure C.2 Legendre polynomials for varying degrees on a unit domain. 130

Figure D.1 Flight controller 3D representation. 131

LIST OF TABLES

Table 2.1 Overview of numerical optimization solvers used for Optimal Control. 35

Table 4.1 Overview of constraints for aerial grasping. 60

Table 6.1 Overview of tolerances and probabilistic thresholds. 77
Table 6.2 Overview of vehicle parameters used in the full nonlinear model. . . . 78
Table 6.3 Overview of BLDC motor parameters. 79
Table 6.4 Surrogate model settings used throughout the thesis. 85
Table 6.5 Evaluation of a mathematical model driven by nominal MPC. 85
Table 6.6 Evaluation of a mathematical model using MPC and L1. 86
Table 6.7 Evaluation of a quasi-physical model driven by nominal MPC. 86
Table 6.8 Evaluation of a quasi-physical model using MPC and L1. 86
Table 6.9 Fixed parameter settings of the CEM and MPC algorithms. 87
Table 6.10 Evaluation of failure probability for the spiral trajectory. 88
Table 6.11 DH parameters of the implemented serial robot and mass of each part. 89
Table 6.12 Rotational friction and rotational limit parameters of the joints. 90
Table 6.13 Evaluation of the end-effector kinematic errors. 101
Table 6.14 Kinematic errors evaluation on optimized controllers. 106
Table 6.15 Aerial grasping accuracy with initial design parameters. 112
Table 6.16 Aerial grasping accuracy with optimized design parameters. 112

Table D.1 Processors and sensors of the flight controller. 131
Table D.2 Electrical and mechanical specifications of the flight controller. 132
Table D.3 Raspberry Pi camera 3 technical specifications. 132
Table D.4 Raspberry Pi camera 3 technical specifications. 132
Table D.5 Waveshare SC15 programmable servo technical specifications. 133
Table D.6 SiK Telemetry Radio V3 technical specifications. 133
Table D.7 Zorro Radio Controller technical specifications. 133
Table D.8 RP1/RP2 ExpressLRS 2.4ghz Nano Receiver technical specifications. . 134
Table D.9 PM02D power module technical specifications. 134
Table D.10 BLHeli S ESC 20A technical specifications. 134
Table D.11 Holybro 2216 KV920 motor technical specifications. 135
Table D.12 Power Distribution Board technical specifications. 135
Table D.13 Gens Ace Battery LiPo technical specifications. 135
Table D.14 Peristaltic mini water pump technical specifications. 136
Table D.15 H-bridge module technical specifications. 136

ABBREVIATIONS AND ACRONYMS

AD	Automatic Differentiation
ADI	Adaptive Dynamic Inversion
AI	Artificial Intelligence
ANN	Artificial Neural Network
AoA	Angle of Attack
APF	Auxiliary Particle Filter
ARP	Aerospace Recommended Practice
BEMT	Blade Element Momentum Theory
BFF	Body-Fixed Frame
BLDC	Brush-Less Direct Current
BVP	Boundary Value Problem
C-EKF	Constrained Extended Kalman Filter
CEM	Cross Entropy Method
CFD	Computational Fluid Dynamics
CoG	Center of Gravity
CPS	Cyber-Physical Systems
DCM	Direction Cosine Matrix
DH	Denavit-Hartenber
DoF	Degrees of Freedom
DRL	Deep Reinforcement Learning
EASA	European Union Aviation Safety Agency
ECEF	Earth-Centered, Earth-Fixed
ECOS	Embedded Conic Solver
EKF	Extended Kalman Filter
EnKF	Ensemble Kalman Filter
EoM	Equations of Motion
ESC	Electronic Speed Controller
FAA	Federal Aviation Administration
FMEA	Failure Mode and Effects Analysis

FOH	First Order Hold
FoV	Field of View
GA	Genetic Algorithm
GNC	Guidance, Navigation, Control
GP	Gaussian Process
gPC	Generalized Polynomial Chaos
GPR	Gaussian Process Regression
GPS	Global Positioning System
GUI	Graphical User Interface
GuSTO	Guaranteed Sequential Trajectory Optimization
IEEE	Institute of Electrical and Electronics Engineers
IMM	Interacting Multiple Model
IMU	Inertial Measurement Unit
IPOPT	Interior Point OPTimizer
IVP	Initial Value Problem
KF	Kalman Filter
KKT	Karush-Kuhn-Tucker
KL	Kullback-Leibler
LHS	Latin Hypercube Sampling
LM	Levenberg-Marquardt
LG	Legendre Gauss
LGL	Legendre Gauss Lobatto
LGR	Legendre Gauss Radau
LP	Linear Programming
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LSR	Least-Square Regression
LTV	Linear Time Varying
MC	Monte Carlo
MHE	Moving Horizon Estimator
MIMO	Multiple Input Multiple Output
ML	Machine Learning

MLE	Maximum Likelihood Estimation
MOSEK	Interior point Optimizer for Linear Programming
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Control
NED	North-East-Down
NLP	Non-Linear Programming
OCP	Optimal Control Problem
ODD	Operational Design Domain
ODE	Ordinary Differential Equation
PANOC	Proximal Averaged Newton-type method for Optimal Control
PCE	Polynomial Chaos Expansion
PDF	Probability Density Function
PF	Particle Filter
PFI	Potential Flow of Images
PID	Proportional, Integral, Derivative
PMCMC	Markov Chain Monte Carlo Particle Filter
PSD	Power Spectral Density
PSO	Particle Swarm Optimization
PWM	Pulse-Width Modulation
QCQP	Quadratically Constrained Quadratic Programming
QP	Quadratic Programming
RBCE	Reliability-Based Cross Entropy
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RNDDR	Recursive Nonlinear Dynamic Data Reconciliation
RPM	Revolutions Per Minute
RRT	Rapidly exploring Random Tree
SA	Simulated Annealing
SCP	Sequential Convex Programming
SCvx	Successive Convexification
SINDy	Sparse Identification of Nonlinear Dynamics
SLP	Sequential Linear Programming

SMC	Sliding Mode Control
SoC	State of Charge
SOCP	Second-Order Cone Programming
SO(3)	Special Orthogonal Group
SQP	Sequential Quadratic Programming
SVD	Singular Value Decomposition
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
WGS84	World Geodetic System 1984

NOMENCLATURE

Symbol	Unit	Description
$\mathbf{A}$	1	State transition matrix in continuous time
A_D	m^2	Vehicle drag area
$\mathbf{A}_s$	1	Adaptation gain Hurwitz matrix
a	rad^{-1}	Lift curve slope
$\mathbf{B}$	1	Control allocation matrix in continuous time
b_m	$N \cdot m \cdot s \cdot rad^{-1}$	Viscous friction constant
$\mathbf{C}$	1	Measurement allocation matrix in continuous time
C_D	1	Drag coefficient
$\mathbf{C}_o$	$N \cdot m$	Coriolis matrix
C_P	1	Power coefficient
C_T	1	Thrust coefficient
c	m	Chord length of the propeller blade
$\mathbf{D}$	1	Feedforward matrix in continuous time
$\mathfrak{D}$	1	Differentiation matrix
d	m	Joint offset length of a robotic arm
$\mathbf{E}$	1	Identity matrix
$\mathbf{F}_a$	N	Total force acting in the frame a
$\mathcal{F}$	1	Reference frame notation
$\mathbf{F}$	1	State transition matrix in discrete time
$\mathbf{G}$	1	Control allocation matrix in discrete time
$\mathbf{G}_w$	1	End-effector pose weighting matrix
$\mathbf{g}$	$m \cdot s^{-2}$	Gravitational acceleration vector
$\mathbf{H}$	1	Measurement allocation matrix
$\mathbf{I}$	$kg \cdot m^2$	Inertia tensor
J_m	$kg \cdot m^2$	Rotor moment of inertia
$\mathbf{J}_R$	1	Rotational Jacobian matrix
$\mathbf{J}_T$	1	Translational Jacobian matrix
$\mathbf{K}$	1	Kalman filter gain
K	1	Number of time steps of optimal control problem
$\mathbf{K}^{cam}$	1	Camera intrinsic matrix
k	1	Rotor disk effective fraction

L	m	Multirotor vehicle arm length
$\mathcal{L}^p$	1	Lagrange polynomial basis
l	m	Link length of a robotic arm
$\mathbf{M}_a$	$N \cdot m$	Total moment acting in the frame a
m	kg	Vehicle mass
N	1	Number of propellers
N_b	1	Number of propeller blades
N_r	1	Number of manipulator joints
O_j	1	Propeller j rotation orientation
$\mathbf{P}$	1	State covariance matrix
$\mathcal{P}$	1	Solution to the Riccati equation
$\mathbf{p}$	m	Vehicle position
$\mathbf{Q}$	1	State error weighting matrix
$\mathbf{Q}^{est}$	1	Process noise covariance matrix
$\mathbf{q}$	1	Time-fixed parameters of the optimal control problem
$\mathbf{q}_{a/b}$	1	Orientation of frame a with respect to frame b
$\mathbf{R}_{a/b}$	1	Rotation matrix of frame a to frame b
$\mathbf{R}$	1	Control effort weighting matrix
$\mathbf{R}^{est}$	1	Measurement noise covariance matrix
R_{pr}	m	Propeller radius
R_s	m	Slant range
$\mathbf{r}$	1	Residual dynamics in continuous time
$\mathbf{S}$	1	Parameter allocation matrix in continuous time
$\mathbf{s}$	1	Residual dynamics in discrete time
$\mathbf{s}^{cam}$	1	Image plane coordinates
$\mathbf{T}$	1	Parameter allocation matrix in discrete time
t_0	s	Initial time
t_f	s	Final time
$\mathbf{U}$	1	Polynomial approximation of the control input
$\mathbf{u}$	1	Control input vector
u, v	1	Image plane coordinates components
$\mathbf{v}_{a/b}$	$m \cdot s^{-1}$	Velocity of frame a with respect to frame b
$\mathbf{v}$	1	Measurement noise vector
$\mathbf{v}^a$	$m \cdot s^{-1}$	Relative airflow
v_i	$m \cdot s^{-1}$	Induced velocity
W	$m \cdot s^{-1}$	Nominal wind speed constant

$\mathbf{W}$	1	Hessian of the Lagrangian function
$\mathbf{w}$	1	Process noise vector
$\mathbf{w}_l$	$m \cdot s^{-1}$	Local wind velocity vector
$\mathbf{w}^g$	1	Gauss quadrature weights
$\mathbf{X}$	1	Polynomial approximation of the state
$\mathfrak{X}$	1	General optimization vector
$\mathbf{x}$	1	State vector
$\mathbf{Z}$	1	Link transformation matrix
$\mathbf{Z}_{end}$	1	Robotic arm end-effector pose
$\mathfrak{Z}$	1	Link transformation matrix
$\mathbf{z}$	1	Set of variables for discretized optimal control
α	rad	Angle of attack
α_c	rad	Angle around common normal
α^{CE}	1	Cross-Entropy Method smoothing parameter
α_h	rad	Horizontal field of view
α_v	rad	Vertical field of view
α^x	1	Solution vector update step size
α^λ	1	Lagrange multiplier update step size
α^σ	1	Bound constraint multiplier update step size
$\boldsymbol{\beta}$	1	Expansion coefficients of the surrogate model
β_{sh}	1	Trust region shrinking constant
$\boldsymbol{\Gamma}$	1	Process noise input matrix
$\boldsymbol{\gamma}_s$	1	Slack variable vector
Δ	1	General free parameter
$\Delta\Theta_s$	rad	Twist angle slope
δt	s	Sampling time
ζ^a	$rad \cdot s^{-2}$	Angular acceleration uncertainties
ζ^t	$m \cdot s^{-2}$	Translational acceleration uncertainties
η	1	Discretization error
η^t	1	Trust region value
Θ_s^0	rad	Propeller pitch at the blade root
θ	rad	Rotation of the manipulator joint
$\boldsymbol{\lambda}$	1	Lagrangian multipliers
λ	1	Rotor inflow ratio
μ	1	Rotor advance ratio
μ_{ip}	1	Barrier parameter

μ_{pa}	1	Penalty parameter of the proximal averaged method
$\boldsymbol{\nu}$	1	Virtual control term
$\boldsymbol{\xi}$	1	Standard random variable vector
ξ_p	1	Penalty parameter update factor
ρ	$kg \cdot m^{-3}$	Air density
$\boldsymbol{\sigma}$	1	Bound constraints multipliers
σ_m	1	Matched uncertainties of the adaptive control loop
σ_{um}	1	Unmatched uncertainties of the adaptive control loop
σ	1	Rotor solidity
$\boldsymbol{\tau}$	$N \cdot m$	Torque on the manipulator joints
τ	1	Transformed time domain variable
$\boldsymbol{\Upsilon}$	1	Control allocation matrix
$\boldsymbol{\Psi}$	1	Multivariate polynomial basis functions
ψ_R	rad	Rotor frame azimuth angle
$\boldsymbol{\Omega}^s$	$rad \cdot s^{-1}$	Skew-symmetric angular velocity matrix
$\boldsymbol{\omega}$	$rad \cdot s^{-1}$	Vehicle angular velocity
ω_{co}	Hz	Cut-off frequency of a low-pass filter
ω_j	$rad \cdot s^{-1}$	Angular velocity of propeller j

Function	Description
$\mathcal{B}(\cdot)$	Ball constraint function
$\mathcal{F}(\cdot)$	Nonlinear continuously differentiable cost function
$\mathcal{F}_c(\cdot)$	Convex continuously differentiable cost function
$\mathbf{f}(\cdot)$	General nonlinear system dynamics
$\mathcal{G}(\cdot)$	Nonlinear convex or non-convex inequality constraints
$g(\cdot)$	Running cost/objective function
$\mathcal{H}(\cdot)$	Nonlinear convex or non-convex equality constraints
$\mathbf{h}(\cdot)$	General nonlinear measurement model
$\mathcal{J}(\cdot)$	Cost/objective function
$\mathcal{L}(\cdot)$	Lagrangian function
$\mathcal{M}(\cdot)$	Surrogate model
$\mathcal{P}_a(\cdot)$	Legendre polynomial of degree a
$\mathcal{R}_\Delta(\cdot)$	Fixed point residual operator
$\mathbf{\Lambda}(\cdot)$	Invertible linear operator
$\lambda^{tr}(\cdot)$	Aerial manipulation phase transition function
$\Xi(\cdot)$	Auxiliary or subproblem cost/objective function
$\xi(\cdot)$	Discretization function
$\Pi_{\mathcal{C}}(\cdot)$	Projection operator
$\sigma^{sg}(\cdot)$	Sigmoid function
$\Phi(\cdot)$	Terminal cost/objective function
$\mathbf{\Psi}(\cdot)$	Boundary constraints
$\mathbf{\Omega}(\cdot)$	Path constraints

Index	Description
$\square_A$	Variable expressed in the aerodynamic reference frame
$\square_B$	Variable expressed in the Body-fixed frame
$\square_C$	Variable expressed in the camera/end-effector reference frame
$\square_E$	Variable expressed in the Earth-centered, Earth-fixed frame
$\square_H$	Variable expressed in blade axis frame
$\square_K$	Variable expressed in the kinematic reference frame
$\square_N$	Variable expressed in the North-East-Down reference frame
$\square_P$	Variable expressed in the payload reference frame
$\square_R$	Variable expressed in rotor axis reference frame
$\square^T$	Matrix transposition
$\square_{lb}$	Lower bound of the variable
$\square_{ub}$	Upper bound of the variable
$\hat{\square}$	Estimate of the variable
$\bar{\square}$	Mean value of the variable
$\tilde{\square}$	Reference value of the variable
$\square^*$	Optimal value of the variable

1 INTRODUCTION

1.1 MOTIVATION AND PROBLEM STATEMENT

Vehicle autonomy has been a rapidly evolving research area since the rise of the aviation industry. In 1914, the first autopilot was demonstrated by an American pilot, Lawrence Sperry, at the Airplane Safety Competition (Concours de la Securité en Aéroplane) held in France. It used a simple gyroscopic stabilizer allowing for the demonstration of aircraft stability augmentation proving the concept of vehicle autonomy. The most iconic feat emphasizing the emerging capabilities of Guidance, Navigation, Control (GNC) systems has been the Apollo program resulting in the development of the Apollo Guidance Computer in 1966, the first computer based on silicon-integrated circuits at the edge of the digital era. GNC systems can now be found in essentially all autonomous or semi-autonomous systems such as spacecraft, aerial vehicles, ships and cars. The increasing safety and operational requirements with regards to systems performing various tasks in diverse, dynamically changing environments, such as autonomous Unmanned Aerial Vehicle (UAV) which can be augmented to extend their capabilities, are motivating further progress in the on-board autonomy of these vehicles. As shown in Figure 1.1, a simple quadcopter UAV architecture can be extended by integrating a robotic arm manipulator, thus enabling UAVs' interaction with the physical world. The standard triplet of key GNC constituents enabling autonomous air and ground vehicles can be described as follows. Guidance refers to the determination

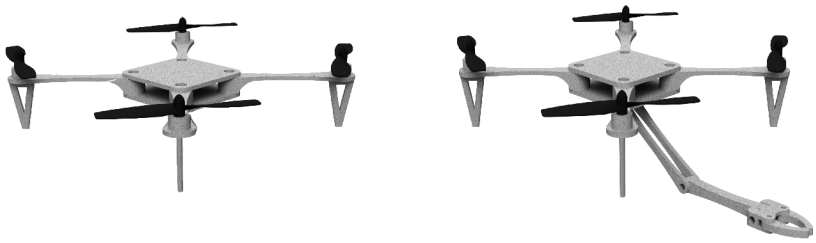

Figure 1.1: Extending a classical quadcopter airframe by robotic arm.

of the desired trajectory from the starting point to the target point along with the required changes in orientation, velocity, and other state variables while ensuring its feasibility with regards to the operational envelope of the vehicle. The process of trajectory determination is typically called trajectory optimization, since it aims to find the best possible trajectory in the context of a specified performance metric/objective. Navigation refers to the determination of the state vector (position, velocity, etc.) of the vehicle at a given time utilizing available measurement signals provided by the on-board sensors. The goal of a navigation algorithm is to provide the best possible estimate of the true state of the vehicle while suppressing noise due to sensor mechanical properties and changing environment. The navigation

algorithm has to handle uncertainties as well as incomplete state measurement. To solve such challenges, state estimation algorithms often utilize computational models of a virtual vehicle dynamics to predict the vehicle's full state based on past measurements and applied control inputs steering the vehicle. Each sensor may operate with a different sample rate and may have different operational bounds. Thus, the precision and accuracy of state estimation is typically highly correlated to the quality of sensors and redundancy improving the overall performance by utilizing a multi-sensor fusion. The hardware design choices for the vehicle navigation system are thus critical and specific for each application with cost to performance trade-offs and concept of operation having impact on the final decision. Control refers to steering the vehicle by its respective engines, propellers, control surfaces etc., such that the desired trajectory is achieved. The classic control theory developed in the 20th century is formally well described, verifiable, and conveniently implementable on embedded devices. It is used in many fields despite the known limitations of the methods. Today, most control applications use either the Proportional, Integral, Derivative (PID) controller, or the Linear Quadratic Regulator (LQR) based on the necessities of requiring a full-state feedback. Most control algorithms rely on utilizing a linear model of system dynamics [36]. Since most real-world systems are nonlinear, a surrogate model composed of locally valid linear models can be used to cover the whole operational envelope of the vehicle. With the recent advances in the Machine Learning (ML) methods and available computational performance, even safety-critical fields such as aviation have been recently enhancing the state-of-the-art navigation and control algorithms by data-driven methods such as Artificial Neural Network (ANN) and adaptive control techniques [12]. However, the lack of explainability and trustworthiness in complex ML models present an important challenge slowing the widespread adoption of ML techniques to safety-critical applications [71] involving GNC. Shifting to purely data-driven techniques may seem tempting given the success of vision-based or language-based ML solutions available today. To counter the argument, leaving behind the physics and control theory principles developed in the past puts heavy demands on the ML inference capabilities as well as training set quality and size. Also, the connections between adaptive control techniques and ML optimization have been recently observed [38] bridging the gap of model-based and data-driven approaches. An important attribute of adaptive control techniques is their online learning capabilities using only a short-term history of input data. Augmenting adaptive control techniques to state-of-the-art optimal control based methodologies such as Model Predictive Control (MPC) and Moving Horizon Estimator (MHE) has the potential to enhance the system capabilities and to guarantee stability. Parametrization of the resulting control system through objective function weights or adaptation gains can be done either manually, or automatically. The challenge associated with automated tuning of GNC algorithms include developing a suitable evaluation metric and test scenarios which would address the maximization of system capabilities given the hardware constraints. This theses will emphasize the intersections of ML and Optimal Control based GNC system and study ML-driven tuning techniques to maximize the capabilities of a selected UAV platform with an integrated robotic arm. Furthermore, verification and validation aspects for the given operational uncertainties will be addressed.

Conceptually, this is a different approach compared to defining formal stability properties based on Lyapunov theory [68]. Specifically, a novel approach of simultaneous parameter optimization and reliability analysis of the GNC will be introduced. The motivation behind this method stems from the need for a failure probability analysis for complex systems which are computationally expensive or hard to test in all intended operational conditions. Such systems (increasingly augmented by ML) are treated as a black-box, and their evaluation is purely statistics-based. Moreover, the identified failure boundary region can help to understand the limitations of the applied control system or the platform itself. The scope of the mentioned topics requires a dedicated review of the state-of-the-art in trajectory optimization, state estimation, and control theory to focus the thesis on relevant challenges in this field of study.

A correct understanding of the terminology used throughout the thesis is also important to avoid ambiguity in the conceptual and technical aspects of the thesis. It is important to clarify that not all the mentioned definitions are standardized and may differ based on the domain of use. In the following paragraphs, important concepts impacting the design choices in this thesis are explained:

- **Adaptive control** – Adaptive control refers to feedback control techniques in which the controller parameters or the internal model of the system are adjusted during operation to compensate for the changing conditions or system functions [62]. The adaptation can be either coupled or decoupled with the main controller based on the selected technique. In contrast to robust control, adaptive control does not specify bounds on the present uncertainties. They are, therefore, more flexible, but also harder to validate and verify.

- **Operational Design Domain** – According to the recommended practice for automated vehicles SAE J3016 [22], Operational Design Domain (ODD) represents a set of conditions for which a system is designed. From the end-user perspective, ODD is the operating environment in which the system should operate safely. ODD typically involves environmental, geographical, or time of day variables which can have impact on automated system functions. The term was initially designed and used for automated vehicles on roadways. However, the taxonomy has recently been used in aerospace industry in the context of black-box verification [28] of on-board machine learning perception algorithms.

- **Quasi-physical model** – A high-fidelity simulation model based on realistic virtual representations of a system [113] having both physical and mathematical roots. It is composed of physical, electrical, hydraulic, or other parts which are connected and form complex interactions. Quasi-physical models also include multi-body interactions with the respective contact forces. Such models sometimes fulfill the criteria of a multi-domain simulation model.

- **Safety-critical system** – A safety-critical or a safety-involved system comprises of software, hardware and all other aspects of a system performing a function in which fail-

ure would cause a significant increase in the safety risk for the people or environment involved [55]. Risks of this sort are usually managed with the methods and tools of safety engineering. The typical design methods include probabilistic risk assessment, a method which combines Failure Mode and Effects Analysis (FMEA) [9] with fault tree analysis. Safety-critical systems are increasingly computer-based.

- **Surrogate model** – Surrogate modeling is an engineering approach used in the cases when an outcome of interest cannot be easily measured or computed, so an approximate mathematical model of the outcome is used instead [49]. Most engineering design problems require experiments and/or simulations to evaluate the design objective and constraint functions as a function of the defined design variables. However, experiments may be prohibitively expensive, or simulations may take minutes, hours, or even days to complete. Evaluating the outcome statistics of the simulation or optimizing the parameters of the system may require thousands or even millions of simulation evaluations. Surrogate models, sometimes referred to as meta-models or emulators, mimic the behavior of the simulation model outcome while being orders of magnitude cheaper to evaluate.

- **Validation** – In terms of modeling and simulation, validation is the process of determining the degree to which a model or a simulation and their input and outputs are an accurate representation of the real world from the perspective of the intended use. Validation thus incorporates functional requirements for the model so that it is adequate for the pursued objective or customer needs [97]. In a broader sense, a standard adopted by the Institute of Electrical and Electronics Engineers (IEEE) defines validation as the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers.

- **Verification** – Verification is the process of determining that a computer model or a simulation accurately represent the developer's conceptual description and specifications [97]. In a broader sense, a standard adopted by IEEE defines verification as the evaluation of whether a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Mechanisms are implemented both for the verification of the hardware, through quality controls, and for the verification of software, through the execution of functional, system, or integration tests.

- **Visual servoing** – Visual servoing is a control technique based on the use of visual feedback provided by a monocular or stereo camera aboard the vehicle [20]. In terms of aerial manipulation, the camera can be mounted either on the airframe or on the end-effector of the manipulator with the aim of detecting the target object and stabilizing the vehicle and the manipulator with respect to the relative pose and velocity in relation to the target object as well as the obstacles present in the operational scene. With the increasing capabilities of computer vision algorithms, visual servoing has the potential to become essential for air and ground vehicle autonomy.

1.2 STATE OF THE ART

The development of GNC components requires considerable multidisciplinary knowledge in order to address all its desired functions and requirements. For this reason, it is important to diligently specify the area of research in the broad scope of GNC technologies. In terms of ML approach utilization, local adaptation law techniques as well as parameter adaptation will be considered since most state-of-the-art GNC techniques such as those having a solid theoretical background in optimal control and optimization are model-based. Thus, such adaptation approaches can result in improvements of the existing optimization-based algorithms such as MPC. The following summary of currently known techniques can serve as a guideline for anyone interested in the addressed topics.

Trajectory optimization often referred to as guidance can be divided into two main categories. Path planning methodology has been a successful approach in autonomous robot applications due to its simplicity and static environment. The goal of path planning is to find the shortest path from start to goal state while avoiding obstacles on the way [63]. Such a task can be handled very well by graph-based algorithms such as Djikstra, A*, or Rapidly exploring Random Tree (RRT). Specialized numerically simple guidance laws [81] are still frequently used in practice. More path planning techniques are listed in Figure 1.2. Although path planning formulation is suitable for simple robot applications, aerospace

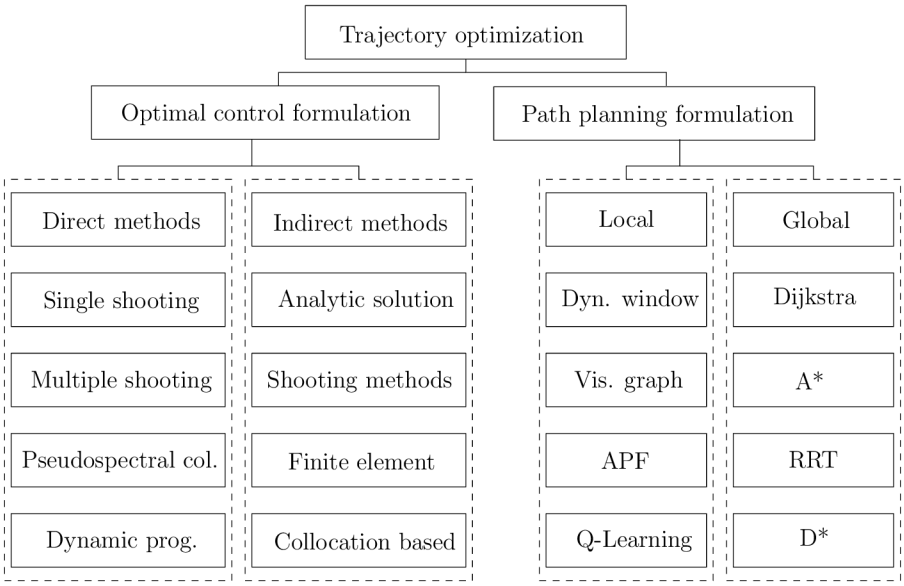

Figure 1.2: Range of trajectory optimization methods.

systems often require high-fidelity trajectories satisfying complex dynamic constraints and focusing on optimizing objectives [85] such as minimum time or minimum fuel burned by the vehicle. Such requirements can be satisfied by utilizing the Optimal Control Problem (OCP) formulation of the guidance problem [8]. The theoretical background for this approach stems from calculus of variations and Pontryagin's maximum principle. Since an analytic solution cannot be usually found (for most nonlinear problems), many numerical techniques were developed to solve these mathematical optimization problems. Indirect

methods require an analytical construction of necessary and sufficient conditions of the OCP, which are then discretized and solved by a numerical optimization algorithm. In contrast, direct methods discretize the OCP to a parametric optimization problem, which is then solved to find an optimal solution. Direct methods use the so-called transcription, which can be performed either by collocation or shooting methods. Collocation methods simultaneously optimize control inputs and state trajectory by posing a set of dynamic equality constraints, while shooting methods optimize only control inputs where the state trajectory has to be simulated by integration methods such as Runge-Kutta. Specialized collocation methods that seek to maximize accuracy of the obtained solution include Pseudospectral collocation methods representing the entire trajectory by high-order orthogonal polynomials.

The state estimation methods can be classified and separated into multiple groups as seen in Figure 1.3. Generally, the state estimation methods are divided into constrained and unconstrained methods. Constrained methods such as MHE [89], Constrained Extended Kalman Filter (C-EKF) [114] or Recursive Nonlinear Dynamic Data Reconciliation (RNDDR) method [115] typically enable setting state inequality constraints on variables leading to increased safety of the output estimates. The widely used unconstrained state estimators can be divided into linear and nonlinear estimators. The optimal linear state estimator is the widely used Kalman Filter (KF) [105] and forms a theoretical basis for many other methods. They include mainly its nonlinear extensions such as Extended Kalman Filter (EKF) [44], Unscented Kalman Filter (UKF) [69] or Ensemble Kalman Filter (EnKF) [52]. Nonlinear filtering has seen a great research attention due to its applicability in industry, especially aerospace and automotive sectors. A popular class of nonlinear filters use the Monte Carlo (MC) sampling methodology. They include a Particle Filter (PF) [30], Auxiliary Particle Filter (APF) [73], Interacting Multiple Model (IMM) [104] filter or Markov Chain Monte Carlo Particle Filter (PMCMC) [6] among others. Learning-based filters based on KF such as [90] have also been proposed. Although computationally more expensive [87], MHE allows for the utilization of various ML-based or hybrid models [72], [51] to improve the state estimation performance.

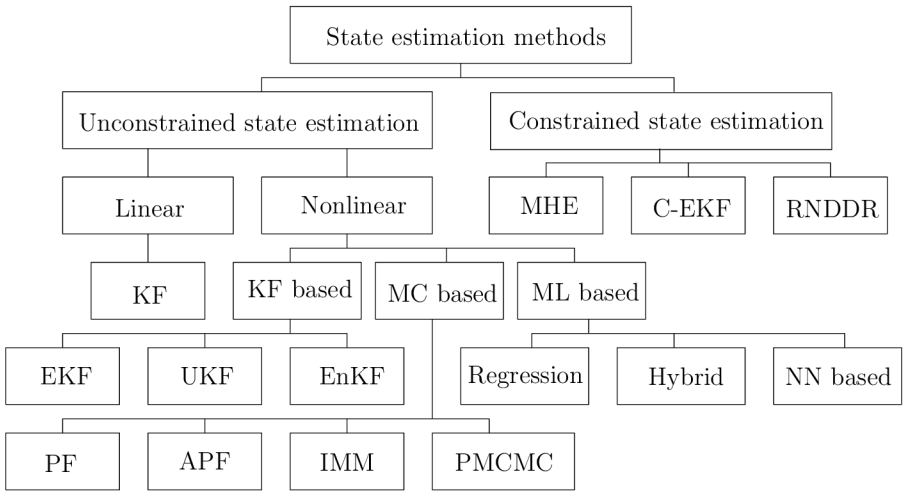

Figure 1.3: Range of state estimation methods.

Finally, the control system design employs a variety of methods with both low and high complexity of the control law. The control techniques focus on different aspects and desired objectives driving the control system architecture. The categories and control system design approaches are summarized in Figure 1.4. Classic control methods rely on the linearity of the model and employ Laplace transform to develop closed-loop systems in the frequency domain. These methods include the popular PID regulators [36], lead-lag compensators, and a pole placement method among others. The development of an optimal control theory led to an analytical solution for a subclass of the Boundary Value Problem (BVP) including linear dynamics and quadratic objective function, now commonly known as LQR, and later Linear Quadratic Gaussian (LQG). Nowadays, the computational methods allow for incorporating constraints and predict future control inputs by repeated optimization on a finite time window. The approach called MPC [34] has seen a lot of interest in the last decade allowing for the integration within ML structures [86], [48]. A very common property, which is desired by control theorists, is resilience to uncertainty. A field of robust control deals with these uncertainties and aims at achieving the desired performance and stability in the presence of bounded modeling errors. Methods such as H_2 or H_∞ control rely on minimizing the sensitivity of the system with respect to disturbances. One of the emerging control techniques is the Sliding Mode Control (SMC), which has shown to be robust with respect to the matched uncertainties. Contrary to static robust controllers, the adaptive control techniques are founded on the notion of parameter estimation and a suitable control update law which adapts to changing dynamics. This complex task may be realized through methods such as Adaptive Dynamic Inversion (ADI) [106], Model Reference Adaptive Control (MRAC) [12], $\mathcal{L}_1$ adaptive control [45], or through some form of adaptive neural network. Neural networks are often utilized in MRAC to model system nonlinearities. Lastly, the ML control methods rely on a large quantity of data or simulations (neural network-based control) allowing for the development of control policies through Reinforcement Learning (RL) [42], or Genetic Algorithm (GA) paradigm.

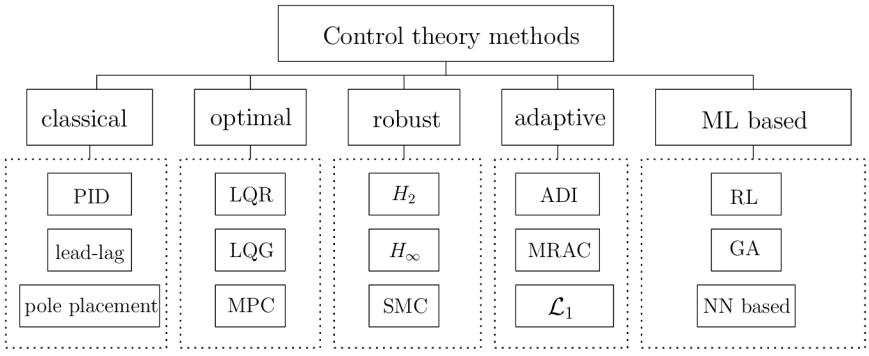

Figure 1.4: Range of control system methods.

This thesis focuses mostly on optimal control-based methods and nonlinear optimization in general. The rigorous formulation with a clearly defined mathematical structure and potential fusion of optimal control with data-based or hybrid models in the presence of state-of-the-art system identification methods [65] makes the approach explainable and verifiable, especially in the domain of Cyber-Physical Systems (CPS). One of the key elements presented in the thesis is the integration of RL within MPC structure adapting its weights on the fly.

1.3 CONTRIBUTIONS OF THE THESIS

The focus of this thesis can be divided into two major parts. The first key objective is to develop an optimization-based GNC system for multirotor UAVs with extended capabilities with regards to operational uncertainty robustness while assuming low-cost embedded devices and sensors so that the proposed vehicle concept is accessible. The UAV capabilities will be extended by an integrated robotic arm enabling complex proximity operations, but requiring an increased complexity of the GNC system given the highly nonlinear coupled dynamics of the vehicle. The following set of hypotheses will be studied and addressed from the engineering perspective:

- Dedicated nonlinear optimization solvers can achieve convergence speeds enabling a real-time operation of MPC and MHE algorithms even on cheap embedded devices.
- UAV concept with an attached robotic arm can be controlled robustly combining MPC with $\mathcal{L}_1$ adaptation loop given the range of encountered uncertainties.
- Treating the vehicle as a coupled system instead of two separately controlled platforms enables extended operational capabilities such as aerial grasping in flight regimes beyond a hover condition.

The second part of the thesis addresses the optimization of the system design parameters and the subsequent verification and validation techniques of the system. Generally, engineering systems are modeled using first-principle knowledge of the underlying physics, thus forming a mathematical model of the system behavior. Mathematical models are typically restrictive since they describe the system from a single domain perspective. The discrepancies emerging when comparing the mathematical model and the real system can thus be substantial. In comparison, Quasi-physical models are composed of interconnected models integrating multiple domains, thus being more accurate representations of a real system, but being computationally more expensive. In recent years, there has also been an increasing interest in the domain of digital twins. They represent a synchronized model that is repeatedly updated by sensor data from the real system. Such models thus become increasingly accurate and follow the life cycle of the real system. Figure 1.5 shows a Venn diagram progressing from low-level mathematical models to the real system adding layers of complexity in the modeling process.

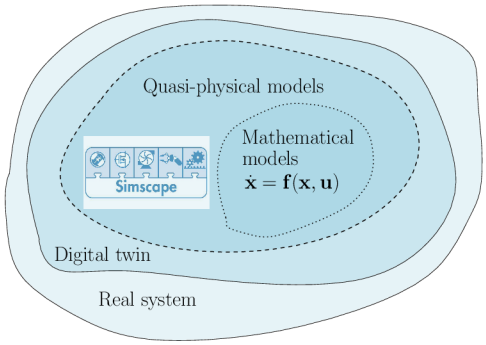

Figure 1.5: Diagram of model types and their relationship.

Following the Aerospace Recommended Practice (ARP) standards, the requirements for the system are developed by initially specifying the intended ODD of the system. The regulators such as European Union Aviation Safety Agency (EASA) recognize requirements such as system failure probability threshold given the ODD as a key aspect in a verification process. The terminology with regards to verification of black-box systems followed the Federal Aviation Administration (FAA) report on adaptive control system verification [32]. An in-depth technical report by EASA on Artificial Intelligence (AI) roadmap for the next two decades [29] introduced the challenges and process of verification to address social acceptance issues and trustworthiness of black-box systems. This thesis, therefore, considers a novel modified V/W-diagram of the system engineering process given by 1.6 incorporating iterative system optimization and failure boundary analysis enabling optimization of performance of the system while evaluating the probability of failure given the ODD. The failure probability determina-

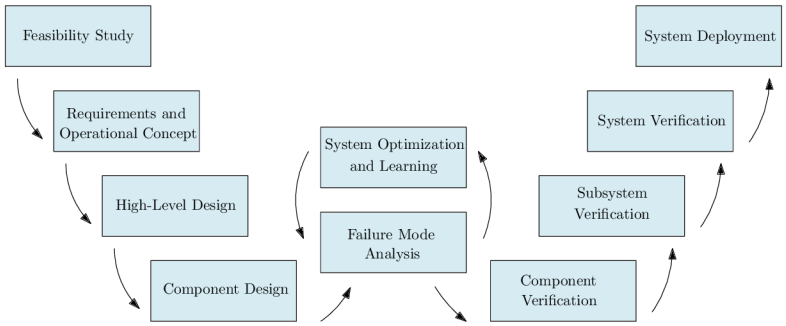

Figure 1.6: Modified V/W-diagram of the system engineering process.

tion can require a large number of sampling points assuming either high-fidelity simulations or experiments. Surrogate modeling utilizing the Polynomial Chaos Expansion (PCE) method is, therefore, used to effectively sample the ODD and model the failure boundary region. Cross Entropy Method (CEM) is used to optimize the parameter space of the system. Mathematical and quasi-physical representations of the multirotor vehicle and the robotic arm are examined to validate a mathematical model used within the MPC controller. The studied aspects mentioned above can be summarized by the following set of hypotheses:

- Parameter optimization coupled with failure boundary evaluation can improve the overall performance of the system while retaining the required robustness with respect to ODD without a major change in hardware configuration.
- PCE surrogate modeling can be an accurate and efficient way of evaluating the failure boundary region requiring less simulations given the likelihood model of the ODD.
- Treating the system as black-box with respect to tuning and verification reduces the computational cost while achieving a satisfactory convergence rate using CEM.

The contributions of the second part of the thesis are a framework for simultaneous engineering system optimization and verification by an ML-driven methodology requiring a low number of evaluation points. By using adaptive evaluation point selection, the failure boundary region can be accurately identified.

1.4 STRUCTURE OF THE THESIS

This thesis is divided into 6 dedicated chapters covering several aspects of GNC system design, implementation and validation with special emphasis on optimal control, and statistical ML methods for system optimization, verification, and validation.

The first chapter covers the theoretical background of mathematical optimization in general and it is subsequently focused on optimal control theory. This chapter also presents a range of numerical examples showing the utilization of the presented methods in the aerospace domain. Special emphasis is given to efficient computational methods for nonlinear optimal control with the assessment of viability for cheap embedded platforms.

The second chapter introduces the first-principle-based modeling approach used to model multirotor UAV and robotic arm kinematics and dynamics. The chapter also introduces a software tool dedicated to UAV configuration modeling and MATLAB®/Simulink® quasi-physical implementations of the multirotor vehicle, serial robotic manipulator and the coupled system utilizing SimscapeMultibody™ extension of Simulink®.

The third chapter presents the proposed GNC architecture for the coupled system utilizing OCP formulation for the coupled problem, MPC control loop augmented by the $\mathcal{L}_1$-adaptive feedback loop, and state estimator coupling Inertial Measurement Unit (IMU) sensor measurements with a vision-based detection algorithm augmenting perception constraints or objective term to the MPC architecture.

In the fourth chapter, a theoretical introduction to the CEM for combinatorial and continuous optimization is presented along with its modification for derivative-free real-valued optimization of the system parameters. This chapter also presents a coupled approach for parameter optimization and system verification using the PCE surrogate modeling approach resulting in the methodology called reliability-based control system optimization.

The fifth chapter focuses on evaluating the numerical experiments and discussion with respect to the hypotheses presented earlier. The chapter introduces numerical testing of multirotor vehicles and the extension of their stability properties by adaptive control. In the second part, robotic arm precision guidance and control is examined, and finally, in the third part, testing of the coupled system is performed in a number of scenarios.

The last chapter summarizes the achieved results presented in the thesis and address the remaining challenges and possible future scientific work or implementation challenges. At the end of this thesis, appendices extending important theoretical aspects used in the work are further described including chapters on orthogonal polynomials, numerical and automatic differentiation, and reference frames used in the aerospace engineering domain.

2 THEORETICAL BACKGROUND

2.1 MATHEMATICAL OPTIMIZATION

The term optimization is used in a variety of contexts and fields. Generally, the process of optimization is always connected to finding a set of variable values, so that the objective of the investigated problem shows improvement compared to the initial set of variable values. If the objective cannot be improved with respect to the variables in play, the solution can be declared globally optimal. If the objective cannot be improved in the vicinity of a current solution, but it can be improved globally, it is called a local optimum. The methods and goals of optimization are highly problem-dependent, but the problem can be usually translated into mathematical equations. The optimization problem in a mathematical form can be either solved analytically, or, most often, it has to be solved using numerical methods. Numerical optimization is a critically important tool in both machine learning and the analysis of physical systems in the context of time-evolving state estimation and optimal trajectory (with respect to a mathematically crafted objective) generation. In this thesis, a numerical optimization to nonlinear and convex programming problems is categorized. Other classifications of numerical optimization problems are often used, e.g. constrained and unconstrained, where constraints are formed by equality or inequality constraints involving optimization variables, or continuous and discrete. Most engineering optimization problems deal with some form of constraints to satisfy system requirements or physical limitations.

2.2 NONLINEAR OPTIMIZATION

A general mathematical formulation of an optimization problem can be described by a Non-Linear Programming (NLP) problem including real-valued nonlinear objective function which depends on the optimization vector $\mathcal{X} \in \mathbb{R}^n$ and the set of equality and inequality constraints given by equation (2.1), where $\mathcal{F}(\mathcal{X}) : \mathbb{R}^n \to \mathbb{R}$ is the continuously differentiable cost function ideally satisfying Lipschitz continuous gradient condition, $\mathcal{G}(\mathcal{X}) : \mathbb{R}^n \to \mathbb{R}^m$ are nonlinear convex or non-convex inequality constraints at least twice continuously differentiable and $\mathcal{H}(\mathcal{X}) : \mathbb{R}^n \to \mathbb{R}^p$ are nonlinear convex or non-convex equality constraints also at least twice continuously differentiable.

$$\begin{aligned} \mathcal{J} &= \min_{\mathcal{X} \in \mathbb{R}^n} \mathcal{F}(\mathcal{X}) \\ \text{s.t. } & \mathcal{G}_i(\mathcal{X}) \leq 0, \; i = 1, \ldots, m \\ & \mathcal{H}_j(\mathcal{X}) = 0, \; j = 1, \ldots, p \end{aligned} \tag{2.1}$$

NLP problems can be hard to solve and there are no theoretical guarantees that a numerical solver will converge to global optimum. Despite this, given a reasonable initial guess based on domain knowledge of the studied problem, problem (2.1) can be solved efficiently.

2.3 CONVEX OPTIMIZATION

In order to introduce convex optimization, it is necessary to define two fundamental objects, a convex set and a convex function. A convex set $\mathcal{C} \subseteq \mathbb{R}^n$ contains any line segment connecting two points $\mathcal{X}_i \in \mathcal{C}$ and $\mathcal{X}_j \in \mathcal{C}$ in the set as written in equation (2.2).

$$\mathcal{X}_i, \mathcal{X}_j \in \mathcal{C} \Rightarrow \Delta\, \mathcal{X}_i + (1-\Delta)\, \mathcal{X}_j \in \mathcal{C} \quad \forall\, \Delta \in [0,1] \tag{2.2}$$

A significant advantage when working with convex sets lies in their intersection property allowing them to combine multiple convex sets while preserving the convexity of the whole set. Figure 2.1 visualizes examples of non-convex and convex sets in a) and b) respectively.

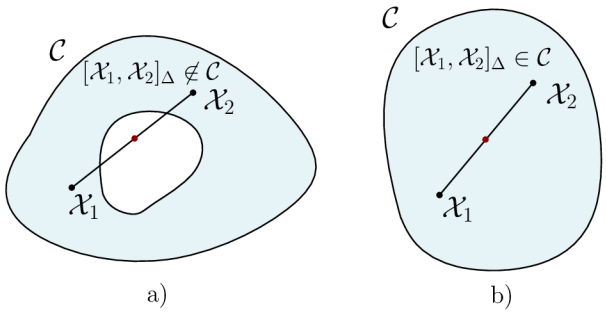

Figure 2.1: Convex and non-convex set examples.

A convex function $\mathcal{F}_c : \mathbb{R}^n \to \mathbb{R}$ requires its domain to be a convex set and $\mathcal{F}_c$ has to lie below the line segment connecting any of its points as written in equation (2.3).

$$\mathcal{X}_i, \mathcal{X}_j \in \operatorname{dom} \mathcal{F}_c \Rightarrow \mathcal{F}_c(\Delta\, \mathcal{X}_i + (1-\Delta)\, \mathcal{X}_j) \leq \Delta\, \mathcal{F}_c(\mathcal{X}_i) + (1-\Delta)\, \mathcal{F}_c(\mathcal{X}_j) \quad \forall\, \Delta \in [0,1] \tag{2.3}$$

Figure 2.2 visualizes examples of non-convex and convex functions in a) and b) respectively. In the context of defined mathematical objects, a convex optimization problem is defined as

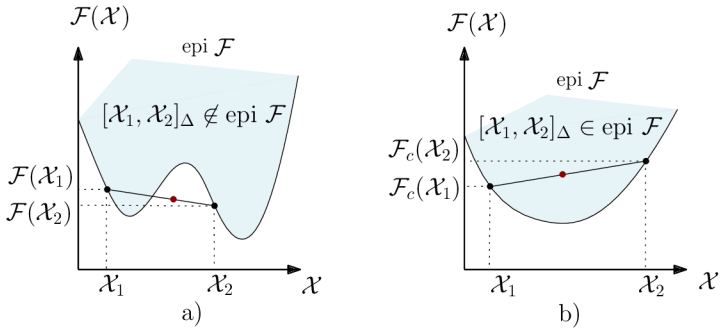

Figure 2.2: Convex and non-convex function examples.

the minimization of a convex function subject to convex inequality constraints (forming a convex set) and affine equality constraints. It can be formally written as in equation (2.4).

$$\begin{aligned} \mathcal{J} &= \min_{\mathcal{X} \in \mathbb{R}^n} \mathcal{F}_c(\mathcal{X}) \\ \text{s.t. } & \mathcal{G}_i(\mathcal{X}) \leq 0,\ i = 1, \ldots, m \\ & a_i^T \mathcal{X} = b_i,\ i = 1, \ldots, p \end{aligned} \tag{2.4}$$

Convex optimization problems include several subclass problems including Linear Programming (LP), Quadratic Programming (QP), Quadratically Constrained Quadratic Programming (QCQP), and Second-Order Cone Programming (SOCP). An LP problem assumes the objective and all constraints to be affine, and the feasible set is defined by a polyhedron. A famous simplex method has guaranteed convergence to an optimal solution if the problem is feasible. The QP problem, on the other, hand assumes a convex quadratic objective function and affine constraints with a feasible set also defined by a polyhedron. A QCQP assumes both the objective and inequality constraints to be quadratic functions. The feasible region is thus an intersection of ellipsoids. SOCP problems are a special case of convex optimization problems, which can be defined by equation (2.5).

$$\begin{aligned} \mathcal{J} = \min_{\mathcal{X}\in\mathbb{R}^n} \; & f^T\mathcal{X} \\ \text{s.t. } & \|A_i\mathcal{X} + b_i\|_2 \leq c_i^T + d, \; i = 1,\ldots,m \\ & F\mathcal{X} = g \end{aligned} \tag{2.5}$$

The problem includes parameters $f \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{n_i \times n}$, $b_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $F \in \mathbb{R}^{p\times n}$, and $g \in \mathbb{R}^p$. The feasible set of SOCP is the intersection of conic regions. Many engineering applications can be formulated by an SOCP and solved efficiently and reliably (compared to NLPs) by interior-point methods.

2.4 INTERIOR-POINT METHODS

The state-of-the-art algorithms for solving general NLP problems are called interior point (barrier) methods. The interior point methods define the auxiliary objective function formulation of the form given by equation (2.6).

$$\mathcal{J} = \min_{\mathcal{X}\in\mathbb{R}^n} \Xi(\mathcal{X}) = \mathcal{F}(\mathcal{X}) - \mu_{ip} \sum_{i\in I_{lb}} \ln(\mathcal{X}^i - \mathcal{X}^i_{lb}) - \mu_{ip} \sum_{i\in I_{ub}} \ln(\mathcal{X}^i_{ub} - \mathcal{X}^i) \tag{2.6}$$

The sets I_{lb} and I_{ub} are defined as $I_{lb} = \left\{i : \mathcal{X}^i_{lb} \neq -\infty\right\}$ and $I_{ub} = \left\{i : \mathcal{X}^i_{ub} \neq \infty\right\}$. The finite bound constraints are, therefore, replaced by the logarithmic barrier terms. In the case of barrier parameter $\mu_{ip} > 0$, the auxiliary objective function goes to infinity if any variable approaches its finite lower or upper bound. A popular implementation of the interior point method is the freely available Interior Point OPTimizer (IPOPT) [116], the primal-dual interior-point algorithm with a filter line-search method. A sequence of barrier problems is solved within its computations, while reducing the value of μ_{ip}. Once $\mu_{ip} \longrightarrow 0$ approaches zero, the final solution should converge to an optimal solution of the original problem. The inequality constraints are transformed to a set of equality constraints by introducing bounded slack variables γ^i_s. The constraint transformation is performed as in equation (2.7) forming a set of equality constraints $\mathcal{H}(\mathcal{X}) = 0$ of size $2m - p$, where p is the number of equality constraints.

$$\begin{aligned} \mathcal{G}_i(\mathcal{X}) - \gamma^i_s &= 0 \\ \gamma^i_{lb} \leq \gamma^i_s &\leq \gamma^i_{ub} \end{aligned} \tag{2.7}$$

The reduction of μ_{ip} is repeated, until the first-order optimality conditions are satisfied up to the user-defined tolerances. They incorporate the Karush-Kuhn-Tucker (KKT) conditions, which are given by equations (2.8).

$$\begin{aligned} \bigtriangledown \mathcal{F}(\mathcal{X}) + \bigtriangledown \mathcal{H}(\mathcal{X})\lambda - \sigma &= 0 \\ \mathcal{H}(\mathcal{X}) &= 0 \\ \mathbf{X}_d \mathbf{H}_d - \mu_{ip}\mathbf{e} &= 0 \\ \mathcal{X}, \sigma &\geq 0 \end{aligned} \tag{2.8}$$

The vector $\lambda \in \mathbb{R}^{2m-p}$ stores the constraint Lagrangian multipliers, and $\sigma \in \mathbb{R}^n$ stores the bound constraints multipliers. The matrix $\mathbf{X}_d = \text{diag}(\mathcal{X})$ is a diagonal matrix containing the variable vector, and $\mathbf{H}_d = \text{diag}(\sigma)$ is a diagonal matrix containing the bound multipliers. The vector $\mathbf{e}$ is defined as $\mathbf{e} = [1, 1 \ldots, 1]^T$. The optimality conditions are satisfied not only by the minimum value of the objective function, but also by the maximum value (if there is any) and saddle points. A Newton-type algorithm using a Hessian regularization is, therefore, applied so that the algorithm doesn't converge to a local maximum or a saddle point. The algorithm proceeds to find a solutions by iterating vector $\mathbf{i}_k = [\mathcal{X}_k \ \ \lambda_k \ \ \sigma_k]$ such that $\mathcal{X}_k > 0$ and $\sigma_k > 0$. The difference vector $\Delta\mathbf{i}_k = [\Delta\mathcal{X}_k \ \ \Delta\lambda_k \ \ \Delta\sigma_k]$ is computed by equation (2.9)

$$\begin{bmatrix} \mathbf{W}_k + \mathbf{X}_d^{-1}\mathbf{H}_d^k + \Delta \cdot \mathbf{E} & \nabla\mathcal{H}(\mathcal{X}_k) & 0 \\ \nabla\mathcal{H}(\mathcal{X}_k)^T & 0 & 0 \\ \mathbf{H}_d^k & 0 & \mathbf{X}_d^k \end{bmatrix} \begin{bmatrix} \Delta\mathcal{X}_k \\ \Delta\lambda_k \\ \Delta\sigma_k \end{bmatrix} = - \begin{bmatrix} \nabla\Xi(\mathcal{X}_k) + \nabla\mathcal{H}(\mathcal{X}_k)\lambda_k \\ \mathcal{H}(\mathcal{X}_k) \\ \mu_{ip}\mathbf{e} - \mathbf{X}_d^k\sigma_k \end{bmatrix} \tag{2.9}$$

The matrix $\mathbf{W}_k$ represents the Hessian of the Lagrangian function, which can be computed by equation (2.10) where ∇^2_{xx} represents the Hessian operator with respect to problem variables.

$$\mathbf{W}_k = \nabla^2_{xx}\mathcal{L}(\mathcal{X}_k, \lambda_k, \sigma_k) = \nabla^2\mathcal{F}(\mathcal{X}_k) + \sum_{j=1}^{2m-p} \lambda_k^j \nabla^2\mathcal{H}^j(\mathcal{X}_k) \tag{2.10}$$

The computation step of difference, therefore, requires a Hessian matrix $\mathbf{W}_k$, a Jacobian of the constraints $\nabla\mathcal{H}(\mathcal{X}_k)$, and a gradient of the auxiliary function $\nabla\Xi(\mathcal{X}_k)$. A line-search is then performed to construct the vector $\mathbf{i}_{k+1}$. The step sizes $\alpha^x_{k,l} = 2^{-l}\alpha^x_{ub}, l = 0, 1, \ldots$ are used to find the step size which leads to sufficient progress, while limited by the maximum step size α^x_{ub}. The current solution and respective parameters are then updated by equation (2.11).

$$\begin{aligned} \mathcal{X}_{k+1} &= \mathcal{X}_k + \alpha^x_{k,l}\Delta\mathcal{X}_k \\ \lambda_{k+1} &= \lambda_k + \alpha^\lambda_{k,l}\Delta\lambda_k \\ \sigma_{k+1} &= \sigma_k + \alpha^\sigma_k\Delta\sigma_k \end{aligned} \tag{2.11}$$

The maximum step sizes α^σ_{ub} and α^λ_{ub} are bounded in the interval $(0, 1]$ and the values are obtained by the fraction-to-the-boundary rule [116]. The value Δ is set to zero and is only increased if the number of negative eigenvalues of the matrix becomes large, which is verified by a factorization. After converging, the solution $\mathcal{X}^*$ is obtained.

2.5 PROXIMAL-AVERAGED METHODS

Let us assume NLP problem formulation (2.1) with assumptions that the problem can be written as in equation (2.12), where $\mathcal{U} \subseteq \mathbb{R}^n$ is a nonempty closed set such that projections can be computed over it, and $\mathcal{C} \subseteq \mathbb{R}^m$ is a convex set.

$$\begin{aligned} \mathcal{J} &= \min_{x \in \mathcal{U}} \mathcal{F}(x) \\ \text{s.t. } & \mathcal{G}(x) \in \mathcal{C} \\ & \mathcal{H}_j(x) = 0,\ j = 1, \dots, p \end{aligned} \tag{2.12}$$

To solve the presented problem, the following modified problem is introduced and solved iteratively with $k = 0, \dots K_{max}$. An augmented Lagrangian (2.13) is constructed assuming $\mu_{pa}^k > 0$ constant penalty parameter, $\lambda_k \in \mathbb{R}^m$ are the Lagrange multipliers for constraints, and $\sigma_k \in \mathcal{C}$ are the bound constraint parameters.

$$\mathcal{L}(x_k, \lambda_k, \sigma_k) = \mathcal{F}(x_k) + \langle \lambda_k, \mathcal{G}(x_k) - \sigma_k \rangle + \frac{\mu_{pa}^k}{2} \|\mathcal{G}(x_k) - \sigma_k\|^2 + \frac{\mu_{pa}^k}{2} \|\mathcal{H}(x_k)\|^2 \tag{2.13}$$

Following proposition (2.14) adopted from [107], the transformation of the Lagrangian function allows for the computation of the inner problem using Proximal Averaged Newton-type method for Optimal Control (PANOC) [108] optimizer.

$$\mathcal{J} = \min_{x_k \in \mathcal{U}, \sigma_k \in \mathcal{C}} \mathcal{L}(x_k, \lambda_k, \sigma_k) = -\frac{1}{2\mu_{pa}^k} \|\lambda_k\|^2 + \min_{x \in \mathcal{U}} \Xi(x_k, \lambda_k, \mu_{pa}^k) \tag{2.14}$$

Auxiliary objective function $\Xi(x_k, \lambda_k, \mu_{pa}^k)$ assuming simple box constraints can be expressed as in (2.15) and solved by PANOC.

$$\Xi(x_k, \lambda_k, \mu_{pa}^k) = \mathcal{F}(x_k) + \frac{\mu_{pa}^k}{2} \left[\|\mathcal{H}(x_k)\|^2 + \min_{\sigma_k \in \mathcal{C}} \|\mathcal{G}(x_k + \frac{1}{\mu_{pa}^k} \lambda_k - \sigma_k)\|^2 \right] \tag{2.15}$$

In each iteration, both the penalty parameter μ_{pa}^k and Lagrange multipliers are updated. The Lagrange multiplier update in equation (2.16) is performed using the projection operator defined as $\Pi_{\mathcal{C}}(x) = \operatorname{argmin}_{y \in \mathcal{C}} \|y - x\|$.

$$\lambda_{k+1} = \Pi_y(\lambda_k) + \mu_{pa}^k \left[\mathcal{G}(x_{k+1}) - \Pi_{\mathcal{C}}(\mathcal{G}(x_{k+1}) + \frac{1}{\mu_{pa}^k} \Pi_y(\lambda_k)) \right] \tag{2.16}$$

The penalty parameter is updated simply as $\mu_{pa}^{k+1} = \xi_p \mu_{pa}^k$. The penalty update factor $\xi_p \in \mathbb{R}$ has to be tuned to optimize the convergence speed of the algorithm. Solving the inner problem of Ξ minimization, the approach presented in [108] is utilized. Newton-type update described by equation (2.17) is used

$$x_{k+1} = x_k - \Lambda_k \mathcal{R}_\Delta(x_k, \lambda_k, \mu_{pa}^k) \tag{2.17}$$

where $\Lambda_k : \mathbb{R}^n \to \mathbb{R}^n$ is an invertible linear operator and $\mathcal{R}_\Delta$ is a fixed point residual operator defined as in equation (2.18) with a fixed constant $\Delta > 0$.

$$\mathcal{R}_\Delta(x_k, \lambda_k, \mu_{pa}^k) = \Delta^{-1} \left[x_k - \Pi_{\mathcal{U}}(x_k - \Delta \cdot \nabla_x \Xi(x_k, \lambda_k, \mu_{pa}^k)) \right] \tag{2.18}$$

PANOC also employs forward-backward envelope to improve convergence when using initial solution x_0, which is further from a global optimum x^*, while achieving a fast iteration evaluation and low memory footprint.

2.6 OPTIMAL CONTROL THEORY FOUNDATIONS

Optimal control theory dates back to 17th century with Johann Bernoulli's famous *brachystochrone* problem, which attempts to find the path along which an object moves between two points in a vertical plane under its own weight in the shortest time. This seemingly simple problem led to beginnings of the optimal control theory. In most practical cases, analytical solutions to OCPs are not available. However, efficient numerical solutions can be obtained by a variety of methods. The trajectory optimization problem is typically solved by the utilization of direct or indirect methods. Indirect methods focus on the construction of the necessary and sufficient conditions for optimality and subsequent discretization of those conditions to a structure of solvable NLP. The direct methods approach the OCP such that the trajectory is directly discretized in the time domain and converted to a parameter optimization problem. Although the indirect approach yields an accuracy metric for the solution, the construction of the adjoint equations can be infeasible for many real-world applications. The direct approach is typically chosen due to its easier formulation and availability of increasingly efficient algorithms for solving them. Optimal control theory can be regarded as the basis for understanding the state-of-the-art GNC algorithms. Solving optimal guidance has been a major topic since the start of space exploration and led to the development of a variety of mathematical methods. On the other hand, in the domains of navigation and control, the optimal control theory allowed for a derivation of the KF for state estimation and LQR for optimal regulation of the system. Although having theoretical limitations, both techniques have been widely used for decades due to their performance as well as their simple implementation. Nowadays, it has become computationally feasible to solve well-posed OCPs very fast, thus promoting estimation and control techniques such as MHE and MPC solving OCPs on the fly. Assuming aerospace or automotive platforms thus means solving OCPs with at least 50 Hz frequency.

Let us consider a general OCP. The goal is to find a control vector $\mathbf{u}(t) \in \mathbb{R}^m$ that minimizes the functional (2.19) referred to as Bolza form.

$$\mathcal{J} = \Phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f, \mathbf{q}) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), \mathbf{q}, t)dt \tag{2.19}$$

The running cost $g : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}$ (Lagrange problem) involves the state vector $\mathbf{x}(t) \in \mathbb{R}^n$ and the control vector along the trajectory. The terminal cost $\Phi : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^l \to \mathbb{R}$ (Mayer problem) involves the state vector at initial $t_0 \in \mathbb{R}$ and final $t_f \in \mathbb{R}$ time. The state vector is subjected to dynamic constraints characterized by an Ordinary Differential Equation (ODE) system (2.20).

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{q}, t), \qquad t \in [t_0, t_f] \tag{2.20}$$

The boundary $\boldsymbol{\Psi} : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^l \to \mathbb{R}^q$ and path constraints $\boldsymbol{\Omega} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}^c$ can be expressed as in equation (2.21).

$$\begin{aligned} \boldsymbol{\Psi}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f, \mathbf{q}) &= 0 \\ \boldsymbol{\Omega}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{q}, t) &\leq 0 \end{aligned} \tag{2.21}$$

with algebraic parameters of the problem $\mathbf{q} \in \mathbb{R}^l$. In contrast to Initial Value Problem (IVP) where the dependent variable is imposed at initial time, the BVP requires the state vector to be determined at both points t_0 and t_f. It can be seen that OCPs in this form involve continuous functions. In practice, solving an OCP requires iterative, typically Newton-based algorithms on a finite set of variables. The process of converting an OCP to a finite-dimensional general NLP problem is called a transcription method [11]. Generally, the transcription method first discretizes the time domain to convert the dynamic constraints to a problem composed of a finite set of variables. The resulting transcribed problem has to be solved using a parameter optimization method, and finally, the accuracy of the finite-dimensional approximation should be assessed. The commonly used transcription methods are briefly summarized in the following paragraphs.

Trapezoidal method

The standard approach in direct transcription methods utilizes a known integration scheme to transcribe the dynamics to a set of equality constraints. Let us assume a variable initial and final times t_0 and t_f. The time domain can then be defined in the interval $0 \leq \tau \leq 1$ by the transformation written in equation (2.22).

$$\tau = \frac{t - t_0}{t_f - t_0} \tag{2.22}$$

The discretization is performed using K equally spaced points in the transformed time domain assuming step size $\delta t = t_f/K$ and transformed time step $\delta\tau = 1/K$. The set of NLP variables defining the OCP can be generally written as in (2.23), where, at time τ_0, the control vector is $\mathbf{u}(\tau_0) = [u_1(\tau_0), u_2(\tau_0) \dots u_m(\tau_0)]$ and the state vector is $\mathbf{x}(\tau_0) = [x_1(\tau_0), x_2(\tau_0) \dots x_n(\tau_0)]$.

$$\mathfrak{X}^T = [u_1(\tau_0), u_2(\tau_0) \dots x_1(\tau_1), x_2(\tau_1) \dots u_m(\tau_K), x_n(\tau_K), t_0, t_f, q_1 \dots q_l] \tag{2.23}$$

Except for initial and final time, the set of l algebraic optimization parameters can be optionally defined as $\mathbf{p} = [p_1, p_2 \dots p_l]$. The dynamic constraints can be formulated using a trapezoidal scheme as in (2.24).

$$\mathbf{x}(\tau_{k+1}) - \mathbf{x}(\tau_k) - \frac{\delta\tau}{2}\left[\mathbf{f}(\mathbf{x}(\tau_{k+1}), \mathbf{u}(\tau_{k+1}), \tau_{k+1}, t_0, t_f) + \mathbf{f}(\mathbf{x}(\tau_k), \mathbf{u}(\tau_k), \tau_k, t_0, t_f)\right] = 0 \tag{2.24}$$

The inconsistency in the constraints can be measured by the so-called defect as a difference between the actual state value and the state which is consistent with the evolving dynamics. A visualization of defects is shown in Figure 2.3. Apart from the chosen trapezoidal scheme, there are various integration schemes well suited for transcription methods. They include Runge-Kutta methods, Hermite-Simpson, and a basic Euler method. The objective function involving path cost function in integral form can be transformed using a trapezoidal integration scheme by (2.25).

$$\mathcal{J} = \mathbf{\Phi}(\mathbf{x}(\tau_0), t_0, \mathbf{x}(\tau_K), t_f) + \frac{t_f - t_0}{2(K-1)} \sum_{k=1}^{K} g(\mathbf{x}(\tau_k), \mathbf{u}(\tau_k), \tau_k, t_0, t_f). \tag{2.25}$$

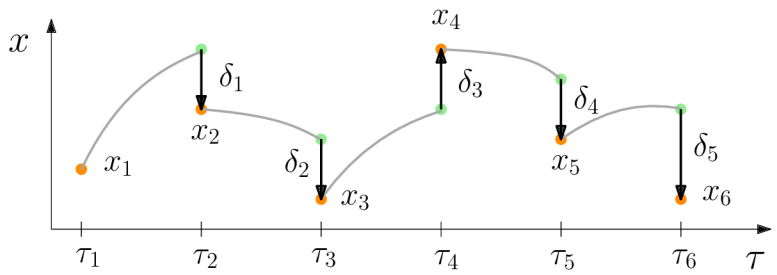

Figure 2.3: Defect visualization over a short time horizon.

The dynamic equality constraints were derived using a trapezoidal integration scheme, and path cost was approximated. The NLP formulation allows for the specification of path constraints (generalized to $\boldsymbol{\Omega}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{q}, t) \leq 0$). The expression typically includes state and control variable constraints (2.26) at time τ_k and path constraints setting bounds on vector function $\boldsymbol{\Omega}$ values.

$$\begin{aligned} \mathbf{x}_{lb} \leq \mathbf{x}(\tau_k) \leq \mathbf{x}_{ub} \\ \mathbf{u}_{lb} \leq \mathbf{u}(\tau_k) \leq \mathbf{u}_{ub} \end{aligned} \tag{2.26}$$

The path constraints take the form given in equation (2.27) assuming both upper and lower bounds.

$$\boldsymbol{\Omega}_{lb} \leq \boldsymbol{\Omega}\left(\mathbf{x}(\tau_k), \mathbf{u}(\tau_k), \mathbf{q}, \tau_k, t_0, t_f\right) \leq \boldsymbol{\Omega}_{ub} \tag{2.27}$$

Once the solution is obtained, the discretization error evaluation has to be performed in order to assess the solution accuracy. The true values of state $\mathbf{x}$ can be computed by integration (2.28).

$$\hat{\mathbf{x}}(t_k + h) = \mathbf{x}(t_k) + \int_{t_k}^{t_k+h} \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt \tag{2.28}$$

The discretization error on the kth interval is then defined by equation (2.29).

$$\eta_k = \max_{i=1}^{n} \{a_i | x_i(t_k + h) - \hat{x}_i(t_k + h)\} \tag{2.29}$$

The idea of discretization error can be extended to metrics such as absolute local error. In all cases, the evaluation of the integral can be computed using a Romberg quadrature algorithm with a tolerance close to machine precision.

First Order Hold method

The First Order Hold (FOH) discretization approach is an efficient interpolating polynomial method for transcription suitable for hardware implementations of OCPs due to its simplicity and control signal discretization. The control signal in FOH is piecewise affine function of normalized time as defined in (2.22) such that $\mathbf{u} : [0\ 1] \to \mathbb{R}^m$. In each time interval $[\tau_k\ \tau_{k+1}]$, a continuous time control signal is obtained by equation (2.30) with $\mathbf{u}_k$ and $\mathbf{u}_{k+1}$ being the control input values at the respective collocation points.

$$\mathbf{u}(\tau) = \frac{\tau_{k+1} - \tau}{\tau_{k+1} - \tau_k}\mathbf{u}_k + \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k}\mathbf{u}_{k+1} \tag{2.30}$$

The notation of (2.30) can be simplified to $\mathbf{u}(\tau) = \Delta_1 \mathbf{u}_k + \Delta_2 \mathbf{u}_{k+1}$. The FOH discretization yields powerful results for Linear Time Varying (LTV) or generally linear dynamics of the state-space form shown in equation (2.31).

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\Delta_1(t)\mathbf{u}_k + \mathbf{B}(t)\Delta_2(t)\mathbf{u}_{k+1} + \mathbf{S}(t)\mathbf{q} + \mathbf{r}(t) \tag{2.31}$$

Given a state transition matrix $\mathbf{F}(\tau, \tau_k) = \mathbf{F}_k$, a solution to (2.31) can be obtained by computing integral in equation (2.32).

$$\mathbf{x}(t) = \mathbf{F}(\tau, \tau_k)\mathbf{x}(\tau_k) + \int_{\tau_k}^{\tau} \mathbf{F}(\tau, \nu)\left[\mathbf{B}(\upsilon)\Delta_1(\upsilon)\mathbf{u}_k\mathbf{B}(\upsilon)\Delta_2(\upsilon)\mathbf{u}_{k+1}\mathbf{S}(\upsilon)\mathbf{q} + \mathbf{r}(\upsilon)\right]d\upsilon \tag{2.32}$$

If $\tau = \tau_{k+1}$, the state is propagated to a desired collocation point. In such case, the obtained discrete version of (2.31) yields the system $\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{G}_k^1\mathbf{u}_k + \mathbf{G}_k^2\mathbf{u}_{k+1} + \mathbf{T}_k\mathbf{q} + \mathbf{s}_k$ with used matrices defined by equation (2.33).

$$\begin{aligned}
\mathbf{G}_k^1 &= \mathbf{F}_k \int_{\tau_k}^{\tau_{k+1}} \mathbf{F}(\upsilon, \tau_k)^{-1}\mathbf{B}(\upsilon)\Delta_1(\upsilon)d\upsilon \\
\mathbf{G}_k^2 &= \mathbf{F}_k \int_{\tau_k}^{\tau_{k+1}} \mathbf{F}(\upsilon, \tau_k)^{-1}\mathbf{B}(\upsilon)\Delta_2(\upsilon)d\upsilon \\
\mathbf{T}_k &= \mathbf{F}_k \int_{\tau_k}^{\tau_{k+1}} \mathbf{F}(\upsilon, \tau_k)^{-1}\mathbf{S}(\upsilon)d\upsilon \\
\mathbf{s}_k &= \mathbf{F}_k \int_{\tau_k}^{\tau_{k+1}} \mathbf{F}(\upsilon, \tau_k)^{-1}\mathbf{r}(\upsilon)d\upsilon
\end{aligned} \tag{2.33}$$

Gauss pseudospectral methods

Pseudospectral methods employ global interpolating polynomials to approximate the state across the entire time interval. Numerical methods including pseudospectral methods can either use a fixed time interval or map a normalized time interval to a scaled one. A transformation of time interval $t \in [t_0, t_f]$ mapped to a general time interval $\tau \in [-1, 1]$ given by (2.34) with free initial and final times is considered.

$$\tau = \frac{2t}{t_f - t_0} - \frac{t_f + t_0}{t_f - t_0} \tag{2.34}$$

To minimize the error in the quadrature approximation of the integral term of the cost function or dynamic constraints discretization, the Gauss quadrature support points are chosen, since Gauss quadrature has the highest accuracy for a given number of points K, and is exact for all polynomials of degree $2K - 1$ or less. The quadrature points are the Legendre Gauss (LG) points [46] defined in the interior of the fixed time interval ($\tau \in (-1, 1)$) as the roots of the Legendre polynomial $\mathcal{P}_K(\tau)$ of degree K in equation (2.35).

$$\mathcal{P}_K(t) = \frac{1}{2^K K!}\frac{d^K}{d\tau^K}\left[\left(\tau^2 - 1\right)^K\right] \tag{2.35}$$

The corresponding Gauss quadrature weights are then found as in equation (2.36).

$$\mathbf{w}_i^g = \frac{2}{1 - \tau_i\left[\dot{\mathcal{P}}_K(\tau_i)\right]^2}, \qquad i = 1, ..., K \tag{2.36}$$

Similarly, another set called Legendre Gauss Radau (LGR) support points on interval $\tau \in [-1, 1)$ are defined as the roots of $\mathcal{P}_K(\tau) + \mathcal{P}_{K-1}(\tau)$. The LGR weights are then given by (2.37).

$$\begin{aligned}
\mathbf{w}_1^g &= \frac{2}{K^2} \\
\mathbf{w}_i^g &= \frac{1}{(1 - \tau_i)\left[\dot{\mathcal{P}}_{K-1}(\tau_i)\right]^2}, \qquad i = 2, ..., K
\end{aligned} \tag{2.37}$$

The third set of support points applicable to the pseudospectral collocation method are the Legendre Gauss Lobatto (LGL) points. The support points lie on the interval $\tau \in [-1, 1]$ including both boundaries, making the quadrature scheme accurate to $2K-3$ polynomial degree. The LGL points are derived as roots of the polynomial $\left(1-\tau^2\right)\dot{\mathcal{P}}_{K-1}(\tau)$. The associated quadrature weights are obtained using equation (2.38).

$$w_i^g = \frac{2}{K(K-1)} \frac{1}{\left[\mathcal{P}_{K-1}(\tau_i)\right]^2}, \qquad i = 1, ..., K \tag{2.38}$$

Pseudospectral methods exploit a rapid convergence rate of the quadrature approximation and are able to provide better approximations with a small number of collocation points.

Having summarized three possible choices for viable support points, the global polynomial transcription can be obtained using a Lagrange polynomial approximation. The choice of Lagrange polynomials has two benefits when used with the quadrature points in the transcription scheme. The function approximation is equal to the true function at the collocation points $\mathbf{x}(\tau_j) = \mathbf{X}(\tau_j)$ and the unique set of support points tends to bypass the so-called Runge phenomenon, which often leads to higher approximation errors near the boundaries when using uniformly spaced support points.

$$\mathcal{L}_i^p(\tau) = \prod_{j=0, j\neq i}^{K} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2.39}$$

Using a basis of $K+1$ Lagrange interpolating polynomials in (2.39), the state and control is approximated in the mapped fixed time domain by equation (2.40).

$$\begin{aligned} \mathbf{x}(\tau) \approx \mathbf{X}(\tau) &= \sum_{i=0}^{K} \mathcal{L}_i^p(\tau)\mathbf{x}(\tau_i) \\ \mathbf{u}(\tau) \approx \mathbf{U}(\tau) &= \sum_{i=1}^{K} \mathcal{L}_i^p(\tau)\mathbf{u}(\tau_i) \end{aligned} \tag{2.40}$$

The control vector polynomial approximation is presented just for consistency purposes. As will be shown, the function can be evaluated only at the set of support points leaving global approximation of the control vector unnecessary. The support points $M = \{\tau_1, ..., \tau_K\}$ (called collocation points) are extended to $M_0 = \{-1, \tau_1, ..., \tau_K\}$ to be of size $K+1$. The discretization point $\mathbf{X}_f$ is collocated at the boundary $\tau_f = 1$. Due to boundary constraints of the original OCP, the final state must be included as a set of discrete variables, despite not being apart of the global polynomial state approximation. The initial and final states can be linked using a Gauss quadrature. The original final state can be integrated as in (2.41) using the initial state.

$$\mathbf{x}(\tau_f) = \mathbf{x}(\tau_0) + \int_{-1}^{1} \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau \tag{2.41}$$

A reformulation of the integration (2.41) using the polynomial approximation and Gauss quadrature leads to a set of equality constraints (2.42).

$$\mathbf{X}(\tau_f) - \mathbf{X}(\tau_0) - \frac{t_f - t_0}{2} \sum_{k=1}^{K} w_k^g \mathbf{f}(\mathbf{X}(\tau_k), \mathbf{U}(\tau_k), \tau_k, t_0, t_f) = 0 \tag{2.42}$$

The mathematical properties of the Gauss quadrature allow for an accurate approximation of the final state. In the Lagrange polynomial reformulation, the dynamic constraints in the time domain interior can be satisfied using the properties of the Lagrange polynomial approximation. The state derivative at LG points for $k = 1, ..., K$ gives a matrix $\mathcal{D}_{ki} \in \mathbb{R}^{K\times(K+1)}$ called LG differentiation matrix.

$$\dot{\mathbf{X}}(\tau_k) = \sum_{i=0}^{K} \dot{\mathcal{L}}_i^p(\tau_k)\mathbf{X}(\tau_i) = \sum_{i=0}^{K} \mathcal{D}_{ki}\mathbf{X}(\tau_k) \tag{2.43}$$

The matrix (2.43) can be determined analytically during the construction of the NLP problem. The LG differentiation matrix computed by (2.44) utilizes properties of the Lagrange interpolating polynomials.

$$\mathcal{D}_{ki} = \dot{\mathcal{L}}_i^p(\tau_k) = \begin{cases} \frac{(1+\tau_k)\dot{\mathcal{P}}_K(\tau_k)+\mathcal{P}_K(\tau_k)}{(\tau_k-\tau_i)[(1+\tau_i)\dot{\mathcal{P}}_K(\tau_k)+\mathcal{P}_K(\tau_k)]}, & i \neq k \\ \frac{(1+\tau_i)\ddot{\mathcal{P}}_K(\tau_i)+2\dot{\mathcal{P}}_K(\tau_i)}{2[(1+\tau_i)\dot{\mathcal{P}}_K(\tau_i)+\mathcal{P}_K(\tau_i)]}, & i = k \end{cases} \tag{2.44}$$

The dynamic constraints can now be composed to a set of equality constraints on a set M. Using the derivative approximation (2.43), we get a set of equality constraints (2.45).

$$\sum_{i=0}^{K} \mathcal{D}_{ki}\mathbf{X}(\tau_k) - \frac{t_f - t_0}{2}\mathbf{f}(\mathbf{X}(\tau_k), \mathbf{U}(\tau_k), \tau_k, t_0, t_f) = \mathbf{0} \tag{2.45}$$

Finally, the Gauss quadrature is utilized to integrate the path cost leading to the final form of the objective function given by equation (2.46).

$$\mathcal{J} = \mathbf{\Phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) + \frac{t_f - t_0}{2}\sum_{k=1}^{K} \mathbf{w}_k^g g(\mathbf{X}(\tau_k), \mathbf{U}(\tau_k), \tau_k, t_0, t_f) \tag{2.46}$$

With minor changes in the handling of the boundary points, the transcription using either LGL or LGR points can be performed.

UAV's trajectory optimization in constrained environments

Varying mission requirements with respect to UAV target location, payload capacity, speed or time of flight combined with environmental constraints such as no-fly zones can be hard to satisfy without the use of modern trajectory optimization techniques. This example, published by the author [76] is focused on the use of pseudospectral methods for UAV mission planning considering known local wind field and constraints such as obstacles with the aim of minimizing energy consumption. The results can be substantial for an extended range of offshore wind farm inspection drones, where the environment is well-suited for this mission planning algorithm. The total energy consumed by the UAV at time t can be derived using the instantaneous power which is then integrated in time to compute the consumed energy.

The objective function (2.47) is derived using the mechanical equation of the Brush-Less Direct Current (BLDC) motor and assuming a quadcopter configuration [124], [125].

$$\mathcal{J} = \int_{t_0}^{t_f} \sum_{j=1}^{N} \left(J_m^j \dot{\omega}_j + \frac{4}{\pi^3} \rho \left[R_{pr}^j \right]^5 \omega_j^2 C_P^j + b_m^j \omega_j \right) \omega_j \, dt \tag{2.47}$$

The objective function assumes the motor angular speed derivatives $\dot{\omega}_j$ to be the control inputs and motor angular speeds ω_j are assumed to be included in the state vector.

Figure 2.4 shows a 3D plot of three trajectories considering varying nominal wind speed, and Figure 2.5 shows their projection to a 2D plot with a wind vector field. It can be seen that with increasing wind speed, the trajectories tend to follow the field.

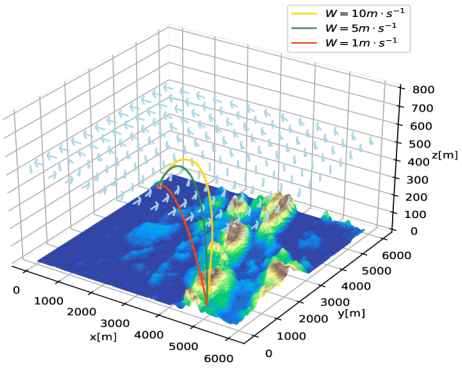

Figure 2.4: Obtained trajectories in a 3D plot.

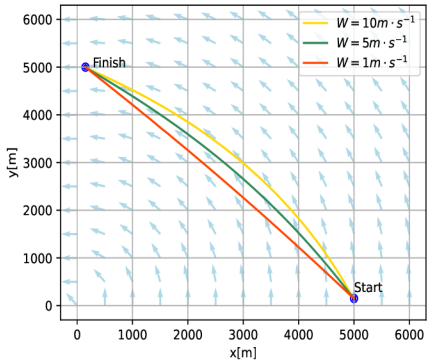

Figure 2.5: Trajectory projection.

Figure 2.6 shows the comparison of all three selected cases with respective velocity profile and battery state of charge. One can see that the battery mass is optimized such that it is discharged exactly to 50 % at the endpoint to satisfy the mission's operational constraint. The trajectories considering the 6-Degrees of Freedom (DoF) dynamic model of the quadcopter were computed by discretizing the transformed time domain to 280 time steps.

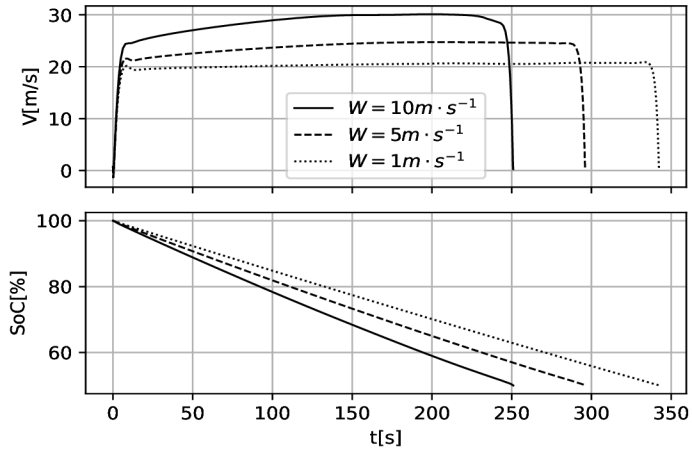

Figure 2.6: The velocity and SoC profiles.

2.7 APPLICATIONS OF OPTIMAL CONTROL THEORY

2.7.1 *Historical perspective: Kalman Filter and Linear Quadratic Regulator*

The state-of-the-art techniques for state estimation are primarily based on the famous KF. The KF was co-invented by Rudolph Kalman in 1958. The method was successfully used on the Apollo GNC computer and has been used in a large variety of applications/domains since then. In contrast to the Wiener filter, the KF operates in the time domain and calculates a weighted average of the measured state and a predicted state using Kalman gain matrix. The Kalman gain is derived from the knowledge of measurement uncertainty. The Kalman filter is a linear optimal state estimator with respect to measurement noise minimal variance. The Kalman filtering technique is also commonly used in combination with LQR to form the LQG control technique. The practical benefits arise from full state knowledge necessity for the state feedback law. Although there are no guaranteed stability margins [27] of the LQG controller, the technique has proven to be of great utility in many state estimation and control applications.

Kalman filtering [44] techniques are based on linear dynamical systems which are discretized in the time domain forming a system with state $\mathbf{x}_k$ and measurement $\mathbf{z}_k$ as in equation (2.48), where $\mathbf{F}_k$ is the state transition matrix, often called the fundamental matrix, $\mathbf{G}_k$ is the control input model, $\mathbf{H}_k$ is the measurement model, and Γ_k is the process noise input matrix.

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{G}_k \mathbf{u}_k + \Gamma_k \mathbf{w}_k \\ \mathbf{z}_k &= \mathbf{H}_k \mathbf{x}_k \mathbf{v}_k \end{aligned} \tag{2.48}$$

The discretization is usually performed by the matrix exponential technique using sample time δt as $\mathbf{F}_k = \mathbf{E} + \mathbf{A}\delta t + \frac{(\mathbf{A}\delta t)^2}{2!} + \dots$, where $\mathbf{A}$ is the continuous time state propagation matrix. The vectors $\mathbf{w}_k$ and $\mathbf{v}_k$ are zero-mean, white uncorrelated noise characteristics of the process and observation with known covariance $\mathbf{Q}_k^{est}$ and $\mathbf{R}_k^{est}$ according to equation (2.49).

$$\begin{aligned} \mathbf{w}_k &\sim \mathcal{N}\left(0, \mathbf{Q}_k^{est}\right) \\ \mathbf{v}_k &\sim \mathcal{N}\left(0, \mathbf{R}_k^{est}\right) \end{aligned} \tag{2.49}$$

In real-time applications, the accuracy of the model and noise covariance affect the performance of the KF. Unmodeled dynamics can degrade the performance or lead to instability of the filter. The goal of the KF is to estimate $\mathbf{x}_k$ based on the knowledge of the underlying dynamics and noisy measurement vector $\mathbf{z}_k$. If all the measurements up to time k are available, the so-called a posteriori estimate $\hat{\mathbf{x}}_{k|k}$ can be formed. If the measurement at time k is not available, a priori estimate $\hat{\mathbf{x}}_{k|k-1}$ can be computed. They can be computed as expected values in equation (2.50).

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \mathbb{E}\left[\mathbf{x}_k \mid \mathbf{z}_1 \dots \mathbf{z}_k\right] \\ \hat{\mathbf{x}}_{k|k-1} &= \mathbb{E}\left[\mathbf{x}_k \mid \mathbf{z}_1 \dots \mathbf{z}_{k-1}\right] \end{aligned} \tag{2.50}$$

A priori estimate is computed before one makes the observation, and the a posteriori estimate is computed after the observation. The filter state is not described solely by the state estimate, but it also stores the covariance matrix of the estimation error described using equation (2.51).

$$\begin{aligned}\mathbf{P}_{k|k} &= \mathbb{E}\left[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^T\right] \\ \mathbf{P}_{k|k-1} &= \mathbb{E}\left[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^T\right]\end{aligned} \tag{2.51}$$

A posteriori estimate is formed by fusing the previous a posteriori state estimate with a difference of a priori model state estimate and current measurement multiplied by the Kalman gain [105]. The KF is initialized with a posteriori state estimate $\hat{\mathbf{x}}_{0|0} = \mathbb{E}[\mathbf{x}_0]$ and a posteriori error covariance matrix $\mathbf{P}_{0|0} = \mathbb{E}\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_{0|0})(\mathbf{x}_0 - \hat{\mathbf{x}}_{0|0})^T\right]$. The predict and update steps alternate in time, updating the a posteriori state estimate when a new measurement is obtained. The prediction step utilizes the last available state estimate to form an a priori state estimate and covariance of the estimation error at the current time step computed using equation (2.52).

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{G}_k\mathbf{u}_k \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{\Gamma}_{k|k-1}\mathbf{Q}_k^{est}\mathbf{\Gamma}_{k|k-1}^T\end{aligned} \tag{2.52}$$

If no new measurements are obtained, the algorithm performs multiple prediction steps so that the estimation algorithm does not freeze awaiting new update. The update step performs the computation of the Kalman gain and a posteriori state estimate by equations (2.53).

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k^{est}\right)^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}\right) \\ \mathbf{P}_{k|k} &= \left(\mathbf{E} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_{k|k-1}\end{aligned} \tag{2.53}$$

As can be seen, computing $\mathbf{P}_{k|k-1}$, $\mathbf{K}_k$ and $\mathbf{P}_{k|k}$ does not depend on the measurement $\mathbf{z}_k$. These calculations can possibly be computed offline and saved to the hardware memory before the operation itself. This benefit will not be possible for EKF and UKF since the system is generally nonlinear. If we assume the state estimation error $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k}$, the KF minimizes the expected value of the estimation error. Even if the noise is not Gaussian, the KF is still the best linear filter.

As was outlined in the derivation of KF, the estimation is optimal under the assumption that the process and measurement models are linear. This limits the applications of the KF significantly. Luckily, there are ways to handle nonlinearity and allow for the use of the KF structure. The simplest way to handle nonlinearity is to linearize the system at the current state estimate. The state estimator following this process is called the EKF [44]. It was one of the first techniques used for nonlinear problems and it is still frequently used today. The challenges associated with the design of EKF are mainly tied to the model accuracy and linearization errors. A general nonlinear dynamical system is described using equations (2.54).

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k)\end{aligned} \tag{2.54}$$

Since the propagation of Gaussian through a nonlinear function is not a Gaussian, the basic principles of the KF are not valid. The linearization at a nominal trajectory can, however,

provide a good estimate of the system behavior. The EKF is initialized in the same way as the KF. The prediction step consists of a state estimate prediction, process linearization, and covariance of the estimation error prediction. In the prediction step, $\mathbf{F}_k$ is computed as a Jacobian matrix at around the previous state estimate. The prediction step can be written by a set of equations (2.55).

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)\\ \mathbf{F}_k &= \left(\frac{\partial f_i(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)}{\partial x_j}\right)_{i,j=1}^{n}\\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k^{est}\end{aligned} \tag{2.55}$$

In the update step, the measurement function is linearized and used within the KF structure as in equations (2.56).

$$\begin{aligned}\mathbf{H}_k &= \left(\frac{\partial h_i(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)}{\partial x_j}\right)_{i,j=1}^{n}\\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k^{est}\right)^{-1}\\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}_k)\right)\\ \mathbf{P}_{k|k} &= (\mathbf{E} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_{k|k-1}\end{aligned} \tag{2.56}$$

The performance of EKF can be improved by selecting a higher order nonlinear function approximation in order to reduce the linearization errors. The EKF depends not only on the computational capabilities of the hardware to compute the Jacobian matrix [19], but also on the initial state estimate, since it propagates in the Jacobian matrix leading to filter divergence if set too incorrectly. To summarize the method, EKF works by linearizing the system at a single point in every time step. That is theoretically sufficient. However, if the state estimate becomes insufficiently accurate, the filter degradation is accelerated by the linearization step. Adapting KF for multisensor and multirate systems has also been investigated by [94] and [50]. This is partly bypassed by the UKF, which uses $2n + 1$ points while avoiding Jacobian matrix computation. The EKF is still considered an industry standard in many domains, despite the discussed difficulties associated with its numerical properties.

The formulation of general OCP has shown a great potential in constrained trajectory optimization using the nonlinear objective function and dynamic constraints. Prior to the development of many numerical techniques capable of solving general OCP, a class of problems utilizing linear dynamic constraints while minimizing a quadratic cost function was shown to have an analytical solution [10]. This led to the development of LQR, an optimal full-state feedback regulator with respect to a defined quadratic cost functional. Let us assume a linear state space system (2.57) where $\mathbf{x}(t)$ is the state vector, $\mathbf{u}(t)$ is the control input vector, and $\mathbf{y}(t)$ is the output vector.

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0\\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)\end{aligned} \tag{2.57}$$

Given the controllable system and full-state information measurements or the observable system with a full-state estimate, the closed loop stability may be achieved by the selection of matrix $\mathbf{K}$, so that eigenvalues of $(\mathbf{A} - \mathbf{B}\mathbf{K})$ are in the left half of the complex plane. The

derivation of LQR regulator assumes an optimal reduction of initial state components $\mathbf{x}_0$ in a time interval $\langle t_0, t_f \rangle$ such that the objective function (2.58) is minimized.

$$\mathcal{J}(\mathbf{x}(t_0), t_0) = \int_{t_0}^{t_f} \left[\mathbf{x}(t)^T \mathbf{Q}_{LQR} \mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}_{LQR} \mathbf{u}(t)\right] dt + \mathbf{x}(t_f)^T \mathbf{Q}^f_{LQR} \mathbf{x}(t_f) \tag{2.58}$$

The matrices $\mathbf{Q}_{LQR}$ and $\mathbf{Q}^f_{LQR}$ are positive semi-definite and matrix $\mathbf{R}_{LQR}$ is positive definite. Using Bellman principle of optimality with an infinitesimally small time interval and using notation in equation (2.59)

$$\begin{aligned} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ g(\mathbf{x}(t), \mathbf{u}(t), t) &= \mathbf{x}(t)^T \mathbf{Q}_{LQR} \mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}_{LQR} \mathbf{u}(t) \\ \Phi(\mathbf{x}(t_f)) &= \mathbf{x}(t_f)^T \mathbf{Q}^f_{LQR} \mathbf{x}(t_f) \end{aligned} \tag{2.59}$$

the optimal $\mathbf{u}^*$ can be obtained by minimizing expression (2.60), where $\mathcal{J}^*(\mathbf{x}(t), t)$ is a minimum value of objective function on interval $\langle t, t_f \rangle$.

$$\mathcal{W}(\mathbf{x}(t), \mathbf{u}(t), t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \left[\frac{\partial \mathcal{J}^*(\mathbf{x}(t), t)}{\partial \mathbf{x}}\right]^T \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.60}$$

To minimize $\mathcal{W}(\mathbf{x}(t), \mathbf{u}(t), t)$, the gradient with respect to the control vector has to be expressed and set to zero. The gradient yields expression (2.61).

$$\begin{aligned} \frac{\partial \mathcal{W}}{\partial \mathbf{u}} &= 2\mathbf{R}_{LQR}\mathbf{u}(t) + \mathbf{B}^T \frac{\partial \mathcal{J}^*(\mathbf{x}(t), t)}{\partial \mathbf{x}} \\ \mathbf{u}^*(t) &= -\frac{1}{2}\mathbf{R}^{-1}_{LQR}\mathbf{B}^T \frac{\partial \mathcal{J}^*(\mathbf{x}(t), t)}{\partial \mathbf{x}} \end{aligned} \tag{2.61}$$

From the gradient expression, it can be seen that the Hessian matrix $\frac{\partial^2 \mathcal{W}}{\partial \mathbf{u}^2} = 2\mathbf{R}_{LQR}$ is positive definite. Control $\mathbf{u}^*$ is, therefore, a minimum of $\mathcal{W}$. The optimal objective value at time t_f is simply a quadratic form $\mathcal{J}^*(\mathbf{x}^*(t_f), t_f) = \mathbf{x}^*(t_f)^T \mathbf{Q}^f_{LQR} \mathbf{x}^*(t_f)$. This extends to arbitrary time t, so that the optimal cost can be computed by equation (2.62), where $\mathcal{P}(t)$ is the solution to the Riccati equation.

$$\mathcal{J}^*(\mathbf{x}(t), t) = \mathbf{x}(t)^T \mathcal{P}(t) \mathbf{x}(t_f) \tag{2.62}$$

The gradient with respect to $\mathbf{x}$ and t is then easily derived resulting in equations (2.63).

$$\begin{aligned} \frac{\partial \mathcal{J}^*(\mathbf{x}(t), t)}{\partial \mathbf{x}} &= 2\mathcal{P}(t)\mathbf{x}(t) \\ \frac{\partial \mathcal{J}^*(\mathbf{x}(t), t)}{\partial t} &= \mathbf{x}^T(t) \frac{d\mathcal{P}}{dt} \mathbf{x}(t) \end{aligned} \tag{2.63}$$

The optimal control at time t can now be written in terms of gain matrix $\mathbf{K}_{LQR}(t)$, which utilizes the solution of Riccati equation $\mathcal{P}(t)$ such that (2.64) is obtained.

$$\begin{aligned} \mathbf{u}^*(t) &= -\mathbf{K}_{LQR}(t)\mathbf{x}(t) \\ \mathbf{K}_{LQR}(t) &= \mathbf{R}^{-1}_{LQR}\mathbf{B}^T \mathcal{P}(t) \end{aligned} \tag{2.64}$$

To find $\mathcal{P}(t)$, equation (2.60) is utilized by substitution of $\mathbf{u}^*(t)$ leading to expression (2.65).

$$\mathcal{W}(\mathbf{x}(t), \mathbf{u}^*(t), t) = \mathbf{x}^T(t) \left(\mathbf{Q}_{LQR} - \mathcal{P}(t)\mathbf{B}\mathbf{R}^{-1}_{LQR}\mathbf{B}^T \mathcal{P}(t) + 2\mathcal{P}(t)\mathbf{A}\right) \mathbf{x}(t) \tag{2.65}$$

From the derivation of minimizing function (2.60), expression (2.66) allows for expressing the Ricatti equation from the minimizing function $\mathcal{W}(\mathbf{x}(t), \mathbf{u}^*(t), t)$.

$$\mathcal{W}(\mathbf{x}(t), \mathbf{u}^*(t), t) = -\mathbf{x}^T(t)\frac{d\mathcal{P}(t)}{dt}\mathbf{x}(t) \tag{2.66}$$

The continuous time Riccati equation is then obtained independently of $\mathbf{x}(t)$ in a form given by equation (2.67).

$$-\frac{d\mathcal{P}(t)}{dt} = \mathbf{Q} - \mathcal{P}(t)\mathbf{B}\mathbf{R}_{LQR}^{-1}\mathbf{B}^T\mathcal{P}(t) + \mathcal{P}(t)\mathbf{A} + \mathbf{A}^T\mathcal{P}(t) \tag{2.67}$$

To solve the Riccati equation, the value $\mathcal{P}(t_f) = \mathbf{Q}_{LQR}^f$ helps to solve the equation backwards in time. However, this is not always necessary, since the infinite horizon extension to linear quadratic problems yields many beneficial properties. In the case of constant matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ in time and observability of all states, the continuous time Ricatti equation converges with $t \to \infty$ to a single solution $\mathcal{P}_\infty$, such that (2.68) holds.

$$0 = \mathbf{Q} - \mathcal{P}_\infty\mathbf{B}\mathbf{R}_{LQR}^{-1}\mathbf{B}^T\mathcal{P}_\infty + \mathcal{P}_\infty\mathbf{A} + \mathbf{A}^T\mathcal{P}_\infty \tag{2.68}$$

The solution $\mathcal{P}_\infty$ can be used to construct the gain matrix leading to a closed loop stability of the system. The quadratic functional of the infinite horizon problem with initial conditions at $t_0 = 0$ yields the expression (2.69) weighting the state error and the control effort.

$$\mathcal{J} = \int_0^\infty \left[\mathbf{x}(t)^T\mathbf{Q}_{LQR}\mathbf{x}(t) + \mathbf{u}(t)^T\mathbf{R}_{LQR}\mathbf{u}(t)\right] dt \tag{2.69}$$

The Riccati equation takes $\mathbf{O}(n^3)$ computations to solve, so it scales with state dimension making it expensive for large systems. Since the solution of the Riccati equation converges to $\mathcal{P}_\infty$, we can express the control law by constant gain matrix given by equations (2.70).

$$\begin{aligned}\mathbf{u}(t) &= -\mathbf{K}_{LQR}\mathbf{x}(t)\\ \mathbf{K}_{LQR} &= \mathbf{R}_{LQR}^{-1}\mathbf{B}^T\mathcal{P}_\infty\end{aligned} \tag{2.70}$$

The matrices $\mathbf{Q}_{LQR}$ and $\mathbf{R}_{LQR}$ balance the cost of effective regulation with the cost of control effort. They weight the deviation of cost or control from zero. Alternatively, they may penalize deviation from an arbitrary state or control vector (equilibrium). LQR control technique is used in applications where full-state feedback is necessary. The technique is limited by the assumption that the dynamics is linear. Some nonlinear systems can be effectively controlled by LQR in the vicinity of an equilibrium point. Highly nonlinear systems are mostly hard to control using standard LQR scheme. However, a gain-scheduling architecture such as in [103] can be utilized to transition among linear controllers. LQR for control and KF/EKF for state estimation benefit from the analytical structure easily implementable on any hardware. In comparison, generalized techniques which are able to handle nonlinear systems such as MPC or MHE require fast on-board nonlinear numerical optimization, which was unfeasible for systems such as UAVs until recently due to fast dynamics.

2.7.2 *Model Predictive Control*

A control technique called MPC is based on predicting a future behavior of the system [101] in a finite time horizon by means of an available model of the system. MPC leverages the optimal control theory perspective to search for control actions that minimize a defined objective function. The structure of the controller allows for the explicit incorporation of safety or operational constraints and explicit bounds on variables of the problem, which motivate its wider use in both industrial and aerospace applications [2]. Typically, each time step of MPC is initialized with the current state estimate and a reference trajectory or setpoint to be tracked. The OCP is solved in a fixed time horizon and only the control inputs in the first time-step from a numerical solution are applied by the actuators. A general formulation of MPC considers solving OCP involving dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ while minimizing $\mathcal{J} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ under a set of constraints on state and input [83]. The OCP is then given by equations (2.71), where $\hat{\mathbf{x}}_0 \in \mathbb{R}^n$ is the current vehicle state estimate, $K \in \mathbb{N}$ is the prediction horizon, and ξ is the applied discretization of system dynamics $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Function $\mathcal{G} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^c$ defines general nonlinear constraints. Lower bounds $\mathbf{x}_{lb} \in \mathbb{R}^n$, $\mathbf{u}_{lb} \in \mathbb{R}^m$ and upper bounds $\mathbf{x}_{ub} \in \mathbb{R}^n$, $\mathbf{u}_{ub} \in \mathbb{R}^m$ of states and control inputs can also be specified. The parameters $\mathbf{q} \in \mathbb{R}^p$ are assumed to be constant in the duration specified by the prediction horizon.

$$
\begin{aligned}
\mathcal{J} = \min_{\mathbf{x}_{0:K}, \mathbf{u}_{0:K-1}} \; & \Phi(\mathbf{x}_0, \mathbf{x}_K, \mathbf{q}) + \sum_{k=0}^{K-1} g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}) \\
\text{subj.to} \quad & \mathbf{x}_0 = \hat{\mathbf{x}}_0 \\
& \mathbf{x}_{k+1} = \xi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}, \mathbf{f}, \delta t) \quad \forall k = 0, \dots, K-1 \\
& \mathcal{G}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}) \leq 0 \quad \forall k = 0, \dots, K-1 \\
& \mathbf{x}_{lb} \leq \mathbf{x}_k \leq \mathbf{x}_{ub} \quad \forall k = 0, \dots, K-1 \\
& \mathbf{u}_{lb} \leq \mathbf{u}_k \leq \mathbf{u}_{ub} \quad \forall k = 0, \dots, K-1
\end{aligned}
\tag{2.71}
$$

The cost function g is often structured to penalize the deviation from the nominal trajectory and control input magnitude in a quadratic sense such as in equation (2.72), where matrices $\mathbf{Q}_{MPC} \in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathbf{R}_{MPC} \in \mathbb{R}^m \times \mathbb{R}^m$ define the weights of the norm.

$$
g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}) = \|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|^2_{\mathbf{Q}_{MPC}} + \|\mathbf{u}_k\|^2_{\mathbf{R}_{MPC}} \tag{2.72}
$$

The structure of (2.71) yields a high versatility such that it solves many issues hard to satisfy using a traditional control approach. It is naturally a Multiple Input Multiple Output (MIMO) controller working in discrete time. MPC has shown to have good properties in controlling systems with model uncertainties or time delays, and most importantly allows for the incorporation of arbitrary constraint. Moreover, the technique is applicable to any model structures including neural networks (black-box models) or hybrid models (gray-box models). A graphical representation of MPC prediction horizon in Figure 2.7 shows a continuously changing reference, state variable and discrete control input for each step. In each time step, a set of optimized control inputs for the time horizon $\mathbf{u}^* = \{\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_K\}$ is computed. Since OCP is solved repeatedly, only the first set of control inputs is applied. The whole OCP solution

may be used to warm-start the numerical optimization solver, thus reducing the number of iterations required for the convergence of the solver. This is a common practice when implementing MPC, as the OCP solution is commonly close for the two consecutive time steps as long as the sampling frequency is high enough.

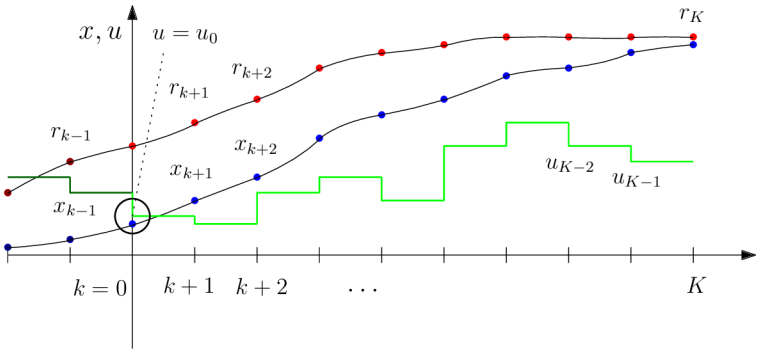

Figure 2.7: Visualization of MPC prediction horizon.

There are many types of MPC controller architectures which vary either in mathematical properties of the problem or the underlying formulation itself, specifying dedicated constraints or cost function terms improving performance, guaranteeing safety, or imposing unique properties to the controller. A linear MPC is defined by linear dynamic constraints as well as linear generic constraints and quadratic cost. MPC formulations guaranteeing safety such as control barrier function MPC or tube MPC have recently gained a lot of popularity. The solution to the linear MPC problem is known to be piecewise affine, which allows for the computation of MPC offline in polyhedric regions. Some MPC problems can also be formulated to be convex, which enables a faster convergence of the optimization algorithm. Still, linear or convex MPC requires the underlying dynamics to be linear. Adaptive MPC techniques can partly solve such issues. Sometimes, a change of the linear model along the trajectory can be predicted. In such case, a LTV-MPC can be used. LTV-MPC uses a different linear dynamic constraint at each time step of the horizon. Path integral MPC [93] leverages the importance sampling scheme to augment the standard MPC. It can thus account for the change of the system's dynamics along a trajectory. Such approach is often effective when a change in model parameters occurs by using linearization around a new operating point. In contrast, nonlinear MPC works with the nonlinear system directly without linearizations, thus requiring more computational resources. The discretization of such OCP can be performed by any chosen method including FOH, Gauss Pseudospectral methods or a trapezoidal method among many others. A nonlinear solver IPOPT is often the first choice when formulating nonlinear problems. Recently, a dedicated solver PANOC was shown to outperform IPOPT and other solvers showing viability for embedded applications. MPC problems may be solved either by assuming a single shooting method, where only control variables are optimized and the dynamics is enforced by a set of functions which are substituted into the objective function or multiple shooting approach. For multiple shooting, the states are also treated as variables of the problem requiring the inclusion of dynamic equality constraints. An increasing number of constraints may result in a problem formulation that cannot yet be accurately solved on embedded platforms at a desired sampling rate of the control system.

Hybrid Modeling Approach for Optimization Based Control of Multirotor UAVs

Optimization-based techniques such as MPC often rely on simplified governing equations of the system omitting complex interactions which are difficult to accurately model, or pose numerical difficulties for the optimization problem solver. This example investigates a hybrid modeling approach based on Sparse Identification of Nonlinear Dynamics (SINDy) [17] for local model improvement within the MPC framework. The presented hybrid modeling approach benefits from the known structure of a physics-based model such that the learning process is computationally lightweight. Numerical experiments assume a multirotor UAV subject to external phenomena such as wind gusts, which are typically encountered in urban environments. Simplified modeling of forces and moments introduces discrepancies between the prediction and true model. However the underlying 6-DoF Equations of Motion (EoM) always stay valid. From additive properties of the forces and moments, it can be assumed that the uncertainties in model parameters, external effects, and unmodeled dynamics can be augmented to the simplified system as shown in equation (2.73).

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \Delta(\mathbf{x}, \mathbf{u}) \tag{2.73}$$

The term Δ represents the discrepancy which is learned locally by the SINDy algorithm.

The performed numerical simulations were conducted assuming a classic MPC and MPC with a locally updated model by SINDy. Figure 2.8 shows the respective responses in varying wind gust magnitude, where solid lines indicate the response of augmented MPC and dashed lines indicate the standard MPC scenario.

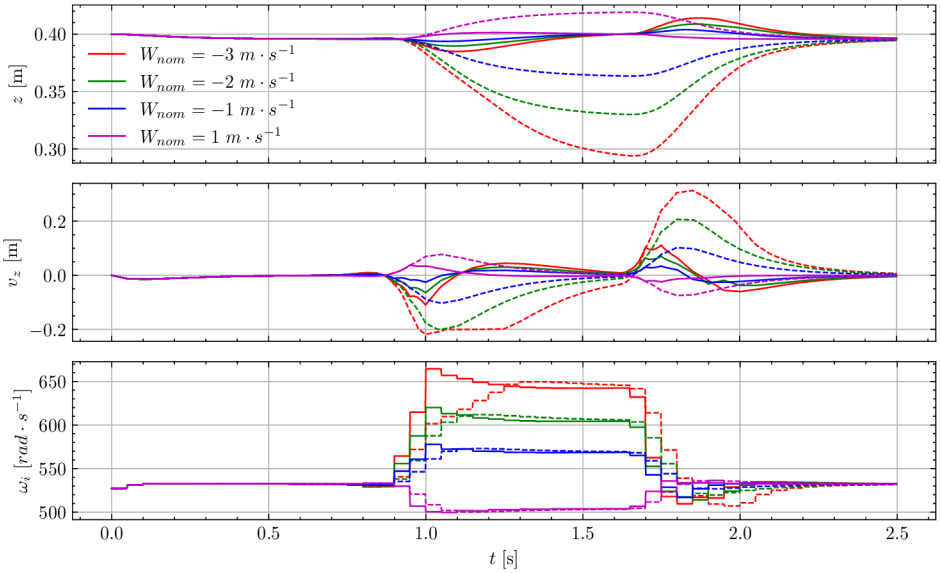

Figure 2.8: Hover stabilization by MPC and SINDy augmented MPC.

The locally augmented model in the MPC framework managed to compensate for the discrepancy resulting in a lower deviation from the nominal hover condition compared to standard MPC reaching position deviation exceeding 0.1 m in a single test case. The results suggest that standard MPC compensates for the gust disturbances much slower since the simplified model does not account for the disturbance in its predictions.

2.7.3 *Moving Horizon Estimation*

A state estimation technique called MHE can be thought of as a counterpart of MPC and generalization of the optimal state estimation often realized by Kalman filters. It has been a heavily researched topic [117], [98] in the recent years due to its general structure allowing for the direct inclusion of a nonlinear prediction model with the possibility of augmenting it by an ANN-based discrepancy model [47]. The goal of MHE is to infer a current state of the system based on a series of past measurements. The structure of MHE is optimization based, meaning that the state estimate is computed based on the minimization of a defined objective function. Estimation horizon is critical in an MHE algorithm setup since it defines the amount of information the estimator works with. However, large estimation horizons lead to higher computational resource requirements, which are oftentimes not feasible with regards to the available hardware and the sampling frequencies of the sensors. A general formulation of the MHE problem is defined by equations (2.74), where $\mathbf{f}$ is the system dynamics, $\mathbf{h}$ is the measurement model function, $\mathbf{w}_k \in \mathbb{R}^n$ represents the process noise vector at step k, and $\mathbf{v}_k$ represents the measurement noise vector at step k. Parameters $\mathbf{q}$ are assumed to be constant in the estimation horizon. The estimation horizon K is bound to horizon duration T by a sampling interval $\delta t \in \mathbb{R}$ such that $T = (t_k - t_{k-1})K = \delta t K$. By shifting the time window, the estimation problem can be repeatedly, efficiently solved by incorporating a new measurement and initializing the solver using the current solution.

$$
\begin{aligned}
\mathcal{J} = \min_{\mathbf{x}_{0:K}, \mathbf{u}_{0:K-1}, \mathbf{q}} \; & \Phi(\mathbf{x}_0, \mathbf{x}_K, \mathbf{q}) + \sum_{k=0}^{K-1} g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}, \mathbf{w}_k, \mathbf{v}_k) \\
\text{subj.to} \quad & \mathbf{x}_{k+1} = \xi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}, \mathbf{f}, \delta t) + \mathbf{w}_k \qquad \forall k = 0, \dots, K-1 \\
& \mathbf{z}_k = \xi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}, \mathbf{h}, \delta t) + \mathbf{v}_k \qquad \forall k = 0, \dots, K-1 \\
& \mathcal{G}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}) \leq 0 \qquad \forall k = 0, \dots, K-1
\end{aligned}
\tag{2.74}
$$

The Mayer term Φ and Lagrange cost function g in MHE generally constitute a quadratic objective structure defined by equations (2.75).

$$
\begin{aligned}
\Phi(\mathbf{x}_0, \mathbf{x}_K, \mathbf{q}) &= \frac{1}{2}\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|^2_{\mathbf{Q}^x_{MHE}} + \frac{1}{2}\|\mathbf{q} - \hat{\mathbf{q}}\|^2_{\mathbf{Q}^q_{MHE}} \\
g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}, \mathbf{w}_k, \mathbf{v}_k) &= \frac{1}{2}\|\mathbf{v}_k\|^2_{\mathbf{Q}^v_{MHE}} + \frac{1}{2}\|\mathbf{w}_k\|^2_{\mathbf{Q}^w_{MHE}}
\end{aligned}
\tag{2.75}
$$

The Mayer term accounts for measurements outside the prediction horizon by imposing the coherence of the previous state estimate with the initial state of the sequence. The Mayer term also penalizes the change in the estimated parameters of the system. The Lagrange term accumulates the cost of noise variables trying to minimize their norm while still satisfying the dynamic constraints and the measurement model [118]. The norms of all present terms are weighted by matrices $\mathbf{Q}^x_{MHE}$, $\mathbf{Q}^p_{MHE}$, $\mathbf{Q}^v_{MHE}$ and $\mathbf{Q}^w_{MHE}$. The MHE problem can be solved in the same manner as MPC by numerical optimization methods such as IPOPT or PANOC.

2.7.4 *Convex Methods for Nonlinear Optimal Control*

Convex optimization has proved to be fast and reliable in many industrial and research applications [127], [109]. This is to be attributed to the inherent mathematical properties of convex optimization problems. Unlike NLP problems, a convex problem can be solved in polynomial-time complexity to a desired accuracy. Unfortunately, most real-world optimization problems are not convex and often include nonlinear dynamics in aerospace engineering mathematical models. Sequential Convex Programming (SCP) is based on solving a sequence of convex approximations of the original non-convex problem. Historically, the development of SCP methods has been largely affected by the development of McCormick relaxations. In reference to optimal guidance, SCP methods have been largely investigated and tested in the scope of autonomous rocket landing problems by companies such as SpaceX and Blue Origin leading to a large-scale reusability of rocket boosters. Generally speaking, SCPs are well suited for safety-critical applications in aerospace and automotive engineering due to theoretical guarantees and performances of convex optimization solvers such as Embedded Conic Solver (ECOS) [26] or Interior point Optimizer for Linear Programming (MOSEK) [3]. Concerning applications considering optimal control, SCP problems resemble trust region methods, since they perform a local optimization around a reference solution. There is a wide variety of SCP methods including Sequential Linear Programming (SLP), Sequential Quadratic Programming (SQP), or more recently Successive Convexification (SCvx) and Guaranteed Sequential Trajectory Optimization (GuSTO) algorithms. For the purposes of fast trajectory generation and obstacle avoidance, the SCvx algorithm has been examined and implemented.

The SCvx algorithm [67], [66] is an SCP adaptation solving a series of local convex approximations around a reference solution representing a trajectory. The initial reference solution $\{\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \tilde{\mathbf{q}}\}$ defines a set of variables (an often infeasible solution) on the normalized time interval $\tau_n \in [0\ 1]$. The initial solution is either generated by linear interpolation using the fixed boundary conditions or by other feasible approximations [35].

In each step of the SCvx algorithm, a convex subproblem in the discrete form listed by (2.78) has to be solved. The matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{S}$ represent the Jacobians of the dynamics with respect to state, control, and parameter variables. The matrices $\nabla_x \boldsymbol{\Psi}$ and $\nabla_q \boldsymbol{\Psi}$ are the Jacobians of the initial boundary condition, and $\nabla_x \boldsymbol{\Psi}$ and $\nabla_q \boldsymbol{\Psi}$ are the Jacobians of the final boundary condition. The dynamic error term is given by equation (2.76).

$$\mathbf{r} = \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \tilde{\mathbf{q}}) - \mathbf{A}\tilde{\mathbf{x}} - \mathbf{B}\tilde{\mathbf{u}} - \mathbf{S}\tilde{\mathbf{q}} \tag{2.76}$$

The constraint error term is given by equation (2.77) requiring the Jacobians of the non-convex path constraints.

$$\tilde{\mathbf{r}} = \boldsymbol{\Omega}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \tilde{\mathbf{q}}) - \nabla_x \boldsymbol{\Omega}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \tilde{\mathbf{q}})\tilde{\mathbf{x}} - \nabla_u \boldsymbol{\Omega}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \tilde{\mathbf{q}})\tilde{\mathbf{u}} - \nabla_q \boldsymbol{\Omega}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \tilde{\mathbf{q}})\tilde{\mathbf{q}} \tag{2.77}$$

In a similar manner, the error terms $\mathbf{l}_0$ and $\mathbf{l}_f$ are derived for the boundary constraints. The dynamic constraint in (2.78) assumes FOH discretization of the underlying state space model.

$$\begin{aligned}
\mathcal{J} = \min_{\mathbf{u},\mathbf{q},\hat{\boldsymbol{\nu}}} \; & \Xi_\lambda(\mathbf{x},\mathbf{u},\mathbf{q},\hat{\boldsymbol{\nu}}) \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{G}_k^1\mathbf{u}_k + \mathbf{G}_k^2\mathbf{u}_{k+1} + \mathbf{T}_k\mathbf{q} + \mathbf{O}_k\boldsymbol{\nu}_k + \mathbf{s}_k \\
& \nabla_x\boldsymbol{\Omega}_k\mathbf{x}_k + \nabla_u\boldsymbol{\Omega}_k\mathbf{u}_k + \nabla_p\boldsymbol{\Omega}_k\mathbf{q} + \tilde{\mathbf{r}}_k \leq \boldsymbol{\nu}_{s,k} \\
& \nabla_x\boldsymbol{\Psi}(\tilde{\mathbf{x}}_0,\tilde{\mathbf{q}})\mathbf{x}_0 + \nabla_q\boldsymbol{\Psi}(\tilde{\mathbf{x}}_0,\tilde{\mathbf{q}})\mathbf{q} + \mathbf{l}_0 + \boldsymbol{\nu}_{ic} = 0 \\
& \nabla_x\boldsymbol{\Psi}(\tilde{\mathbf{x}}_f,\tilde{\mathbf{q}})\mathbf{x}_f + \nabla_q\boldsymbol{\Psi}(\tilde{\mathbf{x}}_f,\tilde{\mathbf{q}})\mathbf{q} + \mathbf{l}_f + \boldsymbol{\nu}_{tc} = 0 \\
& \|\delta\mathbf{x}_k\|_2 + \|\delta\mathbf{u}_k\|_2 + \|\delta\mathbf{q}\|_2 \leq \eta
\end{aligned} \tag{2.78}$$

The constraint inequality mitigating artificial unboundedness is handled by setting a trust region parameter η updated in each SCP iteration, such that change in state, control, and parameters stays within a specified bounded region. Finally, the virtual control terms $(\boldsymbol{\nu}, \boldsymbol{\nu}_s, \boldsymbol{\nu}_{ic}, \boldsymbol{\nu}_{tc})$ are added to handle artificial infeasibility. The matrix $\mathbf{E}$ can be arbitrary as long as $(\mathbf{A}, \mathbf{O})$ is controllable. The cost function in (2.78) is typically augmented by the virtual control penalty term to force the virtual terms closer to zero. Note that $\Xi_\Delta(\mathbf{x},\mathbf{u},\mathbf{q},\hat{\boldsymbol{\nu}})$ is a discretized version of the original cost function defined by (2.79) with modified Mayer and Lagrange costs (2.80) adapted to help the convergence of virtual control terms. The weight $\Delta > 0$ has to be empirically selected.

$$\mathcal{J}_\Delta(\mathbf{x},\mathbf{u},\mathbf{q},\hat{\boldsymbol{\nu}}) = \Phi_\Delta(\mathbf{x}_K,\mathbf{q},\boldsymbol{\nu}_{ic},\boldsymbol{\nu}_{tc}) + \int_0^1 g_\Delta(\mathbf{x},\mathbf{u},\mathbf{q},\mathbf{O}\boldsymbol{\nu},\boldsymbol{\nu}_s)dt \tag{2.79}$$

$$\begin{aligned}
\Phi_\Delta(\mathbf{x}(1),\mathbf{q},\boldsymbol{\nu}_{ic},\boldsymbol{\nu}_{tc}) &= \Phi(\mathbf{x}(1),\mathbf{q}) + \Delta\cdot(\|\boldsymbol{\nu}_{ic}\|_1 + \|\boldsymbol{\nu}_{tc}\|_1) \\
g_\Delta(\mathbf{x},\mathbf{u},\mathbf{q},\mathbf{O}\boldsymbol{\nu},\boldsymbol{\nu}_s) &= g(\mathbf{x},\mathbf{u},\mathbf{q}) + \Delta\cdot(\|\mathbf{O}\boldsymbol{\nu}\|_1 + \|\boldsymbol{\nu}_s\|_1)
\end{aligned} \tag{2.80}$$

The SCvx trust region update rule is based on measuring the convexification accuracy [111]. The metric defined by equation (2.81) defines four possible regions split by constants $\zeta_0,\ \zeta_1,\ \zeta_2 \in (0\ 1)$. The update is performed, such that if $\zeta < \zeta_0$, the trust region becomes $\eta^t = \max(\eta_0, \eta/\beta_{sh})$, where η_{lb}^t is the minimum trust region and β_{sh} is a shrinking constant. In comparison, when $\zeta \geq \zeta_2$, the trust region is updated as $\eta_{k+1}^t = \min(\eta_{ub}, \beta_{gr}\eta_k^t)$, where η_{ub}^t is the maximum trust region and β_{gr} is the growing constant.

$$\zeta = \frac{\mathcal{J}_\Delta(\tilde{\mathbf{x}},\tilde{\mathbf{u}},\tilde{\mathbf{q}}) - \mathcal{J}_\Delta(\mathbf{x}^*,\mathbf{u}^*,\mathbf{q}^*)}{\mathcal{J}_\Delta(\tilde{\mathbf{x}},\tilde{\mathbf{u}},\tilde{\mathbf{q}}) - \Xi_\Delta(\mathbf{x}^*,\mathbf{u}^*,\mathbf{q}^*,\hat{\boldsymbol{\nu}}^*)} \tag{2.81}$$

The $\mathcal{J}_\Delta(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{q}})$ and $\mathcal{J}_\Delta(\mathbf{x}^*,\mathbf{u}^*,\mathbf{q}^*)$ are nonlinear cost functions evaluated at a reference and the newly obtained trajectory respectively, and includes the defect cost, which measures the accuracy in satisfying the evolving dynamics. The term $\Xi_\Delta(\mathbf{x}^*,\mathbf{u}^*,\mathbf{q}^*,\hat{\boldsymbol{\nu}}^*)$ is the subproblem cost function value at its found optimum.

Surrogate Modeling of Optimal Control Based Collision Avoidance System for Multi-rotor UAVs

A dynamically changing operating environment, along with constraints imposed through applicable safety requirements, pose significant challenges to autonomous multirotor manned and unmanned aerial vehicle operations in urban areas. Safety-critical on-board collision avoidance capability requires a fast decision making accounting for uncertainties arising in complex environments. SCvx approach is applied to generate collision avoidance trajectories assuming both static and moving obstacles [75]. The dynamics utilized by the guidance algorithm is simplified, such that only translational dynamics given by (2.82) is utilized.

$$\begin{aligned} \dot{\mathbf{p}}_N &= \mathbf{v}_N \\ \dot{\mathbf{v}}_N &= \frac{1}{m}\left(\mathbf{u}_N - \frac{1}{2}\rho C_D A_D \mathbf{v}_N \parallel \mathbf{v}_N \parallel_2\right) + \mathbf{g}_N \end{aligned} \tag{2.82}$$

However, since the thrust control input $\mathbf{u}^e$ is given in the inertial reference frame, its components constrain a possible set of orientations of the vehicle, since the thrust always acts in the body z_B direction. The assumed obstacles can be modeled as ellipsoidal no-fly zones in a three-dimensional space as given by equation (2.83), where $\tilde{\mathbf{p}}$ represents the center of the ball constraint $\mathcal{B}$.

$$\|\mathcal{B}_j(\mathbf{p}_N(t) - \tilde{\mathbf{p}}_j(t))\|_2 \geq 1, \; j = 1, \ldots, N_{obs} \tag{2.83}$$

The moving obstacles are either modeled by a constant velocity or constant acceleration models, which are typically used within object detection and tracking algorithms.

Figure 2.9 shows a computed collision avoidance trajectory considering both a static and a moving obstacle. The collocation points are colored based on their distance from the moving obstacle center, and each collocation point is assigned with a thrust direction vector of the UAV required to follow the trajectory.

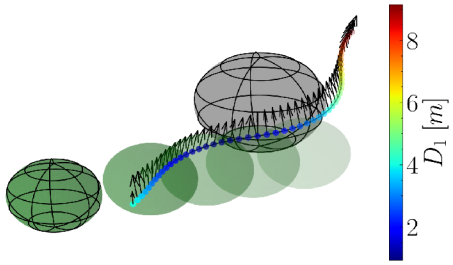

Figure 2.9: UAV trajectory avoiding both static and moving obstacles.

Paper [75] suggests the use of the uncertainty quantification method in the context of uncertainties given by the tracking of the dynamic object. The mean trajectory with respect to kinematic uncertainties of the moving obstacle can, therefore, be used by the vehicle to robustly and efficiently perform the maneuver with a higher error margin.

2.8 SUMMARY AND SOLVER OVERVIEW

Optimization problems arising in GNC applications are often nonlinear, thus requiring dedicated numerical optimization solvers that meet the requirements of on-board computational devices. Although there are many available numerical optimization solvers, the author has identified the solvers listed in Table 2.1 to be highly relevant to solve nonlinear OCPs efficiently. SQP is regarded to be a workhorse of many commercial and open-source nonlinear solvers. It is thus theoretically well studied and documented with a high number of applications. SQP solves a sequence of optimization subproblems with quadratic approximation of the objective in every iteration. For constrained problems, the solver can experience convergence issues requiring more iterations than expected. IPOPT is a highly flexible open-source solver appli-

Table 2.1: Overview of numerical optimization solvers used for Optimal Control.

Solver	Reference	main advantage	main disadvantage
SQP	[15]	fast and proven technique	inner iterative procedures
IPOPT	[116]	robust to initial guess	computational speed
PANOC	[108]	convergence speed for MPC	specific problem structure
ECOS	[26]	embedded applications	only for convex problems
MOSEK	[3]	convergence guarantees	licensed convex solver

cable to large-scale nonlinear problems as well as small-size MPC problems with a quadratic objective function. Although robust to initial guess, IPOPT does not guarantee convergence of nonlinear problems. It has also performance limitations with regards to real-time optimal control applications such as MPC and MHE. Both IPOPT and many SQP solvers can be interfaced using a variety of frameworks such as CasADi [4]. The advantage of using a framework like CasADi is their efficient automatic differentiation (Appendix B) and symbolic framework enabling significant computational time savings when deployed. In comparison, PANOC is a solver tailored for embedded MPC applications focusing on computational efficiency and mitigating the use of computationally heavy operations. It is a part of an open-source solver OpEn [107] and has shown performance improvements over both SQP and IPOPT solvers. The solvers ECOS and MOSEK are well-known convex solvers, which have not been developed with considerations to solving OCPs. However, recent advancements in convexification methods enabled using these solvers on some classes of OCPs with guaranteed convergence, which is a highly desirable requirement for safety-critical applications. An open-source framework called CVXPY [25] facilitates the mentioned solvers. Furthermore, ECOS solver is suitable for embedded applications with its low memory footprint and efficient implementation. The second listed convex solver is MOSEK, which yields a similar performance like ECOS, but has been more robust to numerical issues due to its continuous maintenance and updates. It is licensed for commercial use, and free for academic users. Although convex optimization algorithms gain more importance with regards to GNC applications, the flexibility of the problem structure is not that high compared to general NLP solvers. For comparison, the solvers SQP, IPOPT and PANOC were chosen to evaluate the performance with regards to convergence times on a

selected OCP designed for MPC control of a quadcopter UAV configuration. The assumed OCP was constructed as a single-shooting problem optimizing the control actions in the specified prediction horizon. All computations were performed on a personal computer using Intel(R) Core(TM) i7-10510U CPU and 16 GB RAM. Figure 2.10 shows a statistical evaluation of all the solvers on the given reference problem using 1000 randomly sampled initial conditions and varying the number of collocation points, thus increasing the problem size. It can be seen that PANOC outperforms other solvers on all selected test cases. The evaluation suggests that MPC problems with a lower number of discretization points are viable for real-time applications.

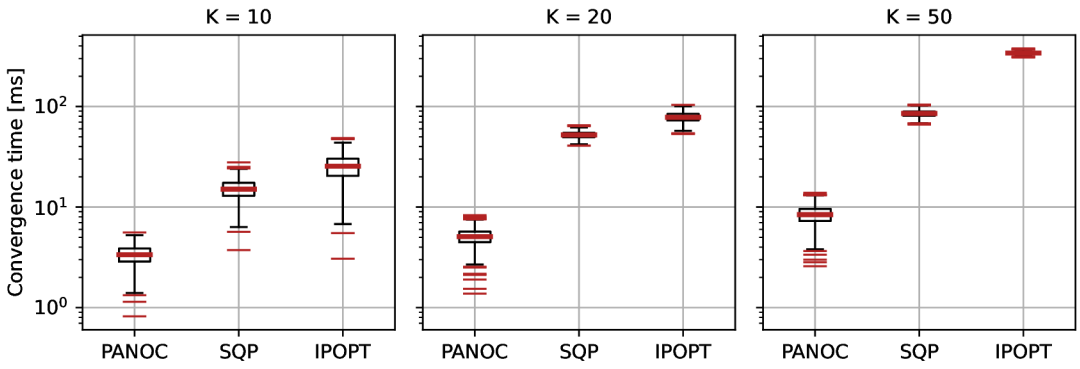

Figure 2.10: Convergence time comparison of three selected solvers.

The solver PANOC was, therefore, investigated further with respect to convergence time on a personal computer: Raspberry Pi4 and Raspberry Pi5 mini computers. As shown by the statistical evaluation in Figure 2.11, even though the performance drop on the Raspberry platform is substantial, it is still satisfactory for a real-time control of robotic or UAV systems which require update frequency of at least 50–100 Hz based on the considered platform. Multirotor UAVs have a typically faster rotational dynamics compared to fixed-wing UAVs, thus requiring a faster stabilization of the attitude.

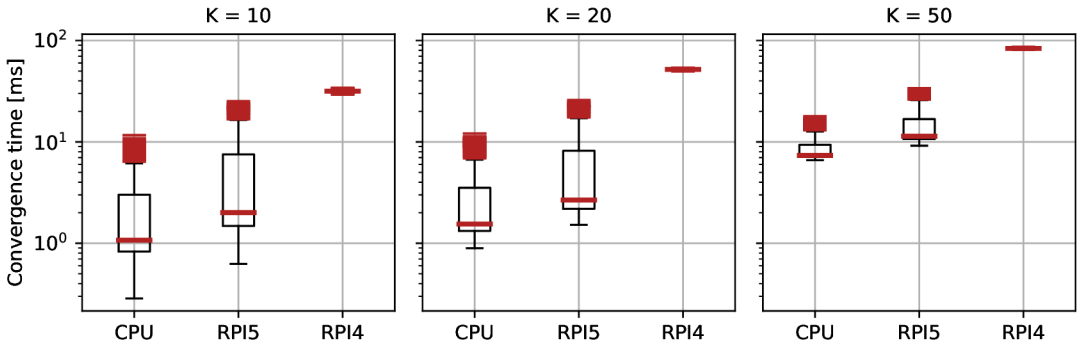

Figure 2.11: Convergence time comparison on three selected platforms.

The choice of the numerical optimization algorithm for solving OCP depends on many factors such as the problem size, mathematical properties of the problem, convergence speed requirements, and many other considerations. Therefore, it requires expert knowledge of mathematical modeling and optimization methods, as well as domain knowledge.

3 FIRST PRINCIPLE BASED MODELING

The first chapter introduced a theoretical background needed to study and implement state-of-the-art GNC systems. As was already mentioned, knowledge of system dynamics plays a key role in GNC system development. The first-principle based modeling approach makes use of the governing laws of motion to establish close-to-reality models while mitigating discrepancies as much as possible. This thesis focuses on multirotor UAVs and their possible extensions such as integrating a robotic arm to the airframe enabling aerial grasping missions. The range of multirotor UAV designs has grown significantly over the last decade. This is to be attributed to their simple mechanical design, along with hovering and maneuvering capabilities, making them a popular choice in applications such as surveillance, inspection, reconnaissance, and data acquisition over remote areas. Major challenges in a large-scale deployment of UAVs in urban areas for applications such as last mile delivery have been observed due to safety requirements and current legislation. Varying mission

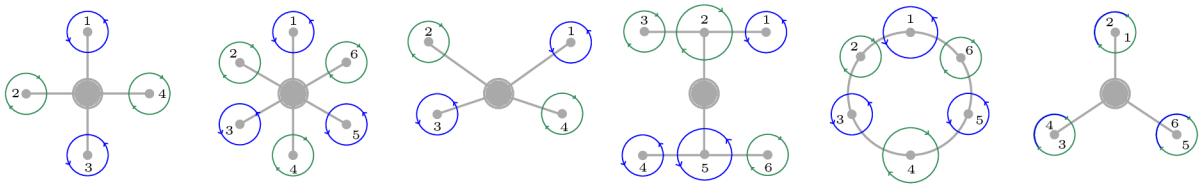

Figure 3.1: Range of multirotor UAV configurations.

requirements with respect to a UAVs' robustness, payload capacity, maximum speed, and time of flight have extended the number of potentially useful multirotor UAV configurations [59] beyond the widely used quadcopter configuration (see Figure 3.1). For example, a hexacopter configuration has been used in applications where an increased robustness due to propeller redundancy is necessary for safe operation [96]. The utilization of contra-rotating propellers can further increase efficiency and improve control due to gyroscopic torque cancellation. A generalized multirotor UAV model was implemented by the author within the simulation framework developed under the European Commission funded COMP4DRONES project. In the following sections, a standard mathematical model of UAV dynamics will be described, the propeller aerodynamic forces and moments derived, and platform independent environment effects including wind field, wind turbulence, and wind gusts described with the emphasis on operations within urban environments. This chapter works with the following assumptions:

1. No flexible parts are assumed (rigid body dynamics).
2. All propellers are oriented along or parallel to the vehicle z_B-axis (no tilting propellers).
3. The model is designed to be differentiable (no absolute value function, etc.).
4. The UAV operation is dependent on an electric energy source (battery).

3.1 MULTIROTOR UNMANNED AERIAL VEHICLE MATHEMATICAL MODELING

Let us assume a general air-frame configuration of multirotor UAV. The 6-DoF EoM evolve in a local inertial reference frame $\mathcal{F}_N = (x_N, y_N, z_N, O_N)$ respecting the North-East-Down (NED) convention and Body-Fixed Frame (BFF) coordinates $\mathcal{F}_B = (x_B, y_B, z_B, O_B)$, both specified in Appendix A. The translation dynamics is given by a set of first-order ODEs [23] in form (3.1), where $\mathbf{p}_N \in \mathbb{R}^3$ is the position vector of the Center of Gravity (CoG) expressed in $\mathcal{F}_N$, vector $\mathbf{v}_B \in \mathbb{R}^3$ is a translation velocity of $\mathcal{F}_B$ with respect to $\mathcal{F}_N$ expressed in $\mathcal{F}_B$, vector $\boldsymbol{\omega}_B \in \mathbb{R}^3$ is the angular velocity of $\mathcal{F}_B$ with respect to $\mathcal{F}_N$ expressed in $\mathcal{F}_B$, and $\mathbf{R}_{N/B}$ is the transformation matrix between BFF and NED in the form of Euler angle rotation matrix or a quaternion-based Direction Cosine Matrix (DCM).

$$\begin{aligned} \dot{\mathbf{p}}_N &= \mathbf{R}_{N/B}\mathbf{v}_B \\ \dot{\mathbf{v}}_B &= m^{-1}\mathbf{F}_B - \boldsymbol{\omega}_B \times \mathbf{v}_B + \mathbf{R}_{B/N}\,\mathbf{g}_N \end{aligned} \tag{3.1}$$

The vector $\mathbf{g}_N \in \mathbb{R}^3$ refers to acceleration due to gravity, $\mathbf{F}_B = [F_x\ F_y\ F_z] \in \mathbb{R}^3$ are the forces acting on the vehicle CoG, and m is the vehicle mass. The rotation dynamics is given by equations (3.2), where $\mathbf{q}_{B/N} \in \mathbb{R}^4$ is a quaternion representation of the vehicle $\mathcal{F}_B$ orientation with respect to $\mathcal{F}_N$, the skew-symmetric matrix $[\boldsymbol{\omega}_B\times] \in \mathbb{R}^{3\times3}$ is defined by equation (3.3), and $\boldsymbol{\Omega}^s(\boldsymbol{\omega}_B) \in \mathbb{R}^{4\times4}$ is a skew-symmetric matrix defined by equation (3.4).

$$\begin{aligned} \dot{\mathbf{q}}_{B/N} &= \frac{1}{2}\boldsymbol{\Omega}^s(\boldsymbol{\omega}_B)\mathbf{q}_{B/N} \\ \mathbf{I}_B\dot{\boldsymbol{\omega}}_B &= \mathbf{M}_B - [\boldsymbol{\omega}_B\times]\,\mathbf{I}_B\boldsymbol{\omega}_B \end{aligned} \tag{3.2}$$

The inertia tensor $\mathbf{I}_B \in \mathbb{R}^{3\times3}$ is usually simplified due to the design's symmetry of a UAV body. The rotational dynamics is driven by a moment vector $\mathbf{M}_B = [M_x\ M_y\ M_z] \in \mathbb{R}^3$.

$$[\boldsymbol{\omega}_B\times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.3}$$

The air-frame configuration and architecture of any UAV can be described by a collection of $k = 1, \ldots, K_p$ components such as frame parts, propellers, payload, and batteries.

$$\boldsymbol{\Omega}^s(\boldsymbol{\omega}_B) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{3.4}$$

Each of the components has its mass m^k and inertia tensor $\mathbf{I}^k$ expressed in the frame of the respective part converted to BFF as $\mathbf{I}^k_B = \mathbf{I}^k + m^k\left[(\mathbf{p}^k_B \cdot \mathbf{p}^k_B)\,\mathbf{E} - \mathbf{p}^k_B \otimes \mathbf{p}^k_B\right]$, where $\mathbf{p}^k_B \in \mathbb{R}^3$ is the displacement vector of the part center of mass to a model center of mass, and $\mathbf{E} \in \mathbb{R}^{3\times3}$ is an identity matrix. The final mass is then computed as $m = \sum_{k=1}^{K} m^k$, and the inertia tensor as $\mathbf{I}_B = \sum_{k=1}^{K} \mathbf{I}^k_B$.

Let us now describe the forces and moments acting on the UAV. A popular quadcopter configuration is often modeled in the literature [112]. The generalization to N propeller UAV will be further described. The resulting force $\mathbf{F}_B$ acting on the UAV CoG was decomposed to several components. Based on UAV configuration and the required model accuracy, respective forces and moments acting on the body can be described, such as aerodynamic force $\mathbf{F}_B^a$ and thrust force $\mathbf{F}_B^t$. Some forces are not naturally expressed in the BFF. Such forces have to be transformed via $\mathbf{R}_{B/N}$ to a BFF. The total external force can be written by equation (3.5).

$$\mathbf{F}_B = \mathbf{F}_B^t + \mathbf{F}_B^a \tag{3.5}$$

The final propeller force of a generalized multirotor UAV is now expressed with R_{pr}^j being the radius of the propeller j. The final force can be derived resulting in equation (3.6), where ρ is the air density, ω_j is the propeller angular velocity, and $C_T^j(\mu_j)$ is the value of propeller thrust coefficient dependent on the value of advance ratio μ_j.

$$\mathbf{F}_B^t = \begin{bmatrix} 0 \\ 0 \\ -\pi \sum_{j=1}^{N} \rho \left[R_{pr}^j\right]^4 \omega_j^2 C_T^j(\mu_j) \end{bmatrix} \tag{3.6}$$

The advance ratio μ_j is a free-stream in-disk plane air velocity to the propeller velocity defined by equation (3.7), where the airspeed $\mathbf{u}_B^a = \mathbf{v}_B - \mathbf{v}_B^w$ is expressed as a difference of the inertial velocity and the local wind velocity expressed in the BFF, and α is the Angle of Attack (AoA) of the vehicle. The airspeed vector has components $\mathbf{u}_B^a = [u_x^a \; u_y^a \; u_z^a]^T$.

$$\mu_j = \frac{\sqrt{(u_x^a)^2 + (u_y^a)^2}\cos\alpha}{\omega_j R_{pr}^j} \tag{3.7}$$

Finally, the aerodynamic force can be decomposed to propeller forces generated by the interaction of the aerodynamic flow with the rotating propellers and the drag force, so that $\mathbf{F}_B^a = \mathbf{F}_B^{prop} + \mathbf{F}_B^{drag}$. The drag force $\mathbf{F}_B^{drag}$ can be simplified according to equation (3.8), where C_D is the drag coefficient and A_D is the reference drag area.

$$\mathbf{F}_B^{drag} = \frac{1}{2}\rho C_D A_D \mathbf{u}_B^a \parallel \mathbf{u}_B^a \parallel_2 \tag{3.8}$$

The aerodynamic forces $\mathbf{F}_B^{prop}$ will be investigated in more detail later in this chapter extending the model beyond the standard formulation using the results of Blade Element Momentum Theory (BEMT).

Following the standard formulation, the resulting moment $\mathbf{M}_B$ of all external moments acting on the UAV CoG can be decomposed as in equation (3.9).

$$\mathbf{M}_B = \mathbf{M}_B^t + \mathbf{M}_B^q + \mathbf{M}_B^j + \mathbf{M}_B^g + \mathbf{M}_B^a \tag{3.9}$$

The decomposed moments contain thrust reaction torque due to $\mathbf{F}_B^t$, the propeller thrust couple, the inertia couple due to the rotating parts of the motor, the gyroscopic torque, and the aerodynamic moment. Let $\mathbf{P}_{prop}^j = \begin{bmatrix} p_x^j & p_y^j & p_z^j \end{bmatrix}^T$ be a position vector of propeller j

with respect to UAV center of mass in the BFF. The thrust reaction torque of the propellers can be generalized to N-propeller UAV by using equation (3.10).

$$\mathbf{M}_B^t = \begin{bmatrix} -\pi\rho \sum_{j=1}^{N} \left[R_{pr}^j\right]^4 \omega_j^2 C_T^j(\mu_j) p_y^j \\ \pi\rho \sum_{j=1}^{N} \left[R_{pr}^j\right]^4 \omega_j^2 C_T^j(\mu_j) p_x^j \\ 0 \end{bmatrix} \tag{3.10}$$

Next, the propeller thrust couples are computed assuming the power coefficient $C_P^j(\mu_j)$ for the propeller j and its rotation orientation $O_j \in \{-1, 1\}$ as shown in equation (3.11).

$$\mathbf{M}_B^q = \begin{bmatrix} 0 \\ 0 \\ -\pi\rho \sum_{j=1}^{N} \left[R_{pr}^j\right]^5 \omega_j^2 C_P^j(\mu_j) O_j \end{bmatrix} \tag{3.11}$$

Using a known value of motor moment of inertia J_m^j of the motor j, the motor inertia couples can be written as in (3.12),

$$\mathbf{M}_B^c = \begin{bmatrix} 0 \\ 0 \\ -\sum_{j=1}^{N} J_m^j \dot{\omega}_j O_j \end{bmatrix} \tag{3.12}$$

where $\dot{\omega}_j$ is the propeller angular velocity speed. Finally, the gyroscopic torques of the propellers in BFF are computed by equation (3.13).

$$\mathbf{M}_B^g = \begin{bmatrix} -q \sum_{j=1}^{N} J_m^j \omega_j O_j \\ p \sum_{j=1}^{N} J_m^j \omega_j O_j \\ 0 \end{bmatrix} \tag{3.13}$$

where p and q are the UAV angular rates. Aerodynamic moments $\mathbf{M}_B^a$ are assumed to be caused by the interaction of the flow with the rotating propellers and are described further in this chapter.

The standard simplified model of a generalized multirotor UAV neglecting complex aerodynamic interactions was presented. The model is driven by assuming angular velocities of each propeller as a control input. This is a rather simplified assumption since the UAV's propellers are usually driven by BLDC motors regulating the angular velocity of each propeller. Extending the model with a set of ODEs describing the dynamics of the BLDC according to [112] leads to (3.14). For motor j, the change of current I_j and motor angular velocity ω_j are the variables, where

$$\begin{aligned} L_j \dot{I}_j &= V_j - k_e^j \omega_j - R_j I_j \\ J_m^j \dot{\omega}_j &= k_m^j I_j - b_m^j \omega_j - \pi\rho \left[R_{pr}^j\right]^5 \omega_j^2 C_P^j(\mu_j) \end{aligned} \tag{3.14}$$

L_j is the inductance, R_j the resistance, k_e^j the electromagnetic force constant, k_m^j the motor torque constant, and b_m^j the viscous friction constant. The applied voltage V_j results in the

desired angular velocity of the motor. Alternatively, V_j can be used as a primary control input. The applied voltage controls the resulting current by a Pulse-Width Modulation (PWM) typically used within modern micro-controllers. The BLDC motors are assumed to be connected to a Li-ion battery model [70] allowing for the State of Charge (SoC) to be measured. The battery cell capacity Q_{bat} determines the battery SoC as in equation (3.15).

$$SoC = 100\left(1 - \frac{\int_{t_0}^{t} I_{bat}\, dt}{Q_{bat}}\right) \quad [\%] \tag{3.15}$$

The voltage across the battery cell is determined from the battery current I_{bat}, battery resistance R_{bat}, and open circuit voltage e_m as shown in equation (3.16).

$$V_{bat} = e_m - R_{bat} I_{bat} \tag{3.16}$$

The open circuit voltage is computed using the measured constants such as the open circuit voltage at full load e_0, the bias voltage U_{bias}, exponential voltage U_{exp}, and exponential capacity C_{exp}. The open circuit voltage can be expressed as in equation (3.17).

$$e_m = e_0 - U_{bias}\left(\frac{Q_{bat}}{Q_{bat} - \int_{t_0}^{t} I_{bat}\, dt}\right) + U_{exp} e^{-C_{exp}\int_{t_0}^{t} I_{bat}\, dt} \tag{3.17}$$

The set of open circuit voltage parameters can be determined from the discharge curve. Battery capacity is usually measured either in Ah (Amp-hours), or Wh (Watt-hours). Compared to the other rechargeable battery technologies including nickel-cadmium and nickel-metal-hydride, Li-ion batteries have a number of advantages. They have one of the highest energy densities of any battery technology today ($100 - 265$ Wh/kg or $250 - 670$ Wh/L) and can deliver a higher voltage. The selection of battery highly depends on the mission objectives and the estimated flight time.

The generalized multirotor UAV model was implemented in a dedicated software tool, allowing for customized UAV dynamic model development. Figure 3.2 shows the Graphical User Interface (GUI) of the modeling tool for UAV configuration development and environment specification together with a graphical representation of the modeled UAV and environment. The software was implemented in Python and it is composed of the Multirotor modeling tool, the environment modeling tool, the trajectory optimization tool, and a simulation tool. The interfaces of the multirotor modeling tool and the environmental modeling tool serve as user input acquisition tools with a corresponding 3D visualization. The user can specify all components defining the dynamical model starting with geometry and physical properties of the air-frame to aerodynamic properties of the selected propeller. The environment modeling tool allows for terrain and wind characteristics specification. The specified data are used either within the trajectory optimization tool or a simulation tool, which implement algorithms for the optimal trajectory development and simulation of the closed-loop UAV dynamics that emulate the dedicated mission profiles.

A simplified standard model of UAV is the basis of many navigation and control systems. However, neglecting the aerodynamic interactions can lead to a severe decrease in perfor-

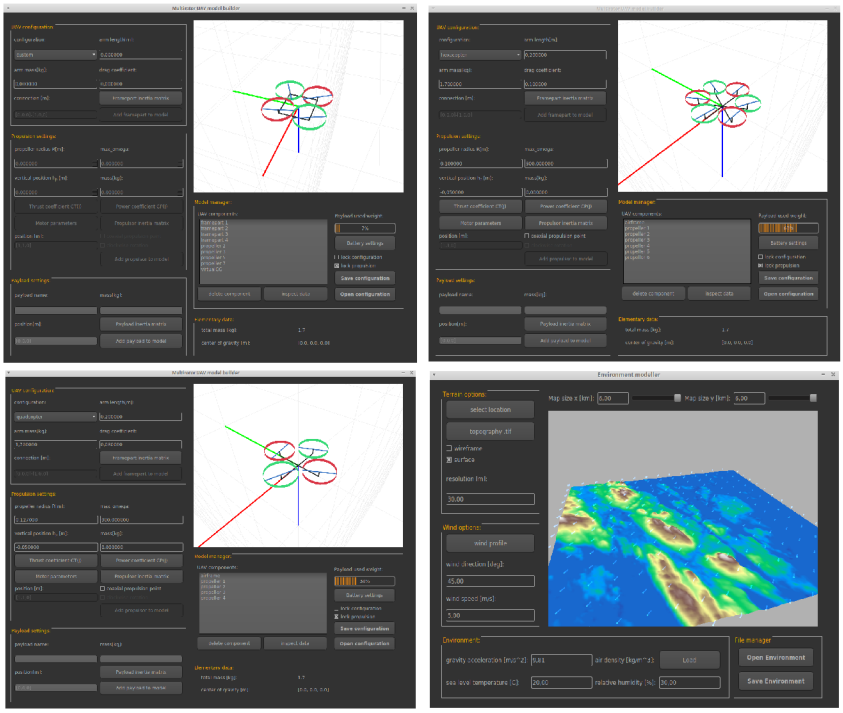

Figure 3.2: Multirotor UAV modeling tool and a terrain model.

mance or even failure of the system. The model will now be revisited to account for such interactions based on the BEMT modeling [54]. In order to correctly express the quantities with respect to rotor blades and the rotor itself, the rotor axes frame $\mathcal{F}_R = (x_R, y_R, z_R, O_R)$ and blade axes frame $\mathcal{F}_H = (x_H, y_H, z_H, O_H)$ are defined. Positive rotation is counter-clockwise assuming the top view. Initially, the air velocity with respect to a section of a blade can be

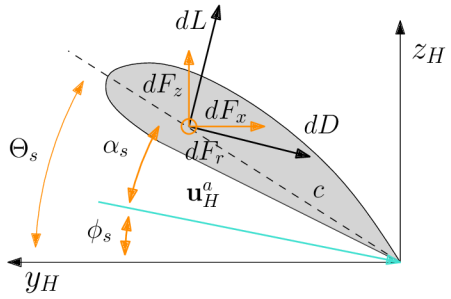

Figure 3.3: Section cut of propeller's blade.

expressed by a vector (3.18), where $U = \sqrt{(u_x^a)^2 + (u_y^a)^2}$ is the in-plane velocity magnitude, r is a radial coordinate of the section blade, ω is the angular velocity of the propeller, v_i is the induced velocity, and ψ_H is the yaw angle of the blade with respect to airflow direction.

$$\mathbf{u}_H^a = [U\cos\psi_H \quad -(U\sin\psi_H + \omega r) \quad -(u_z^a + v_i)]^T \tag{3.18}$$

For illustration, Figure 3.3 shows a section cut of the blade and the relevant quantities in the blade's reference frame. In contrast, Figure 3.5 shows a top and side view of the rotor disk with the associated quantities and airflow. The induced velocity is the result of rotor wake inflow. An example of the computed induced velocity maps at fixed vertical airflow magnitudes is shown in Figure 3.4. There are essentially two modes to be considered. In a hover condition, the induced velocity can be expressed as a function of the total thrust as

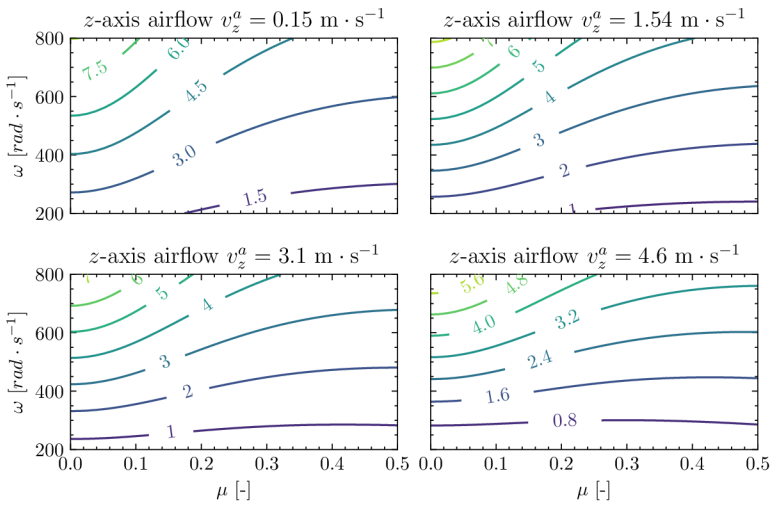

Figure 3.4: Induced velocity grid map at fixed vertical airflow magnitudes.

in equation (3.19). However, in forward flight, climb or descent, the expression becomes a transcendental equation which has to be solved numerically and saved as a lookup table. It is given by v_i^f.

$$v_i^h(T) = \sqrt{\frac{T}{2\rho\pi R_{pr}^2 k}} \qquad v_i^f = \frac{C_T}{2k}\frac{\omega R_{pr}}{\sqrt{\mu^2+\lambda^2}} \tag{3.19}$$

Defining the inflow ratio by equation (3.20), equation (3.18) can be rewritten with the substi-

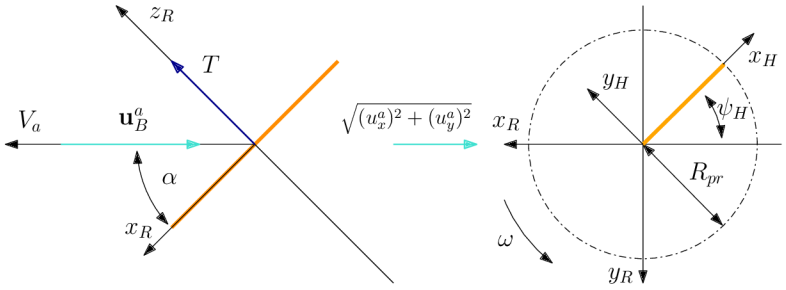

Figure 3.5: Rotor disk from side and top view.

tuted inflow and advance ratio terms as shown in equation (3.21) assuming a blade section.

$$\lambda = \frac{u_z^a + v_i}{\omega R_{pr}} \tag{3.20}$$

The flow velocity with respect to a blade section now allows one to compute lift and drag at that section, which can then be integrated along the entire blade.

$$\mathbf{u}_H^a = \begin{bmatrix} \omega R_{pr}(\mu\cos\psi_H) & -\omega R_{pr}\left(\frac{r}{R_{pr}} + \mu\sin\psi_H\right) & -\omega R_{pr}\lambda \end{bmatrix}^T \tag{3.21}$$

Assuming the section's AoA to be $\alpha_s = \Theta_s - \phi_s$ such that $\phi_s = \tan^{-1}\left(\lambda^{-1}\left(\frac{r}{R_{pr}} + \mu\sin\psi_H\right)\right)$, Θ_s is the blade section pitch angle in the blade coordinates. Commonly, the section pitch is approximated linearly using the known pitch at blade root Θ_s^0 and twist angle slope $\Delta\Theta_s$ such that the pitch, at any point of the blade, is computed by equation (3.22).

$$\Theta_s = \Theta_s^0 + \frac{r}{R_{pr}}\Delta\Theta_s \tag{3.22}$$

With assumption of small inflow angle ϕ_s, the lift and drag differential at the blade section can be expressed using the approximation (3.23), where $\bar{C}_D$ is the mean drag coefficient of the rotor disk, $\bar{c}$ is the mean chord of the blade section, and a is the slope of the lift curve.

$$\begin{aligned} dL &\approx \frac{1}{2}\rho\left[\omega R_{pr}\left(\frac{r}{R_{pr}} + \mu\sin\psi_H\right)\right]^2 a\left(\Theta_s - \phi_s\right)\bar{c}\, dr \\ dD &\approx \frac{1}{2}\rho\left[\omega R_{pr}\left(\frac{r}{R_{pr}} + \mu\sin\psi_H\right)\right]^2 \bar{C}_D\bar{c}\, dr \end{aligned} \tag{3.23}$$

The blade's section forces can be expressed by using the small angle approximation in (3.24).

$$\begin{aligned} dF_z &= dL\cos\phi_s - dD\sin\phi_s \approx dL \\ dF_x &= dL\sin\phi_s + dD\cos\phi_s \approx dL\phi_s + dD \\ dF_r &= dD_r \approx 0 \end{aligned} \tag{3.24}$$

Calculating the steady force and moment coefficients involves integration along the blade span, and integration of the average across the whole rotor disk, which assumes uniformity of the flow with regards to the rotor, while neglecting the root cutout and tip loss. The resulting force and moment coefficient expressions are given in equations (3.25).

$$\begin{aligned} C_X &\approx -\frac{N_b}{2\pi R_{pr}}\int_0^{2\pi}\int_0^1\left\{\frac{dF_x}{R_{pr}}\sin\psi_H\right\}d\psi_H \\ C_Y &\approx \frac{N_b}{2\pi R_{pr}}\int_0^{2\pi}\int_0^1\left\{\frac{dF_x}{R_{pr}}\cos\psi_H\right\}d\psi_H \\ C_Z &\approx \frac{N_b}{2\pi R_{pr}}\int_0^{2\pi}\int_0^1\left\{\frac{dF_z}{R_{pr}}\right\}d\psi_H \\ C_{M_x} &\approx -\frac{N_b}{2\pi R_{pr}}\int_0^{2\pi}\int_0^1\left\{\frac{dF_z r}{R_{pr}^2}\right\}\sin\psi_H d\psi_H \\ C_{M_y} &\approx \frac{N_b}{2\pi R_{pr}}\int_0^{2\pi}\int_0^1\left\{\frac{dF_z r}{R_{pr}^2}\right\}\cos\psi_H d\psi_H \\ C_{M_z} &\approx -\frac{N_b}{2\pi R_{pr}}\int_0^{2\pi}\int_0^1\left\{\frac{dF_x r}{R_{pr}^2}\right\}d\psi_H \end{aligned} \tag{3.25}$$

Evaluating the integrals in (3.25) and substituting rotor solidity defined as $\sigma = N_b c/\pi R_{pr}$ assuming N_b blades, the resulting coefficients are given by equations (3.26).

$$\begin{aligned} C_X(\mu,\lambda) &\approx -\frac{\sigma a}{2}\left[\left(\frac{\Theta_s^{0.75}}{3} + \frac{\Theta_s^0}{6}\right)\lambda + \frac{\bar{C}_D}{2a}\right]\mu \\ C_Y(\mu,\lambda) &\approx 0 \\ C_Z(\mu,\lambda) &\approx \frac{\sigma a}{2}\left[\left(\frac{\Theta_s^{0.75}}{3} - \frac{1}{2}\lambda\right) + \left(\frac{\Theta_s^{0.75}}{3} + \frac{\Theta_s^0}{6}\right)\mu^2\right] \\ C_{M_x}(\mu,\lambda) &\approx -\frac{\sigma a}{2}\left(\frac{\Theta_s^{0.75}}{3} - \frac{1}{4}\lambda\right)\mu \\ C_{M_y}(\mu,\lambda) &\approx 0 \\ C_{M_z}(\mu,\lambda) &\approx -\frac{\sigma a}{2}\left[\left(\frac{\Theta_s^{0.75}}{3} - \frac{1}{2}\lambda\right)\lambda + (1+\mu^2)\frac{\bar{C}_D}{4a}\right] \end{aligned} \tag{3.26}$$

The value $\Theta_s^{0.75}$ refers to a pitch angle at 75 % radius of the blade. The derived force and moment coefficients assume small inflow angles and no wake interactions among neighboring rotors. Despite some simplifications, the model becomes much more accurate in the case of non-nominal conditions and in the presence of wind turbulence and vertical gusts making scenario testing and failure probability estimates introduced in the next chapter more accurate and realistic.

Revisiting the simplified forces and moments derived earlier and augmenting them by the derived coefficients results in a concise form shown in equations (3.27) and (3.28).

$$\mathbf{F}_B = \rho R_{pr}^4 \sum_{j=1}^{N} \omega_j^2 \begin{bmatrix} C_X^j(\mu_j, \lambda_j) \cos\psi_R^j \\ C_X^j(\mu_j, \lambda_j) \sin\psi_R^j \\ -C_T^j(\mu_j, \lambda_j) \end{bmatrix} + \mathbf{F}_B^{drag} \tag{3.27}$$

Note that a transformation of the coefficient by an azimuth angle ψ_R^j of the j-th rotor has to be performed, since the rotor disk's frame x_R-axis always points towards the airflow as was shown in Figure 3.5.

$$\mathbf{M}_B = \rho R_{pr}^5 \sum_{j=1}^{N} \omega_j^2 \begin{bmatrix} O_j C_{M_x}^j(\mu_j, \lambda_j) \cos\psi_R^j - \frac{p_y^j}{R_{pr}} C_T^j(\mu_j, \lambda_j) \\ O_j C_{M_x}^j(\mu_j, \lambda_j) \sin\psi_R^j + \frac{p_x^j}{R_{pr}} C_T^j(\mu_j, \lambda_j) \\ O_j C_{M_z}^j(\mu_j, \lambda_j) + \frac{C_X^j}{R_{pr}} (p_x^j \sin\psi_R^j - p_y^j \cos\psi_R^j) \end{bmatrix} + \mathbf{M}_B^g + \mathbf{M}_B^c \tag{3.28}$$

The azimuth angle of each rotor frame can be computed simply as $\psi_R^j = \tan^{-1}\left(u_y^a / u_x^a\right)$, thus depending on the wind velocity magnitude.

3.2 MULTIROTOR UNMANNED AERIAL VEHICLE QUASI-PHYSICAL MODELING

This work utilizes SimscapeMultibody™ library of MATLAB® and Simulink® to construct a quasi-physical representation of a quadcopter UAV. A 3D visualization of the component composition of a selected quadcopter is shown in Figure 3.6 from three different viewpoints.

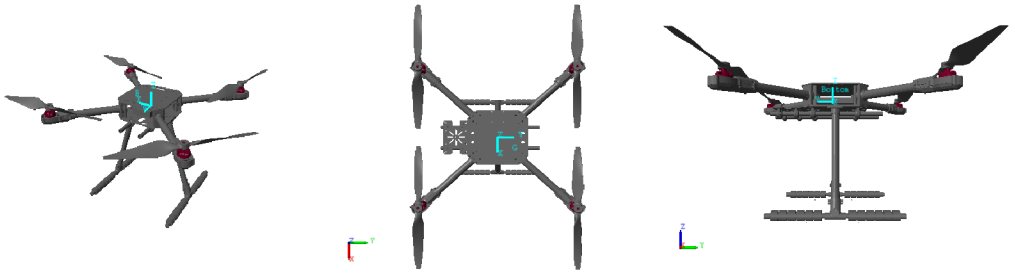

Figure 3.6: Quadcopter model visualization shown in mechanical explorer.

The composed model in SimscapeMultibody™ library is shown in Figure 3.7 incorporating BLDC motor models and a battery model alongside the airframe and propeller models.

The BLDC motors integrate effects such as electrical losses, rotor inertia, damping or temperature dependence on efficiency. The model also integrates measurements of kinematic state in the BFF subject to measurement noise from sensors such as IMU or magnetometer. The ground is modeled as a flat surface connected to the UAV via contact forces on quadcopter landing legs. The environment block is present to incorporate disturbances according to the ODD of the vehicle.

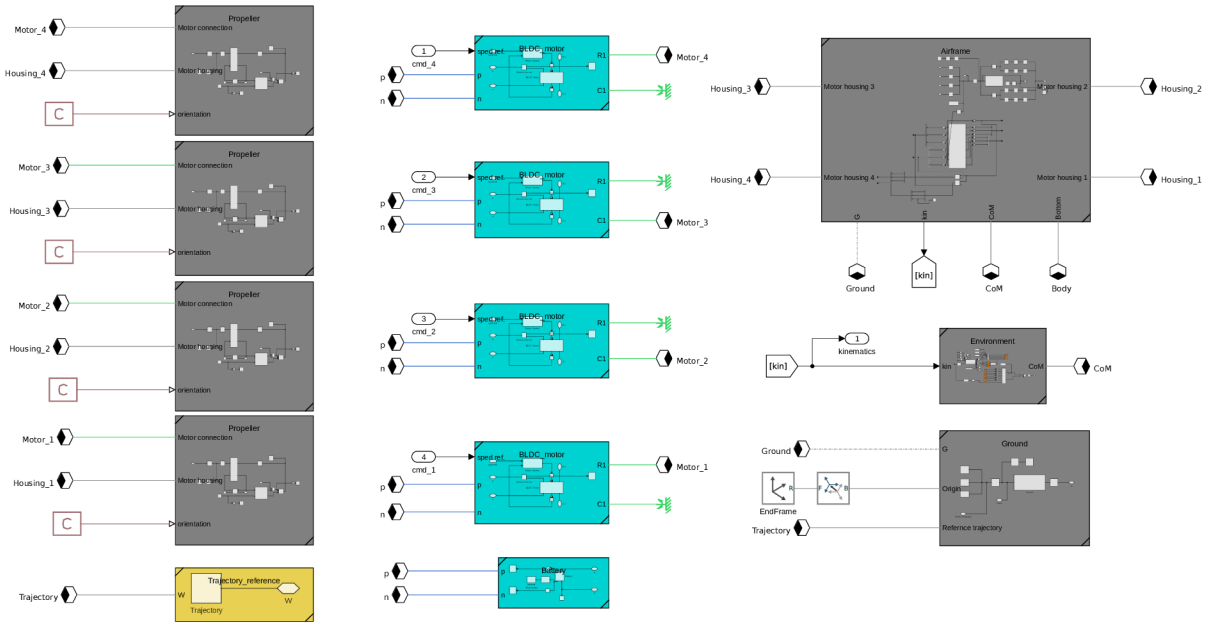

Figure 3.7: Quasi-physical model components of a quadcopter drone.

3.3 SERIAL ROBOTIC MANIPULATOR MATHEMATICAL MODELING

Serial manipulators are the most widely used industrial robots consisting of a series of links connected by motorized joints. The structure of the manipulator extends from its base to end-effector typically equipped with a gripping mechanism designed to meet the operational domain criteria. Finger-like grippers, vacuum grippers, jamming, or magnetic grippers are some of the possible extensions of the end-effector. Assuming a manipulator with N_r actuated joints, the generalized dynamics derived from Euler-Lagrange equations can be expressed by equation (3.29), where $\mathbf{I} \in \mathbb{R}^{N_r \times N_r}$ is the generalized inertia matrix, $\mathbf{C}_o \in \mathbb{R}^{N_r \times N_r}$ is the Coriolis matrix, and $\mathbf{g} \in \mathbb{R}^{N_r}$ is the gravity vector. $\boldsymbol{\Theta} \in \mathbb{R}^{N_r}$, $\dot{\boldsymbol{\Theta}} \in \mathbb{R}^{N_r}$ and $\ddot{\boldsymbol{\Theta}} \in \mathbb{R}^{N_r}$ are the joint positions, angular velocity, and angular accelerations respectively. The applied torque $\boldsymbol{\tau} \in \mathbb{R}^{N_r}$ is the control vector of this dynamical system.

$$\mathbf{I}(\boldsymbol{\Theta})\ddot{\boldsymbol{\Theta}} + \mathbf{C}_o(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})\dot{\boldsymbol{\Theta}} + \mathbf{g}(\boldsymbol{\Theta}) = \boldsymbol{\tau} \tag{3.29}$$

The generalized inertia matrix can be expressed using equation (3.30), where $m_i \in \mathbb{R}$ is the mass of link i and $\mathbf{I}_i \in \mathbb{R}^{3\times3}$ is the inertia tensor of the link with respect to a frame attached to the CoG of the manipulator part. $\mathbf{J}_{R_i} \in \mathbb{R}^{3\times N_r}$ and $\mathbf{J}_{T_i} \in \mathbb{R}^{3\times N_r}$ are the rotational and translation Jacobian matrices.

$$\mathbf{I} = \sum_{i=1}^{N_r} \left(m_i \left[\mathbf{J}_{T_i} \right]^T \mathbf{J}_{T_i} + \mathbf{J}_{R_i}^T \mathbf{I}_i \mathbf{J}_{R_i} \right) \tag{3.30}$$

The Coriolis term can be derived as in equation (3.31). The operation $\otimes$ denotes Kronecker product operator.

$$\mathbf{C}_o = \frac{1}{2} \left[\frac{\partial \mathbf{I}}{\partial \boldsymbol{\Theta}} \left(\mathbf{E} \otimes \dot{\boldsymbol{\Theta}} \right) + \frac{\partial \mathbf{I}}{\partial \boldsymbol{\Theta}} \left(\dot{\boldsymbol{\Theta}} \otimes \mathbf{E} \right) - \left(\frac{\partial \mathbf{I}}{\partial \boldsymbol{\Theta}} \left(\dot{\boldsymbol{\Theta}} \otimes \mathbf{E} \right) \right)^T \right] \tag{3.31}$$

A control mechanism for a serial manipulator is typically based on torque control or a position control based on a type of actuators and sensors integrated to the robot. In the case of torque control and path following, the dynamical system is augmented by using a path $\mathbf{s}(t)$. In the case of position control, the Euler-Lagrange formulation is not used within the MPC formulation, but rather a simple double-integrator dynamics for the joint state can be applied. In this thesis, a Denavit-Hartenber (DH) convention for robot arm kinematics is adopted. The end-effector pose $\mathbf{Z}_{end} \in \mathbb{R}^{4\times4}$ with respect to the manipulator base can thus be derived by joint and link successive transformations as formalized in equation (3.32), where $\mathbf{Z}_i \in \mathbb{R}^{4\times4}$ is the joint i transformation and $\mathcal{Z}_i \in \mathbb{R}^{4\times4}$ is the link i transformation.

$$\mathbf{Z}_{end} = \mathbf{Z}_1 \mathcal{Z}_1 \mathbf{Z}_2 \mathcal{Z}_2 \dots \quad \mathcal{Z}_{N_r-1} \mathbf{Z}_{N_r} \mathcal{Z}_{N_r} \tag{3.32}$$

The joint movement can be described by the transformation (3.33), where $\theta_i \in \mathbb{R}$ is the rotation around the common frame z-axis, and $d_i \in \mathbb{R}$ is the offset in the z-axis direction. In a typical configuration, the vector $\boldsymbol{\Theta} = [\theta_1, \theta_2 \dots \theta_N]^T$ is the state vector of the manipulator dynamics.

$$\mathbf{Z}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.33}$$

The link transformation defined by a screw displacement around a common normal between each two joints is defined by equation (3.34), where $l_i \in \mathbb{R}$ is the link length and $\alpha_l^i \in \mathbb{R}$ is the angle around common normal defining the new z-axis direction.

$$\mathcal{Z}_i = \begin{bmatrix} 1 & 0 & 0 & l_i \\ 0 & \cos\alpha_l^i & -\sin\alpha_l^i & 0 \\ 0 & \sin\alpha_l^i & \cos\alpha_l^i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.34}$$

While DH parameter convention has its limitations with regards to arbitrary robotic systems, it is still widely used as a structured approach for the description of serial robot kinematics.

3.4 SERIAL ROBOTIC MANIPULATOR QUASI-PHYSICAL MODELING

For the quasi-physical model of a robotic manipulator, the work utilizes SimscapeMultibody™ library of MATLAB® and Simulink®. A 3D visualization of the component composition of a selected six-joint robotic manipulator is shown in Figure 3.8 from three different viewpoints. The robotic manipulator is modeled by links and joints

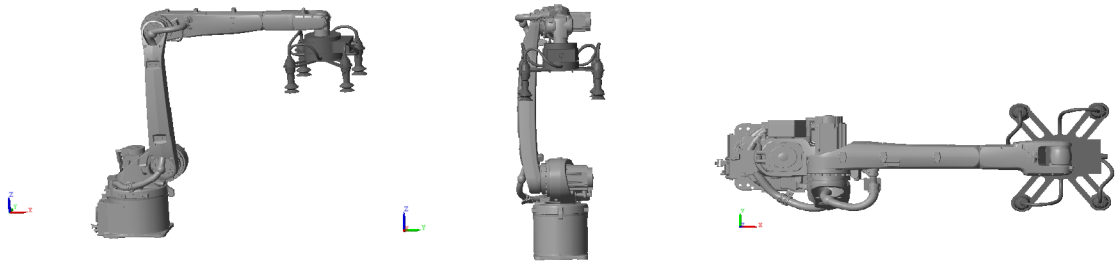

Figure 3.8: Robotic arm model visualization shown in mechanical explorer.

connected in series. The joint angles have physical limits constraining the motion and operational envelope of the robotic arm. A single joint connection is shown in Figure 3.9 with the joint movement subject to friction torque composed of Stribeck, Coulomb, and viscous friction. The input to each joint is the applied torque from the dedicated servo motor.

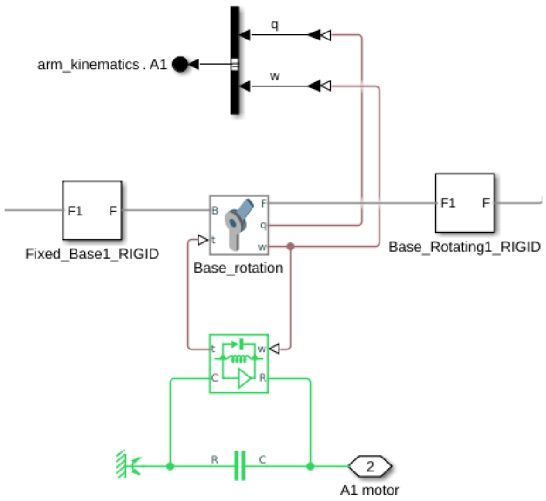

Figure 3.9: Implementation of a single manipulator joint with friction.

Each joint outputs the joint angular position Θ and joint angular velocity $\dot{\Theta}$. Each revolute joint connects two rigid parts of the manipulator. The vacuum gripper is modeled by a combination of contact forces and internal forces modeling the holding force of each suction cup. The suction cups are activated when the contact force of the target and the gripper exceeds a given threshold value. The end-effector equipped with a series of suction cups requires a high precision with regards to the desired orientation so that all four suction cups make contact with the considered payload. In comparison to finger-like grippers, suction cups often provide a softer manipulation with the target, but they are restricted to objects having a flat top surface in case the suction cups cannot be tilted.

3.5 DISTURBANCE MODELING

The developed model of multirotor UAV is affected by the environment during its operation in a realistic setting. The goal of this chapter is to address the relevant external influences occurring during UAV operations. The wind field modeling, turbulence, and gust modeling are the primary external factors that have a significant effect on the stability and control of UAVs during flight. Terrain is a primary constraint to UAV mission/path planning and is either natural (mountains) or man-made (buildings). Especially in an urban setting, the flow field of the wind is composed of many turbulent structures originating from shedding around horizontal and vertical edges, which can significantly affect the stability and performance of small UAVs. Such flow fields can be approximated by performing Computational Fluid Dynamics (CFD) simulations in a known scenario.

3.5.1 *Wind field and standard turbulence models*

Wind affects the stability and maneuvering capabilities of UAVs, which can be particularly dangerous when operating near obstacles or other aerial vehicles. Let us initially assume a stationary wind field $\mathbf{v}_N^w(x, y, z) = [w_x \quad w_y \quad w_z]^T$, which is defined as an arbitrary function of position. Locally, the wind field has a nominal wind speed W and direction α_W. The components of a stationary wind field can be approximated as $w_x = W \cos \alpha_W$ and $w_y = W \sin \alpha_W$. The change of altitude usually affects the nominal wind speed. The so-called wind profile power law was used for the vertical extrapolation of the nominal wind speed W in the atmospheric surface layer [31]. In order to estimate the wind speed at a certain height h, the power law (3.35) is used to extrapolate $W(h)$ using known wind speed $W(h_A)$ at reference height h_A.

$$W(h) = W(h_A) \left(\frac{h}{h_A} \right)^n \tag{3.35}$$

The exponent n is an empirically derived coefficient that varies with the stability of the atmosphere. In neutral stability conditions, the value $n = 0.143$ is used for open land surfaces, and value $n = 0.11$ is used for flying over open water surfaces. The applications of wind profile power law include wind power assessments or atmospheric pollution dispersion models. A realistic wind field is not static and should, therefore, include turbulence and gusting. A turbulence is defined as an abrupt or irregular movement of air that creates sharp and quick updrafts and downdrafts [21]. Gusting is defined as a sudden, brief increase in wind speed. Gusts usually have a much higher amplitude than turbulence and occur in an unpredictable manner. In the context of vehicle control design, the robustness of the controller should guarantee performance in the presence of such events. There are several stochastic turbulence models in the literature for both over-land and over-water environments [79]. Simulations considering open land surfaces have been studied well, and models such as Dryden and von Kármán models were developed and are part of a military specification MIL-F-8785C and MIL-HDBK-1797. Both models were developed for aircraft operating in higher altitudes than multirotor UAVs. However, the low-altitude von Kármán model showed a good agreement with the measured data for small aerial vehicles (< 610 m) [121]. The model is specified by

Power Spectral Density (PSD) of velocity and angular velocity disturbances as in equation (3.36)

$$\begin{aligned}
\Phi_u(\omega) &= \frac{2\sigma_u^2 L_u}{\pi W(h)} \frac{1}{\left(1+(1.339 L_u \Gamma_{PSD})^2\right)^{5/6}} & \Phi_p(\omega) &= \frac{\sigma_w^2}{L_w W(h)} \frac{0.8\left(\frac{\pi L_w}{4l}\right)^{1/3}}{1+(8L\Gamma_{PSD}\pi^{-1})^2} \\
\Phi_v(\omega) &= \frac{\sigma_v^2 L_v}{\pi W(h)} \frac{1+8/3\,(1.339 L_v \Gamma_{PSD})^2}{\left(1+(1.339 L_v \Gamma_{PSD})^2\right)^{11/6}} & \Phi_q(\omega) &= \frac{\pm\Gamma_{PSD}^2}{1+(8L\Gamma_{PSD}\pi^{-1})^2}\Phi_w(\omega) \\
\Phi_w(\omega) &= \frac{\sigma_w^2 L_w}{\pi W(h)} \frac{1+8/3\,(1.339 L_w \Gamma_{PSD})^2}{\left(1+(1.339 L_w \Gamma_{PSD})^2\right)^{11/6}} & \Phi_r(\omega) &= \frac{\pm\Gamma_{PSD}^2}{1+(6L\Gamma_{PSD}\pi-1)^2}\Phi_v(\omega)
\end{aligned} \tag{3.36}$$

The coefficient Γ_{PSD} is defined according to equation (3.37), where ω is the frequency.

$$\Gamma_{PSD} = \frac{\omega}{W(h)} \tag{3.37}$$

The von Kármán model scaling lengths are $L_u = \frac{h}{(0.177+0.000823h)^{1/2}}$, $L_v = L_u$ and $L_w = h$, and the intensities are $\sigma_u = \frac{\sigma_w}{(0.177+0.000823h)^{0.4}}$, $\sigma_v = \sigma_u$ and $\sigma_w = 0.1W(h)$. The constant L is assumed to be the UAV arm length, or half the aircraft wingspan. Figure 3.10 shows a sample of a generated turbulence using the von Kármán model.

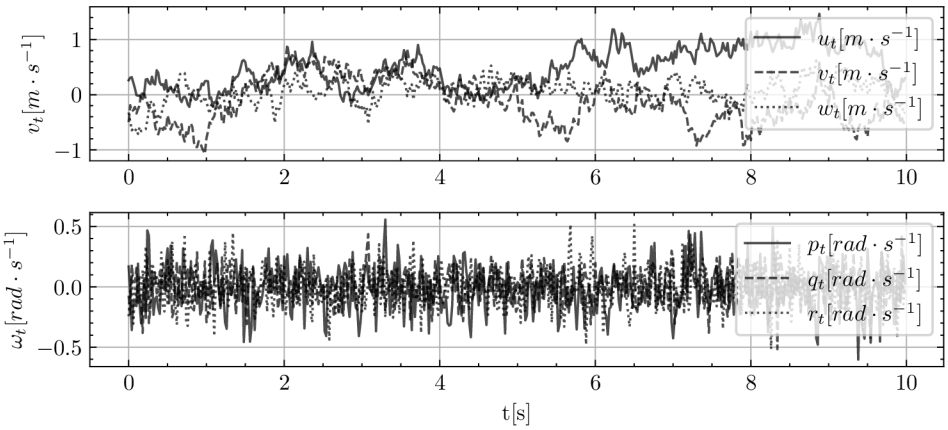

Figure 3.10: Generated von Kármán translation and rotation velocities.

3.5.2 *Ground effect*

Many UAV applications require the vehicle to fly near the ground, walls, or other objects, thus changing the characteristics of the airflow. The wake of the multirotor UAV is often affected such that it rapidly expands parallel to the ground alternating the slip stream and the induced velocity of the vehicle [95]. The ground effect has been studied extensively for helicopters, and most models for the ground effect are still based on those works. Some unique characteristics of this problem for multirotor UAVs, such as the possibility of a partial ground effect or a fountain effect, are also important to consider. One of the most mentioned models of the ground effect is the Potential Flow of Images (PFI) method, which yields a simple analytical model. It has been experimentally shown to provide good estimates of

the relationship of rotor thrust and height above the ground. It models a three-dimensional velocity potential source of the rotor as expressed in equation (3.38), where v_i is the induced velocity of the rotor and (x_0, y_0, z_0) is the location of the potential source.

$$\Pi = \frac{-R_{pr}^2 v_i}{4\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}} \tag{3.38}$$

The results for varying the z coordinate lead to derivation of a ratio of thrust change due to the ground effect as written in (3.39) for $z > R_{pr}/4$ with thrust modified by the ground effect denoted by $\tilde{T}$.

$$\tilde{T} = \frac{T}{1 - (R_{pr/4z})^2} \tag{3.39}$$

In contrast to the model (3.39) derived for a single rotor, [95] provides a model accounting for all the propellers of a quadcopter UAV. The model is given by equation (3.40), where d is the separation distance of two adjacent rotor axes.

$$\tilde{T} = \frac{T}{1 - (R_{pr/4z})^2 - R_{pr}^2(z/\sqrt{(d^2+4z^2)^3}) - (R_{pr}^2/2)(z/\sqrt{(2d^2+4z^2)^3})} \tag{3.40}$$

Due to the interference of other rotors, the ground effect has much more significance for multirotor vehicles. Finally, the experiments in [95] compared with the theoretical models showed inconsistency such that the ground effect encountered was even stronger in real experiments. This has been attributed to the fountain effect, which is caused by the flow interactions between the rotors, reversing the flow in the central region of the UAV back, and thus increasing the upward force. The modified model accounting for the fountain effect has an added term parametrized by the empirical body lift coefficient C_{Lb} and distance of two opposite rotor axes $2L$. The final model is then given by equation (3.41), where $K_1 = (R_{pr/4z})^2$, $K_2 = R_{pr}^2(z/\sqrt{(d^2+4z^2)^3})$, $K_3 = (R_{pr}^2/2)(z/\sqrt{(2d^2+4z^2)^3})$ and $K_4 = 2R_{pr}^2(z/\sqrt{(4L^2+4z^2)^3})C_{Lb}$.

$$\tilde{T} = \frac{T}{1 - K_1 - K_2 - K_3 - K_4} \tag{3.41}$$

In comparison, modeling a partial ground effect such that some propellers are close to a surface can be modeled by altering the thrust of individual rotors by PFI models (3.39) or (3.40). Alternatively, the ground effect can be studied by performing CFD simulations such as the one shown in Figure 3.11.

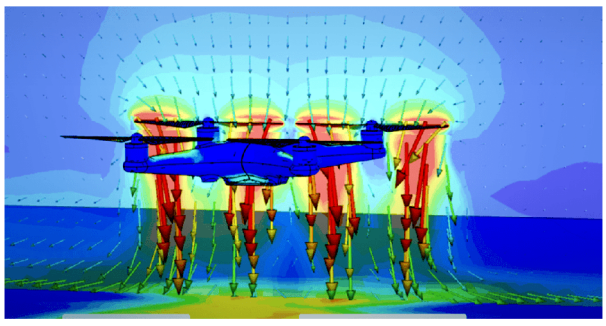

Figure 3.11: Example CFD simulation of ground effect. Source: [53]

3.5.3 *Wind gust and vortex modeling in urban environment*

Modeling wind fields in urban environments poses many challenges due to the presence of structures generating turbulent flows in the wind field. In many cases, flow field wake interactions of adjacent buildings lead to even more complex flow structures. Instead of modeling the whole dynamics of the vortices emerging by shedding from horizontal or vertical edges of urban structures, static horizontal and vertical vortices will be modeled for practical purposes and testing of the control system robustness. The assumed stationary vortex will have a defined intensity Γ, which can be used to compute an increase in tangential velocity as defined in equation (3.42) with r_v being a distance from the vortex axis.

$$W_v = \frac{\Gamma}{2\pi r_v} \tag{3.42}$$

In such a modeling scenario, the proximity to the vortex axis has to be avoided since equation (3.42) is singular at the vortex axis. Stationary vortex modeling yields a good approximation since the system response of a UAV has much faster dynamics than the vortex evolution in space. Figure 3.12 depicts a range of horizontal or vertical wind variations acting on the UAV passing through the vortex with minimum flyby distance d and vortex intensity Γ.

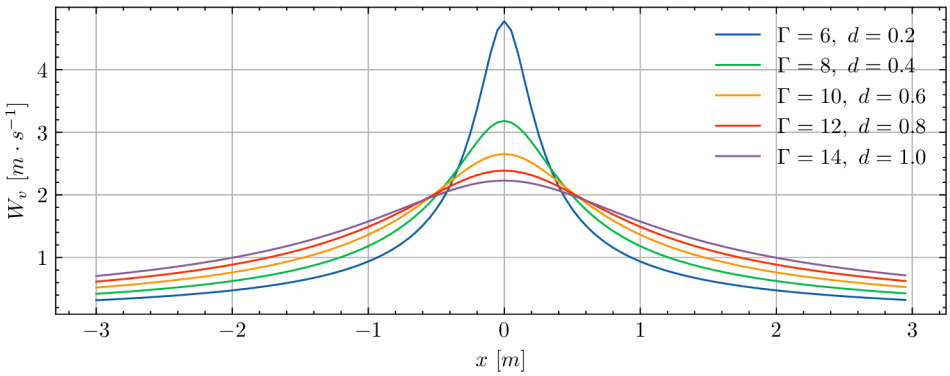

Figure 3.12: Vortex magnitudes based on intensity and fly-by distance.

The direction of wind changes accordingly from upward to downward, or changes direction in the horizontal plane as shown in Figure 3.13. The modeling choice for the vortex allows for the testing of different shapes of vortex profiles along the flown trajectory and study the control system response.

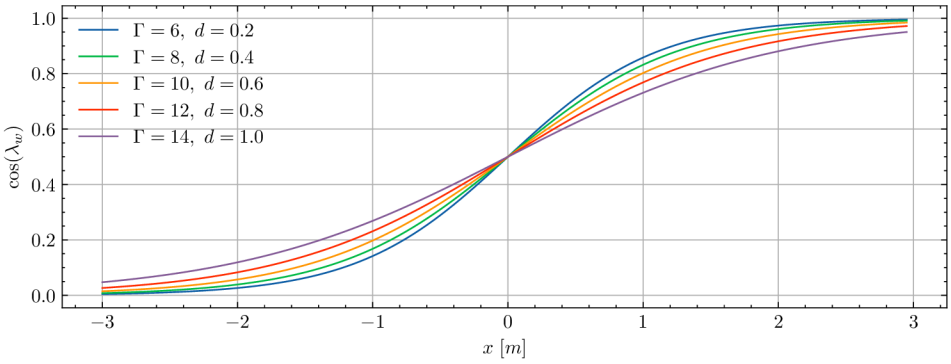

Figure 3.13: Vortex direction based on intensity and fly-by distance.

In comparison, wind gusts are brief increases in wind speed encountered in a short time duration affecting the translation components of the wind field. Following [21], an empirical model of wind gusts based on wind farm statistical data was obtained. The generalized shape of the wind gust can be described by equation (3.43). There are more relevant models of wind gusts such as the standard "1-cosine" shape specified by MIL-F-8785C. However, this model provides more flexibility with regards to the shape and duration of the gust. Moreover, the presented gust model assumes to be a discrete event with a temporary increase of wind speed which returns to its original value.

$$W_g(t) = \begin{cases} g_1(1-(g_{2r}t-g_3)^2)e^{-(g_{2r}t-g_3)^2/g_4}, & 0 < t < \frac{1}{2}t_{g1} \\ g_1, & \frac{1}{2}t_{g1} < t < \frac{1}{2}t_{g1}+t_h \\ g_1(1-(g_{2f}t^*-g_3)^2)e^{-(g_{2f}t^*-g_3)^2/g_5}, & t > \frac{1}{2}t_{g1}+t_h \end{cases} \tag{3.43}$$

The amplitude g_1 of the wind gust can be computed as in equation (3.44) and is based on nominal wind speed W_{nom}.

$$g_1 = \frac{52}{43}W_{nom} - \frac{1}{8} \tag{3.44}$$

In equation (3.43), rising and falling times are modeled as $g_{2r} = 2g_3/t_{g1}$ and $g_{2f} = 2g_3/t_{g2}$. The constant $g_3 = 6$ represents a dissipation time, $g_4 > 0$ models the dip of magnitude before the gust and $g_5 > 0$ models the dip of magnitude after the gust. The time t_h represents the duration of the peak of gust magnitude, and the time t^* can be computed as $t^* = t + \frac{1}{2}(t_{g2} - t_{g1}) - t_h$. The shape also depends on the time from gust start to peak t_{g1} and peak to the gust end t_{g2}. The total duration of the gust is thus $t_g = \frac{1}{2}t_{g1} + \frac{1}{2}t_{g2} + t_h$. Although there are significant variations in gust duration, a linear dependency given in [21] suggests that $t_g = 0.71(W_g - W_{nom}) + 3.51$. Figure 3.14 depicts a range of possible wind gusts with a varying peak magnitude and dip magnitudes assuming $t_g = 10\,s$.

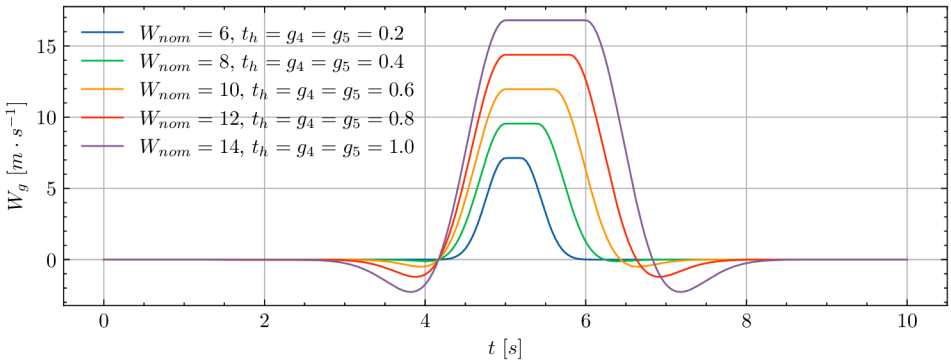

Figure 3.14: Varying wind gust profiles with symmetric shape.

Modeling disturbances such as wind gust or wind turbulence is important with regards to UAV control system validation procedure, as it emulates realistic conditions that have to be compensated for during operation. The disturbance models are typically based on specifications or experimental data.

4 SYSTEM ARCHITECTURE FOR THE COUPLED PROBLEM

The system architecture of the GNC algorithm for the coupled problem including multi-rotor UAV and a robotic manipulator has been developed with the intention of studying autonomous grasping of objects using this configuration. Aerial manipulation formed by a multirotor UAV and multi DoF robotic arm has been recently proposed in the context of multiple application domains such as autonomous transportation, or infrastructure repair. The grasping mechanism for an UAV has also been investigated in the context of bio-inspired perching [91] or soft gripping by the utilization of the jamming gripper technology [60]. Simple control techniques employed by many UAVs, such as PID control, show insufficient performance tasks involving coupled dynamics, such as the envisioned aerial manipulation, due to the system's nonlinear dynamics, and increased presence of uncertainties [41]. An adaptive two-layer control mechanism was proposed in [18], forming a cascaded control loop. However, introducing constraints to ensure that the payload stays within camera Field of View (FoV) or considering operational limits of the vehicle has not been fully addressed.

This thesis investigates an autonomous grasping maneuver driven by MPC scheme utilizing a coupled mathematical model of the system. The schematics of the grasping maneuver considered is shown in Figure 4.1 split into three distinct phases.

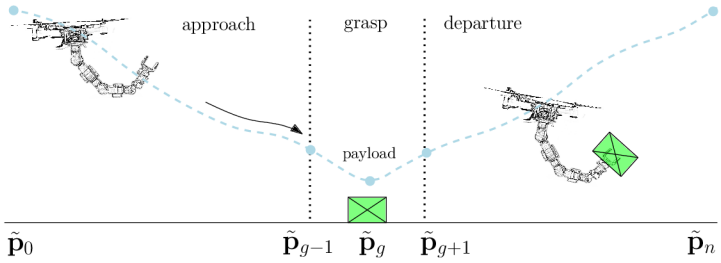

Figure 4.1: Autonomous grasp maneuver with three distinct phases.

$\mathcal{L}_1$-adaptive control augmentation loop is subsequently considered to compensate for the dynamic effects of the manipulator and the payload. The architecture assumes a visual servoing approach in the grasp maneuver flight phase. A complete system architecture of the UAV and manipulator control loop is shown in Figure 4.2. The maneuver planner implements the specification of the reference values $\tilde{\mathbf{p}}_0, \tilde{\mathbf{p}}_{g-1}, \tilde{\mathbf{p}}_g, \tilde{\mathbf{p}}_{g+1}$ and $\tilde{\mathbf{p}}_n$. The carrot chasing guidance law then outputs a reference $\tilde{\mathbf{x}}^q$. A nonlinear MPC block computes the control inputs $\tilde{\mathbf{u}}^q$ for the UAV, and $\mathbf{u}^m$ for the manipulator. The $\mathcal{L}_1$-adaptive augmentation block adopted from [43] modifies the input $\tilde{\mathbf{u}}^q$ to $\mathbf{u}^q$ to reduce the coupling effects of the UAV and manipulator system, and increase the sampling rate of the control loop.

The computer vision and tracking block has the task of payload detection and tracking assuming a 3D stereo camera attached at the end-effector. Based on the estimated position of the

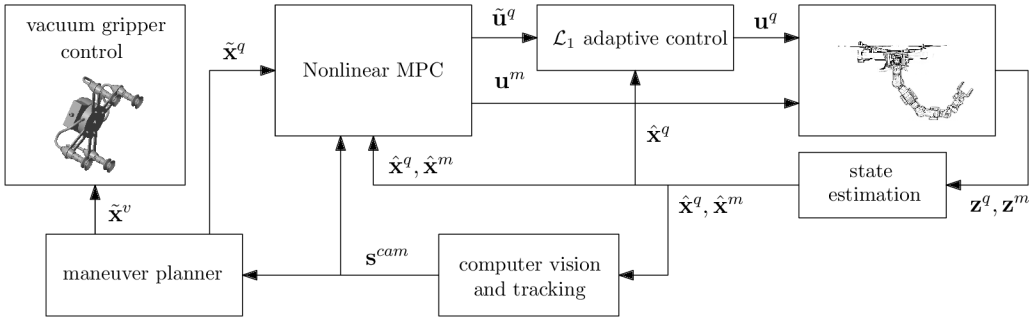

Figure 4.2: Control system architecture of UAV coupled with a manipulator.

payload relative to the end-effector, the vacuum gripper is activated to grasp the payload on contact. Finally, the state estimation algorithm (either a standard EKF or optimization-based MHE) uses the available measurements from the system sensors to estimate the complete state vector of the UAV and a manipulator. The controlled system is a quasi-physical model implemented in Simscape®.

4.1 NONLINEAR CONTROLLER DESIGN

The proposed MPC controller for the quadcopter control uses the following dynamic constraints given by equation (4.1). The translational and rotational dynamics predict the UAV state [74] using the vector $\mathbf{x} = [\mathbf{p}_N \ \mathbf{v}_B \ \mathbf{q}_{B/N} \ \boldsymbol{\omega}_B]^T$. The forces and moments are expressed in a simplified manner allocating the propeller angular velocities. The angular velocity control vector is normalized using the upper bound $\mathbf{u}_{ub}$ and lower bound $\mathbf{u}_{lb}$ such that $\omega_i^{norm} \in (0,1)$. The normalization helps the convergence of the OCP. The control vector yields $\mathbf{u} = [\omega_1^{norm} \ \omega_2^{norm} \ \omega_3^{norm} \ \omega_4^{norm}]^T$.

$$\mathbf{x}_{k+1} = \mathbf{f}_{quad}(\mathbf{x}_k, \mathbf{u}_k, \delta t) = \mathbf{x}_k + \begin{bmatrix} \mathbf{R}_{N/B}\mathbf{v}_B\delta t \\ \left(\frac{1}{m}\mathbf{F}_B(\mathbf{x}_k,\mathbf{u}_k) - \boldsymbol{\omega}_B \times \mathbf{v}_B + \mathbf{R}_{B/N}\,\mathbf{g}_N\right)\delta t \\ \frac{1}{2}\boldsymbol{\Omega}^s(\boldsymbol{\omega}_B)\mathbf{q}_{B/N}\delta t \\ \mathbf{I}_B^{-1}\left(\mathbf{M}_B(\mathbf{x}_k,\mathbf{u}_k) - [\boldsymbol{\omega}_{B/N}\times]\,\mathbf{I}_B\boldsymbol{\omega}_{B/N}\right)\delta t \end{bmatrix} \tag{4.1}$$

In a similar manner, a simplified double integrator model for the MPC of the robotic manipulator is proposed. The discretized kinematic equations assuming the state vector $\mathbf{x} = [\Theta \ \dot{\Theta}]^T$ for the joint motion can be described by equation (4.2). The control vector yields angular acceleration of the respective joints $\mathbf{u} = [\ddot{\theta}_1 \ \ddot{\theta}_2 \ \ddot{\theta}_3 \ \ddot{\theta}_4 \ \ddot{\theta}_5 \ \ddot{\theta}_6]^T$.

$$\mathbf{x}_{k+1} = \mathbf{f}_{arm}(\mathbf{x}_k, \mathbf{u}_k, \delta t) = \mathbf{x}_k + \begin{bmatrix} \dot{\Theta}\delta t + \frac{1}{2}\mathbf{u}\delta t^2 \\ \mathbf{u}\delta t \end{bmatrix} \tag{4.2}$$

Alternatively to the implemented double integrator kinematics of the joints, MPC can be applied as a torque controller when using the whole dynamics.

We construct the following optimization problem following a single-shooting/multiple-shooting scheme [2] resulting in OCP defined by a cost function, dynamic constraints for the

UAV and the manipulator with respective initial condition, grasp constraints and state/input constraints given by problem (4.4). Based on solver convergence, the single or multiple shooting scheme will be utilized in the rest of the experiments. The problem definition in this chapter differentiates the state vector of the multirotor vehicle $\mathbf{x}^q$ from state vector of the robotic manipulator $\mathbf{x}^m$. State and control input upper and lower bounds for the UAV system $\mathbf{x}^q_{lb}, \mathbf{x}^q_{ub}, \mathbf{u}^q_{lb}, \mathbf{u}^q_{ub}$ and the manipulator $\mathbf{x}^m_{lb}, \mathbf{x}^m_{ub}, \mathbf{u}^m_{lb}, \mathbf{u}^m_{ub}$ can be specified to prevent maneuvers outside of the operational envelope of the vehicle. The cost function tailored for the aerial manipulation tasks is composed of the following sum of the objectives described in (4.3).

$$\mathcal{J}(\mathbf{x}^q, \mathbf{x}^m, \mathbf{u}^q, \mathbf{u}^m) = \mathcal{J}_t(\mathbf{x}^q) + \mathcal{J}_p(\mathbf{x}^q, \mathbf{x}^m) + \mathcal{J}_u(\mathbf{u}^q, \mathbf{u}^m) + \mathcal{J}_g(\mathbf{x}^q, \mathbf{x}^m) \tag{4.3}$$

The cost function is composed of the tracking cost $\mathcal{J}_t(\mathbf{x}^q)$, perception cost $\mathcal{J}_p(\mathbf{x}^q, \mathbf{x}^m)$, and the control effort cost $\mathcal{J}_u(\mathbf{u}^q, \mathbf{u}^m)$.

$$\begin{aligned}
\mathcal{J} = \min_{\mathbf{u}^q_{1:K}, \mathbf{u}^m_{1:K}} \; & \mathcal{J}(\mathbf{x}^q_K, \mathbf{x}^m_K) + \sum_{k=0}^{K-1} \mathcal{J}(\mathbf{x}^q_k, \mathbf{x}^m_k, \mathbf{u}^q_k, \mathbf{u}^m_k) \\
\text{subj.to} : \; & \mathbf{x}^q_{k+1} = \mathbf{f}_{quad}(\mathbf{x}^q_k, \mathbf{u}^q_k, \delta t), \quad \mathbf{x}^q_0 = \hat{\mathbf{x}}^q_0 \\
& \mathbf{x}^m_{k+1} = \mathbf{f}_{arm}(\mathbf{x}^m_k, \mathbf{u}^m_k, \delta t), \quad \mathbf{x}^m_0 = \hat{\mathbf{x}}^m_0 \\
& \boldsymbol{\Omega}(\mathbf{x}^q_k, \mathbf{x}^m_k) \leq 0 \\
& \mathbf{x}^q_{lb} < \mathbf{x}^q_k < \mathbf{x}^q_{ub}, \quad \mathbf{u}^q_{lb} < \mathbf{u}^q_k < \mathbf{u}^q_{ub} \\
& \mathbf{x}^m_{lb} < \mathbf{x}^m_k < \mathbf{x}^m_{ub}, \quad \mathbf{u}^m_{lb} < \mathbf{u}^m_k < \mathbf{u}^m_{ub}
\end{aligned} \tag{4.4}$$

The described problem can be solved by standard NLP solvers such as IPOPT, or one of many implementations of the SQP solvers. We implement the OCP formulation using the CasADi package and solve it using the PANOC solver with penalty implemented alongside the augmented Lagrangian method in a dedicated Optimization Engine (OpEn). Low memory requirements and computational speed make PANOC a very effective algorithm for embedded applications utilizing MPC. Code generation function of OpEn is used to store the final design. The solver can subsequently be invoked by TCP/IP interface in Simulink® to run the solver for the quasi-physical model simulation.

In order to control the end-effector pose with respect to the vehicle coordinate system $\mathcal{F}_B$ and the inertial reference system $\mathcal{F}_N$, we define the end-effector or camera coordinate system $\mathcal{F}_C = (x_C, y_C, z_C, O_C)$. We assume that this coordinate system is coincident with the camera coordinate system similar to [102]. The end-effector pose with respect to the UAV body $\mathbf{Z}^B_{end} \in \mathbb{R}^{4\times4}$ and the inertial frame $\mathbf{Z}^N_{end}$ are computed from the joint angles by utilizing the DH convention so that the successive joint and link transformations form the end-effector pose. The pose $\mathbf{Z}^B_{end}$ contains the end-effector position $\mathbf{p}^c_B \in \mathbb{R}^3$ and orientation $\mathbf{R}_{B/C} \in \mathbb{R}^{3\times3}$. The end-effector position $\mathbf{p}^c_N \in \mathbb{R}^3$ in the inertial frame and the orientation with respect to inertial frame $\mathbf{R}_{C/N}$ can be computed by equation (4.5).

$$\begin{aligned}
\mathbf{p}^c_N &= \mathbf{p}_N + \mathbf{R}^T_{B/N} \mathbf{p}^c_B \\
\mathbf{R}_{C/N} &= \mathbf{R}^T_{B/N} \mathbf{R}_{B/C}
\end{aligned} \tag{4.5}$$

Considering the estimated payload position and orientation $\hat{\mathbf{p}}^{N}_{P/N}$ and $\hat{\mathbf{R}}_{P/N}$ respectively, which are assumed to be detected by a 3D Stereo Camera sensor, we can define a payload reference frame $\mathcal{F}_P = (x_P, y_P, z_P, O_P)$. Alternatively, a cost function term can be defined considering the error of the end-effector position and orientation given by equations (4.6)

$$\begin{aligned} \Delta_p &= \hat{\mathbf{p}}^{N}_{P/N} - \mathbf{p}^{N}_{C/N} \\ \Delta_r &= \operatorname{diag}\left(\mathbf{R}^{T}_{C/N}\hat{\mathbf{R}}_{P/N}\right) - 1 \end{aligned} \tag{4.6}$$

The grasping quadratic error term can then be expressed by equation (4.7).

$$\mathcal{J}_g = [\Delta_p \ \Delta_r]\, \mathbf{G}_{MPC}\, [\Delta_p \ \Delta_r]^T \tag{4.7}$$

The grasp constraints are the path constraints $\boldsymbol{\Omega}$ of the problem and will be enforced in the grasp phase resulting in constraints (4.8), where $\lambda^{tr}(\hat{\mathbf{p}}^{N}_{P/N}, \mathbf{p}^{N}_{C/N})$ is a phase transition function which ensures strict enforcement of the constraints in the grasp phase, and relaxes the constraints in the approach and depart phase.

$$\begin{aligned} \|\Delta_p\|^2 &\leq \lambda^{tr}(\hat{\mathbf{p}}^{N}_{P/N}, \mathbf{p}^{N}_{C/N}) \\ \|\Delta_r\|^2 &\leq \lambda^{tr}(\hat{\mathbf{p}}^{N}_{P/N}, \mathbf{p}^{N}_{C/N}) \end{aligned} \tag{4.8}$$

Figure 4.3 shows a visualization of a multirotor UAV with an attached robotic arm. For clarity, all previously introduced reference frames and corresponding transformations are also visualized.

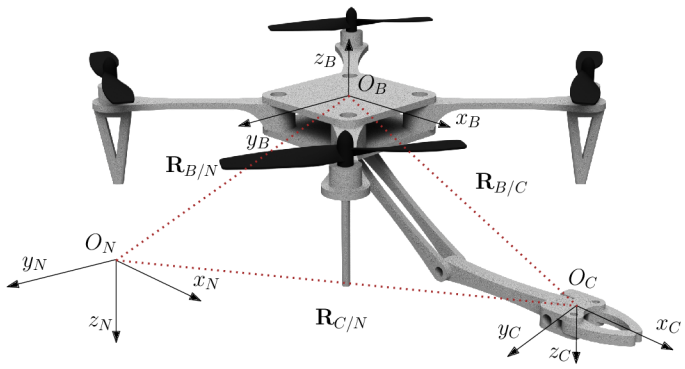

Figure 4.3: Visualization of a coupled system with respective reference frames.

The cost function (4.3) will now be described. The tracking cost is designed to guide the UAV from point of approach $\tilde{\mathbf{p}}_0$ to point of grasp transition $\tilde{\mathbf{p}}_{g-1}$. The grasp phase consists of point $\tilde{\mathbf{p}}_g$ and departure transition point $\tilde{\mathbf{p}}_{g+1}$, and, finally, the maneuver ends with the nominal trajectory transition point $\tilde{\mathbf{p}}_n$. The tracking cost function (4.9) takes the form of a quadratic cost function. The reference point $\tilde{\mathbf{p}}$ of reference state $\tilde{\mathbf{x}}^q$ changes linearly from $\tilde{\mathbf{p}}_0$ to $\tilde{\mathbf{p}}_n$ by integration of a simple carrot chasing law ensuring a smooth transition of the reference points.

$$\mathcal{J}_t(\mathbf{x}^q) = [\mathbf{x}^q - \tilde{\mathbf{x}}^q]^T\, \mathbf{Q}_{MPC}\, [\mathbf{x}^q - \tilde{\mathbf{x}}^q] \tag{4.9}$$

Let $\mathbf{s}^{cam} = [u\ v]^T$ represent image plane coordinates. Following the formulation in [33] and assuming a pinhole camera model, the transformation of world coordinates $\hat{\mathbf{p}}^N_{P/N}$ to image plane coordinates is performed by equation (4.10), where $\mathbf{K}^{cam} \in \mathbb{R}^{3\times3}$ is the camera intrinsic calibration matrix and w^s is a scale factor inferred using a depth map.

$$w^s\ \mathbf{s}^{cam} = \mathbf{K}^{cam}\left[\mathbf{R}_{B/C}|\mathbf{p}^c_B\right]\hat{\mathbf{p}}^C_{P/N} \tag{4.10}$$

Expressing the horizontal and vertical FoV of the camera by $\Psi^{fov} = [\alpha_h\ \alpha_v]^T$, the inequality (4.11) checks if the target lies within the ellipsoid placed around its image center.

$$\frac{u^2}{(\tan\alpha_v/2)^2} + \frac{v^2}{(\tan\alpha_h/2)^2} < 1 \tag{4.11}$$

The perception cost function is described in equation (4.12) relaxing the constraint and placing it to the logistic sigmoid function σ^{sg} with free parameters Δ_1 and Δ_2. Figure 4.4 shows a schematics of the image plane with a detected target. The blue circle represents a shape of the sigmoid function penalizing target detection further from the image center. The presence of

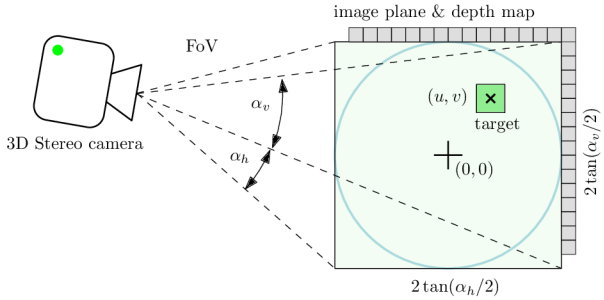

Figure 4.4: Schematics of soft perception constraints.

the phase transition function weights the perception term positively before the grasp phase. The term becomes negligible in the departure phase.

$$\mathcal{J}_p(\mathbf{x}^q, \mathbf{x}^m) = \left[1 - \lambda^{tr}(\hat{\mathbf{p}}^N_{P/N}, \mathbf{p}^N_{C/N})\right]\ \sigma^{sg}\left(\frac{u^2}{(\tan\alpha_v/2)^2} + \frac{v^2}{(\tan\alpha_h/2)^2}, \Delta_1, \Delta_2\right) \tag{4.12}$$

The control effort penalty composed of the manipulator joint acceleration penalty weighted by $\mathbf{R}^m_{MPC}$, and the UAV rotor angular velocity penalty weighted by $\mathbf{R}^q_{MPC}$ is described in equation (4.13) .

$$\mathcal{J}_u(\mathbf{u}^q, \mathbf{u}^m) = [\mathbf{u}^q]^T\,\mathbf{R}^q_{MPC}\mathbf{u}^q + [\mathbf{u}^m]^T\,\mathbf{R}^m_{MPC}\mathbf{u}^m \tag{4.13}$$

The defined cost function ensures a smooth transition through all phases of the grasp maneuver while considering the payload position in the camera FoV and the control effort.

There are more constraints which might be useful in the problem design process. As for now, the kinematics of the arm does not define a safe space for the arm to operate. In practice, that could result in a failure if the arm performs a collision with the UAV airframe. For this reason, second-order cone constraints, as shown in Figure 4.5, restrict the end-effector position to a defined operational space. The second-order cone is specified by a constant $\alpha_l \in \mathbb{R}^+$, which defines the cone shape. Increasing the parameter α_l results in a flatter cone shape. The cone origin is shifted in BFF to a position $\mathbf{p}^l_B = [x^l_B\ y^l_B\ z^l_B]$. The direction of the

cone is set to allow for the end-effector movement in front and under the UAV. The shape of the cone can be specified by the safety requirements such that the arm does not operate directly under any of the propellers.

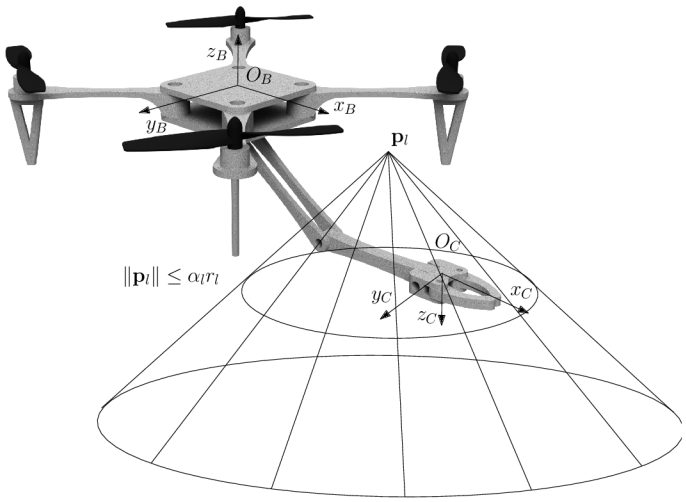

Figure 4.5: Visualization of a second-order cone safety constraint.

The equation (4.14) defines the second-order cone constraints assuming the shifted origin of the cone.

$$\sqrt{(x_B - x_B^l)^2 + (y_B - y_B^l)^2} \leq -\alpha_l (z_B + z_B^l) \tag{4.14}$$

To conclude, table 4.1 summarizes the constraints which can be imposed on the coupled problem. Trade-offs with respect to computational time have to be considered when applying these constraints.

Table 4.1: Overview of constraints for aerial grasping.

Name	Type	Parameters	Severity
Operational space	Second-order cone	$[\alpha_l]$	safety-critical
Target perception	Euclidian ball	$[\Delta_1\ \Delta_2]$	mission-critical
Effector grasp	Simplex	$[\lambda^{tr}]$	mission-critical

4.2 ADAPTIVE AUGMENTATION LOOP DESIGN

The $\mathcal{L}_1$-adaptive augmentation loop implements a so-called reference model and estimates the matched and unmatched uncertainties in the vehicle motion [45], [120]. The translational acceleration uncertainties $\boldsymbol{\zeta}^t = \begin{bmatrix}\zeta_x^t & \zeta_y^t & \zeta_z^t\end{bmatrix} \in \mathbb{R}^3$ and angular acceleration uncertainties $\boldsymbol{\zeta}^a = \begin{bmatrix}\zeta_x^a & \zeta_y^a & \zeta_z^a\end{bmatrix} \in \mathbb{R}^3$ are assumed to be estimated by the augmentation loop. The matched uncertainties are composed of vertical linear acceleration and all angular velocities $\boldsymbol{\sigma}_m = \begin{bmatrix}\zeta_z^t & \zeta_x^a & \zeta_y^a & \zeta_z^a\end{bmatrix}^T \in \mathbb{R}^4$ while unmatched uncertainties are linear accelerations in the horizontal plane of the UAV such that $\boldsymbol{\sigma}_{um} = \begin{bmatrix}\zeta_x^t & \zeta_y^t\end{bmatrix}^T \in \mathbb{R}^2$. Let $\mathbf{u}_{\mathcal{L}_1}^g \in \mathbb{R}^4$ be the adaptive

control input [123] such that the final control input $\mathbf{u}^q \in \mathbb{R}^4$ is computed as in equation (4.15).

$$\mathbf{u}^q = \tilde{\mathbf{u}}^q + \mathbf{u}^q_{\mathcal{L}_1} \tag{4.15}$$

The compensation control input is computed at every time step of the augmented loop by applying a strictly proper stable low-pass filter on the estimated matched uncertainties. It is applied in discrete time as in equation (4.16), where $\omega_{co} \in \mathbb{R}$ is a cutoff frequency of the low-pass filter, and $\delta t \in \mathbb{R}$ is the sampling time of the filter, which can be set differently from the sampling time of nonlinear MPC.

$$\mathbf{u}^g_{\mathcal{L}_1,k} = \mathbf{u}^g_{\mathcal{L}_1,k-1} e^{-\omega_{co}\delta t} - \sqrt{\frac{\hat{\sigma}_{m,k}}{\rho\pi C_T R^4_{pr}}}(1 - e^{-\omega_{co}\delta t}) \tag{4.16}$$

Following the computation of the augmented control input, the $\mathcal{L}_1$ observer update is performed as in equation (4.17) assuming reduced state estimate $\mathbf{z}^l = [\mathbf{v}_N \ \boldsymbol{\omega}_B]^T$ and $\tilde{\mathbf{z}}^l \in \mathbb{R}^6$ is the state estimate coming from the standard state estimator.

$$\hat{\mathbf{z}}^l_{k+1} = \hat{\mathbf{z}}^l_k + \left[\mathbf{F}^{\mathcal{L}_1}_{1,k} + \mathbf{F}^{\mathcal{L}_1}_{2,k}\left(\mathbf{u}^g_{\mathcal{L}_1,k} + \hat{\boldsymbol{\sigma}}_{m,k}\right) + \mathbf{F}^{\mathcal{L}_1}_{3,k}\hat{\boldsymbol{\sigma}}_{um,k} + \mathbf{A}_s\tilde{\mathbf{z}}^l_k\right]\delta t \tag{4.17}$$

The nominal dynamics term $\mathbf{F}^{\mathcal{L}_1}_{1,k} \in \mathbb{R}^6$ can be derived as in equation (4.18), where the nominal thrust force $\mathbf{F}^t_N$ is derived from the MPC control inputs.

$$\mathbf{F}^{\mathcal{L}_1}_{1,k} = \begin{bmatrix} \mathbf{g}_N + \left(\mathbf{F}^t_N + \mathbf{F}^d_N\right) m^{-1} \\ \mathbf{I}_B^{-1}\mathbf{M}_B - [\boldsymbol{\omega}_B\times]\,\mathbf{I}_B\boldsymbol{\omega}_B \end{bmatrix} \tag{4.18}$$

The dynamic term of the matched uncertainty $\mathbf{F}^{\mathcal{L}_1}_{2,k} \in \mathbb{R}^{6\times4}$ derived at a thrust level is defined by equation (4.19), where $\mathbf{R}^z_{B/N}$ is the column representing body z-axis orientation transformation. At the same time, the angular velocity uncertainties are converted to individual rotor thrust components by the allocation matrix $\boldsymbol{\Upsilon} \in \mathbb{R}^{3\times4}$.

$$\mathbf{F}^{\mathcal{L}_1}_{2,k} = \begin{bmatrix} \frac{\mathbf{R}^z_{B/N}}{m} & \frac{\mathbf{R}^z_{B/N}}{m} & \frac{\mathbf{R}^z_{B/N}}{m} & \frac{\mathbf{R}^z_{B/N}}{m} \\ \multicolumn{4}{c}{\mathbf{I}_B^{-1}\boldsymbol{\Upsilon}} \end{bmatrix} \tag{4.19}$$

Finally, the term defining the unmatched uncertainty component of the dynamics $\mathbf{F}^{\mathcal{L}_1}_{3,k} \in \mathbb{R}^{6\times2}$ is derived as in equation (4.20) with the orientation transformation of body x and y components represented by $\mathbf{R}^x_{B/N}$ and $\mathbf{R}^y_{B/N}$.

$$\mathbf{F}^{\mathcal{L}_1}_{3,k} = \begin{bmatrix} \frac{\mathbf{R}^x_{B/N}}{m} & \frac{\mathbf{R}^y_{B/N}}{m} \\ 0 & 0 \end{bmatrix} \tag{4.20}$$

Based on the defined components of the dynamics and the observer structure [122], the adaptation law takes the form of equation (4.21), where $\mathbf{A}_s \in \mathbb{R}^{6\times6}$ is the adaptation gain Hurwitz matrix.

$$\begin{bmatrix} \hat{\boldsymbol{\sigma}}_{m,k} \\ \hat{\boldsymbol{\sigma}}_{um,k} \end{bmatrix} = -\mathbf{E}\left[\mathbf{F}^{\mathcal{L}_1}_{1,k}\ \mathbf{F}^{\mathcal{L}_1}_{2,k}\right]^{-1}\boldsymbol{\Phi}^{-1}_{\mathcal{L}_1}\exp(\mathbf{A}_s\delta t)(\hat{\mathbf{z}}^l_k - \tilde{\mathbf{z}}^l_k) \tag{4.21}$$

The term $\mathbf{\Phi}_{\mathcal{L}_1} \in \mathbb{R}^6 \times \mathbb{R}^6$ is defined by equation (4.22) and is a fixed matrix computed based on the adaptation gain matrix.

$$\mathbf{\Phi}_{\mathcal{L}_1} = \mathbf{A}_s^{-1} \left[\exp(\mathbf{A}_s \delta t) - \mathbf{E}\right] \tag{4.22}$$

The unmatched uncertainties are not compensated for by the adaptation law. However, they improve the overall state estimate. They can be subsequently compensated by the nonlinear MPC being the baseline controller.

4.3 WEIGHT ADAPTATION

The process of MPC cost function weight selection requires a significant manual effort and domain expertise, thus often leading to a suboptimal design of the controller [58]. Furthermore, the weight setting may be valid only for specific operating conditions which were assumed during the selection process. The objective adaptation process gains importance especially for coupled dynamical systems such as the one considered since the preference with regards to UAV motion or robotic manipulator motion may not be intuitive. Decision vector (4.23), which includes objective function weights, adaptation gains and cutoff frequencies of the controller, is considered to be optimized. In experiments without $\mathcal{L}_1$ adaptation law, only the first three vectors are considered.

$$\mathbf{z}_w = \left[\mathrm{vec}(\mathbf{Q}_{MPC})^T \ \mathrm{vec}(\mathbf{R}_{MPC})^T \ \mathrm{vec}(\mathbf{G}_{MPC})^T \ \mathrm{vec}(\mathbf{A}_s)^T \ \boldsymbol{\omega}_{co}\right] \tag{4.23}$$

The parameter vector can be set to a fixed set of values, or changed based on a suitable feature vector. Because of a greater adaptability to specific tasks and the overall mission considered, weight adaptation will be mainly considered. Generally speaking, the parameter setting $\mathbf{z}_w$ is a function of input feature vector $\mathbf{y}_w$ and parameter vector $\mathbf{\Theta}_w$ as in equation (4.24).

$$\mathbf{z}_w = \mathbf{f}_w(\mathbf{y}_w, \mathbf{\Theta}_w) \tag{4.24}$$

This description perfectly fits ANN. Due to the complexities involving the application of a back-propagation algorithm for this task, we assume a derivative-free optimization approach based on CEM for tuning the network weights $\mathbf{\Theta}_w$. In ML community, the approach of neural network training using iterative sample selection and evaluation is often referred to as Deep Reinforcement Learning (DRL). The architecture of the weight adaptation loop is shown in Figure 4.6.

Compared to the direct use of RL as the inherent control law, which requires a substantial number of simulations and larger network architectures while being susceptible to robustness issues, our approach leverages the optimal control formulation of the control law and uses RL to appropriately modify the control law objective function at runtime in order to adapt to mission needs. Naturally, a question arises with regards to feature vector $\mathbf{y}_w$ selection. What would be the features which could be relevant in the context of controller parameter/objective adaptation? Assuming the weight adaptation for a coupled system, the following set of features was chosen, representing:

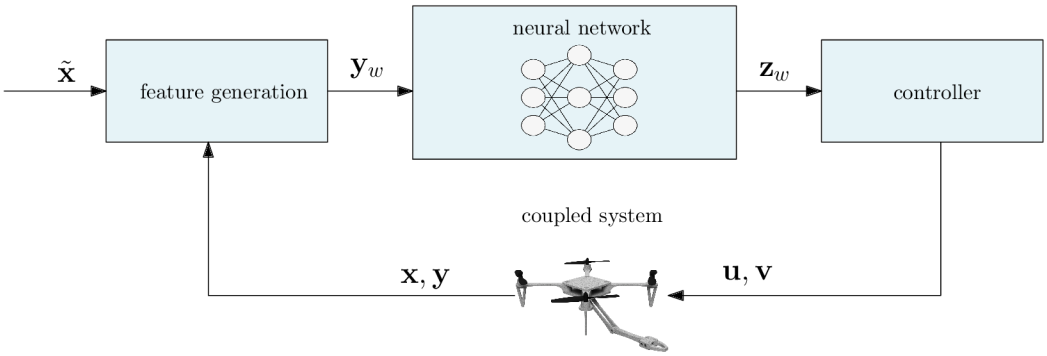

Figure 4.6: Schematics of weight adaptation loop trained by DRL approach.

1. **Slant range** – current distance to payload. This feature is critical with respect to prioritizing the end-effector's positioning and orientation rather than vehicle path tracking.

2. **End-effector velocity** – velocity of the end-effector with respect to the inertial frame. This quantity is important in the terminal maneuvering stage.

3. **End-effector angular velocity** – angular velocity of the end-effector with respect to the inertial frame. This quantity is important in the terminal maneuvering stage.

4. **Vehicle velocity** – current vehicle speed is also a valid feature that affects the tracking performance. At higher speeds, the prediction horizon of the controller may be too short. It is computed as $v_f = \|\mathbf{v}^B_{B/N}\|_2$.

5. **Cross-track error** – the distance between the UAV position and flight profile, calculated transversely to the course. The UAV can temporarily deviate from a given path to perform the grasping maneuver with a higher accuracy. It is denoted by e_f.

6. **Horizontal error** – the distance between the current and the expected position at a given time. An increase in speed typically reduces horizontal error. It is denoted by l_f.

The mentioned features form an input vector $\mathbf{y}_w \in \mathbb{R}^{10}$ and an output for the controller parametrization at a given time. The slant range can be computed using the Euclidian distance $r_f = \|\hat{\mathbf{p}}^p_N - \mathbf{p}_N\|_2$. The end-effector's velocity $\mathbf{v}^B_{C/B} \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega}^B_{C/B} \in \mathbb{R}^3$ can be expressed by the manipulator Jacobian $\mathbf{J}(\mathbf{q})$ as in equation (4.25).

$$\begin{bmatrix} \mathbf{v}^B_{C/B} \\ \boldsymbol{\omega}^B_{C/B} \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{4.25}$$

Combining the translation and angular velocities of the robotic arm with the UAV yields expression (4.26) for velocity $\mathbf{v}^B_{C/N} = [v_x \ v_y \ v_z]^T$, and angular velocity $\boldsymbol{\omega}^B_{C/N} = [\omega_x \ \omega_y \ \omega_z]^T$ of the end-effector's frame with respect to the inertial frame.

$$\begin{bmatrix} \mathbf{v}^{B}_{C/N} \\ \boldsymbol{\omega}^{B}_{C/N} \end{bmatrix} = \begin{bmatrix} \mathbf{v}^{B}_{B/N} + \mathbf{v}^{B}_{C/B} + \boldsymbol{\omega}^{B}_{C/B} \times \mathbf{p}^{B}_{C/B} \\ \boldsymbol{\omega}^{B}_{B/N} + \boldsymbol{\omega}^{B}_{C/B} \end{bmatrix} \tag{4.26}$$

The features are concatenated as in equation (4.27) and used as the ANN inputs. The network structure was kept as a single-layer neural network with a sigmoid activation function limiting the ANN outputs and featuring nonlinearity.

$$\mathbf{y}_w = [r_f \;\; v_x \;\; v_y \;\; v_z \;\; \omega_x \;\; \omega_y \;\; \omega_z \;\; v_f \;\; e_f \;\; l_f]^T \tag{4.27}$$

The ANN outputs are summed with nominal objective function values such that the network varies the weights around a nominal parametrization $\mathbf{z}_w^{nom}$. Since the input layer size is smaller than the output layer size, the resulting ANN can be structured principally in a decoder architecture. The weight adaptation law structure can thus be written as in equation (4.28) with trained weights being $\boldsymbol{\Theta}_w = [\mathrm{vec}(\mathbf{W}) \;\; \mathbf{b}]$. A single layer ANN is beneficial for this kind of application due to lower training demands and explainability of the results.

$$\mathbf{z}_w = \mathbf{f}_w(\mathbf{y}_w, \boldsymbol{\Theta}_w) = \mathbf{z}_w^{nom} + \sigma(\mathbf{W}\mathbf{y}_w + \mathbf{b}) \tag{4.28}$$

4.4 MANEUVER PLANNING

The maneuver planning algorithm is responsible for generating the reference state $\tilde{\mathbf{x}}^q$ to the nonlinear MPC controller. There are two distinct cases to be considered. In the first scenario, the UAV is designated to follow a nominal trajectory $\tilde{\mathbf{p}}$, in which case a carrot chasing nonlinear guidance law is adopted to steer the vehicle to a virtual target point represented by the target state vector $\tilde{\mathbf{x}}^q$. In the second case, the camera is assumed to have detected the target payload. In such case, a set of reference values $\tilde{\mathbf{p}}_0, \tilde{\mathbf{p}}_{g-1}, \tilde{\mathbf{p}}_g, \tilde{\mathbf{p}}_{g+1}$ and $\tilde{\mathbf{p}}_n$ are generated to express a reference descend path to the target and back to the nominal trajectory. The generated points are set according to the control algorithm, which is subsequently switched to the coupled MPC controlling the aerial grasp until the vehicle reaches the nominal trajectory. Simultaneously, the planner sends commands $\tilde{\mathbf{x}}^v$ to the vacuum gripper in the final approach phase to smoothly establish the connection with a payload. From the first detection point, the approach phase starts. For varying detection ranges, the respective approach trajectories are shown in Figure 4.7.

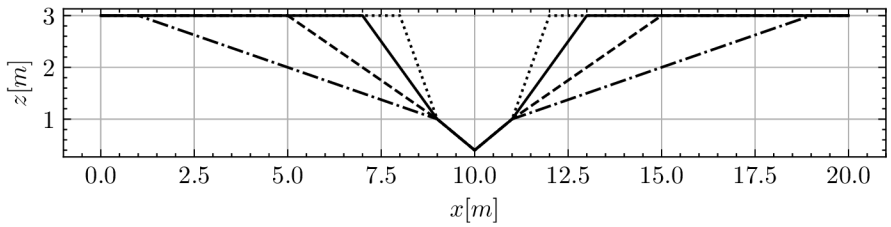

Figure 4.7: Varying approach trajectories to the target payload.

In the vicinity of the payload location, a connection transition phase is designed to follow the same trajectory in all the cases. The return to nominal path is then mirroring the approach phase.

5 RELIABILITY-ASSISTED SYSTEM OPTIMIZATION

As a result of uncertainties, an engineering system may behave outside of its intended nominal range causing a system failure resulting in a possibly unsafe operation or the loss of control. Accounting for uncertainties in the design of the system to increase robustness has been a common engineering practice. On a system level, complex multi-domain interactions may cause hard-to-predict failure events which may not be detected in a subsystem level testing. At the same time, the rigor with regards to uncertainty quantification for CPS has greatly improved by the notion of specifying the ODD of the system. Despite a known structure of the system, a black-box approach to reliability assessment may be necessary to test the system in its ODD. This chapter combines the system reliability assessment with a black-box design optimization technique - making no assessment of the complexity of the objective function - that seek the best parametrization of the system. The goal is to meet the probabilistic requirements on system failure, which may be necessary in the certification process of the system. The proposed Reliability-Based Cross Entropy (RBCE) method combines the failure probability surrogate modeling with design optimization, which will be used in the development phase of the GNC system for a multirotor UAV.

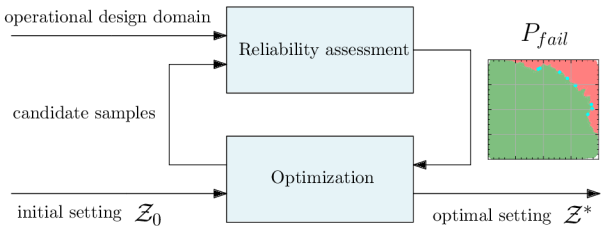

Figure 5.1: Schematics of reliability-assisted system optimization.

Figure 5.1 shows a schematic block diagram of a reliability-assisted system optimization approach employed in this thesis. The probability of failure P_{fail} is used as performance metric of the current design and is used to optimize the design until it meets the reliability requirements.

5.1 CROSS-ENTROPY METHOD

The CEM was originally motivated and developed as an adaptive importance sampling algorithm for estimating rare event probabilities [14]. However, CEM showed to be effective for other classes of problems including combinatorial [24] and continuous optimization [92]. In that sense, CEM can be classified as a stochastic search algorithm among well-known methods such as GA, Simulated Annealing (SA), and others. The CEM method is based on a sequential update of a parametrized sampling distribution, storing the information of best

candidate samples using the sampling distribution.

The notion of cross-entropy describes a metric of probability distribution distance through the Kullback-Leibler (KL) distance. Assuming a random variable ξ, the KL distance $\mathcal{D}$ of probability density functions $\mathcal{P}_1$ and $\mathcal{P}_2$ is described in equation (5.1).

$$\mathcal{D}(\mathcal{P}_1, \mathcal{P}_2) = \mathbb{E}_{\mathcal{P}_1}\left[\log \frac{\mathcal{P}_1(\xi)}{\mathcal{P}_2(\xi)}\right] = \int_{\xi} \mathcal{P}_1(\xi) \log \mathcal{P}_1(\xi) \mathrm{d}\xi - \int_{\xi} \mathcal{P}_1(\xi) \log \mathcal{P}_2(\xi) \mathrm{d}\xi \tag{5.1}$$

A process of minimization of the distance of distributions $\mathcal{P}_1$ and $\mathcal{P}_2$, where the distribution $\mathcal{P}_2$ is parametrized by $\mathfrak{Z}$, is the equivalent of minimizing the cross-entropy defined by equation (5.2) using the KL distance, where $\mathcal{H}(\mathcal{P}_1)$ denotes the entropy of distribution $\mathcal{P}_1$.

$$\mathcal{H}(\mathcal{P}_1, \mathcal{P}_2) = \mathcal{H}(\mathcal{P}_1) + \mathcal{D}(\mathcal{P}_1, \mathcal{P}_2) = -\mathbb{E}\left[\log \mathcal{P}_2(\xi)\right] = -\int_{\xi} \mathcal{P}_1(\xi) \log \mathcal{P}_2(\xi \mid \mathfrak{Z}) \mathrm{d}\xi \tag{5.2}$$

Based on this derivation, the minimization of cross entropy can be rewritten as shown in equation (5.3). The CEM aims to find this minimum as efficiently as possible.

$$\mathcal{J} = \min_{\mathfrak{Z}} -\int_{\xi} \mathcal{P}_1(\xi) \log \mathcal{P}_2(\xi \mid \mathfrak{Z}) \mathrm{d}\xi \tag{5.3}$$

The initial goal of CEM was in the context of rare event estimation, thus the problem formulation can be written as in equation (5.4), where $\gamma^{\mathrm{CE}} \in \mathbb{R}$ is the objective threshold. Hence, the reformulation of cross-entropy minimization of the proposal and unknown distributions to probability estimation problem is performed.

$$p = P_{\mathfrak{Z}}(\mathcal{J}(\xi) \geq \gamma^{\mathrm{CE}}) \tag{5.4}$$

This probability can be rewritten as expectation (5.5) where $\mathbb{1}$ is a function that indicates, whether the inequality holds.

$$p = \mathbb{E}_{\mathfrak{Z}}\left[\mathbb{1}_{(\mathcal{J}(\xi) \geq \gamma^{\mathrm{CE}})}\right] \tag{5.5}$$

The estimation of this probability can be done by the MC approach. However, in the case of small probability estimation, importance sampling strategy is a more adequate approach. The optimization based on importance sampling strategy takes the form of equation (5.6).

$$\mathfrak{Z}^* = \arg\min_{\mathfrak{Z}} -\int_{\xi} \frac{\mathbb{1}_{(\mathcal{J}(\xi) \geq \gamma^{\mathrm{CE}})} \mathcal{P}_2(\xi \mid \mathfrak{Z})}{p} \log \mathcal{P}_2(\xi \mid \mathfrak{Z}) \mathrm{d}\xi \tag{5.6}$$

Since $\mathcal{P}_1$ and $\mathcal{P}_2$ are assumed to belong to the same family of distributions, the importance sampling term is further simplified such that the optimization becomes (5.7).

$$\mathfrak{Z}^* = \arg\min_{\mathfrak{Z}} \mathbb{E}_{\mathfrak{Z}}\left[\mathbb{1}_{(\mathcal{J}(\xi) \geq \gamma^{\mathrm{CE}})} \log \mathcal{P}_2(\xi \mid \mathfrak{Z})\right] \tag{5.7}$$

The CEM is an iterative method involving the update of parameters $\mathfrak{Z}_k$ to $\mathfrak{Z}_{k+1}$ while performing the importance sampling. In practice, a fitting of the new distribution is performed using a set of elite samples. The CEM will now be described in the context of continuous optimization of the black-box objective function.

In a world of continuous derivative-free optimization scenario where the distribution is modeled by a multivariate Gaussian, the update of the parameters in equation (5.8) and (5.9) corresponds to the well-known Maximum Likelihood Estimation (MLE) method, where N^{CE} is the number of elite samples. Figure 5.2 shows a visualization of two steps of a CEM method showing an elite set selection with respect to true objective function shape.

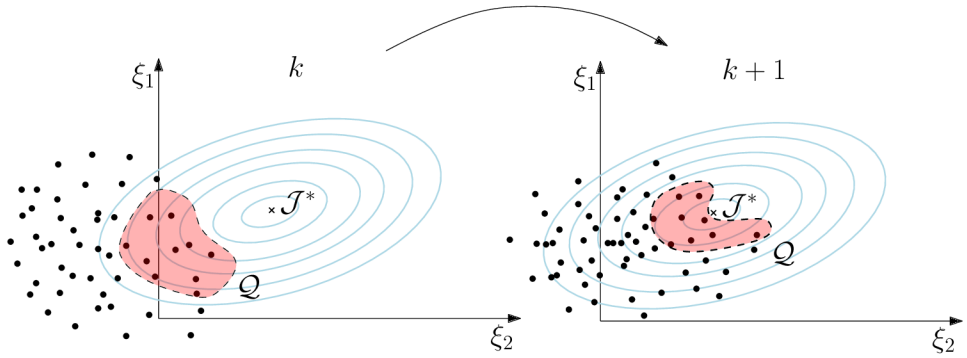

Figure 5.2: Example of the CEM update through the elite set of samples.

The algorithm is started by specifying the initial proposal distribution $\hat{\mu}_0^{CE}$ and $\hat{\sigma}_0^{CE}$.

$$\tilde{\mu}_{k,i}^{CE} = \sum_{j \in \mathcal{Q}} \frac{\mathbf{z}_{i,j}}{N^{CE}} \tag{5.8}$$

The elite samples of each iteration of CEM are stored in the set $\mathcal{Q}$. The drawn samples from the proposal distribution are denoted $\mathbf{z}_1 \dots \mathbf{z}_{M^{CE}}$, where M^{CE} is the number of drawn samples.

$$\tilde{\sigma}_{k,i}^{CE} = \sum_{j \in \mathcal{Q}} \frac{(\mathbf{z}_{i,j} - \hat{\mu}_{k,i}^{CE})^2}{N^{CE}} \tag{5.9}$$

The updated mean and standard deviation of the multivariate Gaussian is updated using equation (5.10), where $\alpha^{CE} \in \mathbb{R}$ is a smoothing factor of the CEM weighting the current proposal distribution derived using the elite samples and the distribution from the last iteration.

$$\begin{aligned} \hat{\mu}_k^{CE} &= \alpha^{CE} \tilde{\mu}_k^{CE} + (1 - \alpha^{CE}) \hat{\mu}_{k-1}^{CE} \\ \hat{\sigma}_k^{CE} &= \alpha^{CE} \tilde{\sigma}_k^{CE} + (1 - \alpha^{CE}) \hat{\sigma}_{k-1}^{CE} \end{aligned} \tag{5.10}$$

One of the possible stopping criteria of the CEM checks the maximum value of the variance. If this value is lower than a fixed threshold $\epsilon^{CE} \in \mathbb{R}$ as written in equation (5.11), the optimization process is stopped.

$$\max_j(\hat{\sigma}_{k,j}^{CE}) < \epsilon^{CE} \tag{5.11}$$

The algorithm is parametrized by a number of drawn samples M^{CE}, number of elite samples N^{CE}, smoothing factor α^{CE}, and stopping criterion threshold ϵ^{CE}. In a combinatorial optimization variant, the stopping criterion often requires a certain percentage of converged bits of the optimization vector.

5.2 POLYNOMIAL CHAOS EXPANSION

The stochastic nature of CPS (cars, aircrafts, drones etc.) and environments they operate in motivate a need to quantify the uncertainties associated with their operation. We may refer to such uncertainties as "*known unknowns*", since the model parameters subject to uncertainty are often known and their statistical moments can be often approximated. MC-based sampling methods are the industry standard when it comes to uncertainty quantification, but they often require a large amount of computational model evaluations. In recent years, surrogate modeling of computational models has been widely used [84] to approximate the behavior of complex systems by a number of evaluations/measurements. The literature on surrogate modeling often mentions PCE [61], krigging method also known as Gaussian Process Regression (GPR) [13], Support Vector Machines (SVMs), and ANNs [16].

The PCE methods have the advantage of requiring only a small amount of problem evaluations and having a high accuracy results in terms of computing statistical moments of the model response. The PCE surrogate modeling approach has often been applied in reliability analysis of structures [78] and CFD uncertainty propagation [56]. Figure 5.3 shows an example of uncertainty quantification through PCE with two random uncorrelated input variables and two response variables.

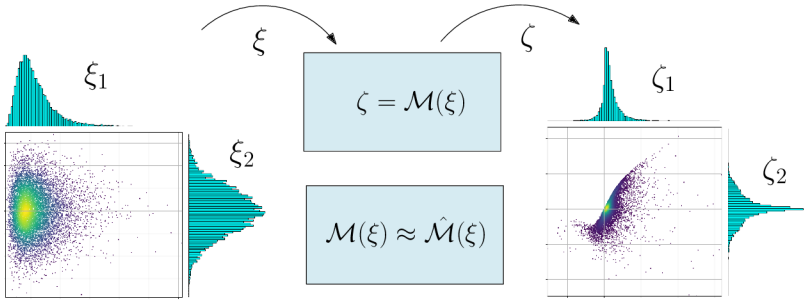

Figure 5.3: Schematics of PCE computational model approximation.

The PCE propagating standard random variables with finite variance $\boldsymbol{\xi} = [\xi_1\ \xi_2\ \ldots \xi_r]^T \in \mathbb{R}^r$ through an arbitrary computational model can be expressed by an infinite weighted sum of orthogonal polynomials in equation (5.12), where β_i are the expansion coefficients and $\boldsymbol{\Psi}_i(\boldsymbol{\xi})$ are the multivariate polynomial basis functions. The surrogate model is denoted as $\hat{\mathcal{M}}$ approximating the true computational model response $\mathcal{M}$. The response random variable vector is denoted as ζ.

$$\zeta = \mathcal{M}(\boldsymbol{\xi}) = \sum_{i=0}^{\infty} \beta_i \boldsymbol{\Psi}_i(\boldsymbol{\xi}) \tag{5.12}$$

The basis functions forming $\boldsymbol{\Psi}_i(\boldsymbol{\xi})$ can be selected through a Wiener-Askey scheme [5] based on the distribution $\boldsymbol{\xi}$. Wiener-Hermite PCE assumes standard normal input variables $\boldsymbol{\xi}_i \sim \mathcal{N}(0, 1)$, which is not satisfied in most applications. Therefore, the modeled probability space is often transformed by a Nataf transformation in order to be modeled as uncorrelated

standard normal space ξ. For practical purposes, the expansion (5.12) is truncated to P terms. The truncated set of basis functions $\mathcal{A}^{r,p}$ is defined in equation (5.13).

$$\mathcal{A}^{r,p} = \left\{ \alpha \in \mathbb{N}^r : | \alpha | = \sum_{i=1}^{r} \alpha_i \leq p \right\} \tag{5.13}$$

The relationship of maximal order of used polynomials p with a number of terms is defined by equations in (5.13).

$$\text{card } \mathcal{A}^{r,p} = \frac{(r+p)!}{r!p!} \tag{5.14}$$

As described in [77], the truncated series in (5.15) have an error term ε. The coefficients β_α can be found by the minimization of ε.

$$\mathcal{M}(\xi) = \sum_{\alpha \in \mathcal{A}^{r,p}} \beta_\alpha \Psi_\alpha(\xi) + \varepsilon \tag{5.15}$$

Least-Square Regression (LSR) on the truncated PCE is used to determine the coefficients. A data set of evaluations $\mathcal{Y}^{PCE}$ for sample points $\mathcal{X}^{PCE}$ of size N^{PCE} is used to minimize the L_2-norm given by equation (5.16). The samples can be selected by various sampling methodologies such as Latin Hypercube Sampling (LHS).

$$\hat{\beta} = \arg \min \frac{1}{N^{PCE}} \sum_{i=1}^{N^{PCE}} \left(\beta^T \Psi(\xi^i) - \mathcal{M}(\mathcal{X}_i^{PCE}) \right)^2 \tag{5.16}$$

The minimization (5.16) has a solution in the form given by (5.17) with the defined data matrix.

$$\begin{aligned} \hat{\beta} &= (\Psi^T \Psi)^{-1} \Psi^T \mathcal{Y}^{PCE} \\ \Psi &= \left\{ \Psi_{i,j} = \Psi_j(\xi^i), i = 1, \ldots, r, j = 0, \ldots, P-1 \right\} \end{aligned} \tag{5.17}$$

The term $\Psi^T \Psi$ is decomposed using Singular Value Decomposition (SVD) to avoid ill-conditioned terms in the computation. The approximation of first and second statistical moments given the trained model can be recovered by (5.18).

$$\begin{aligned} \mu_\zeta &= \mathbb{E}[\zeta] \approx \sum_{i=0}^{P} \beta_i \mathbb{E}[\Psi_i] = \beta_0 \\ \sigma_\zeta^2 &= \mathbb{E}\left[(\zeta - \mu_\zeta)^2 \right] \approx \sum_{i=0}^{P} \beta_i^2 \mathbb{E}[\Psi_i \Psi_i] \end{aligned} \tag{5.18}$$

The response variable mean is denoted as μ_ζ while the variance is denoted as σ_ζ^2. The validation of PCE through its prediction accuracy is often performed using the $\mathcal{R}^2$ coefficient in (5.19) of determination or leave-one-out cross-validation $\mathcal{Q}^2$ as defined in (5.20), which is less prone to overfitting.

$$\mathcal{R}^2 = 1 - \frac{\frac{1}{N^{PCE}} \sum_{i=1}^{N^{PCE}} \left(\mathcal{M}(\xi_i) - \hat{\mathcal{M}}(\xi_i) \right)^2}{\sigma_\zeta^2} \tag{5.19}$$

$$\mathcal{Q}^2 = 1 - \frac{\frac{1}{N^{PCE}} \sum_{i=1}^{N^{PCE}} \left[\frac{\left(\mathcal{M}(\xi_i^j) - \hat{\mathcal{M}}(\xi_i^j) \right)^2}{1 - h_j} \right]}{\sigma_\zeta^2} \tag{5.20}$$

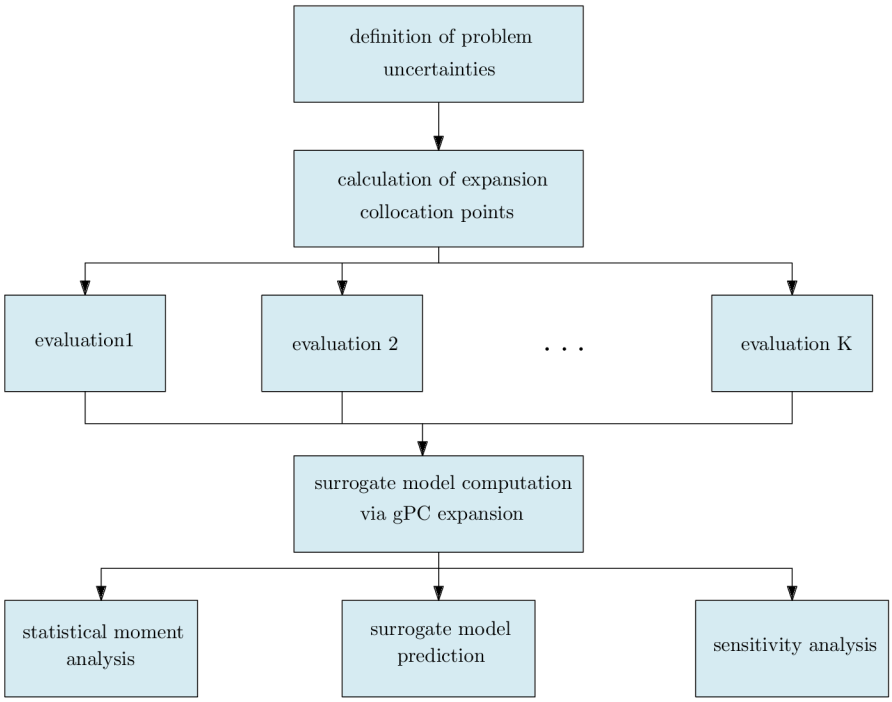

Figure 5.4: General approach for PCE surrogate model computation.

Equation (5.20) utilizes diagonal elements of matrix $\Psi(\Psi^T\Psi)^{-1}\Psi^T$ denoted as h_j. The Q^2 metric always sets a single validation point apart to build a PCE without it such that this process is repeated. The metrics can be useful with regards to the comparison of surrogate models of the scenarios including mathematical and quasi-physical models, thus performing a mathematical model validation through the surrogate operating on the ODD stochastic space.

5.3 RELIABILITY-BASED SYSTEM OPTIMIZATION ALGORITHM

PCE with Kriging meta-model [100] assumes the response probability distribution of a computational model $\mathcal{M}$ to be initially modeled by a weighted set of orthogonal polynomials as in equation (5.12). The estimation of PCE coefficients is obtained by LSR on the truncated PCE. This is achieved by evaluating the model response at a number of sampled points. For simplicity, we use the standard LHS method to sample the ODD. The PCE model is subsequently used to generate a large number of predictions, which are used by the Kriging interpolation method.

The meta-model constructed over PCE surrogate has a structure as defined in equation (5.21), where $\kappa_{GP}^T \mathbf{Y}_{GP}(\xi)$ is the mean value of the underlying Gaussian Process (GP) and $\mathbf{Z}_{GP}(\xi, \omega)$ is zero-mean, unit-variance GP.

$$\mathcal{M}(\xi) \approx \tilde{\mathcal{M}}(\xi) = \kappa_{GP}^T \mathbf{Y}_{GP}(\xi) + \sigma_{GP}^2 \mathbf{Z}_{GP}(\xi, \omega) \tag{5.21}$$

The parameters κ_{GP}^T and σ_{GP}^2 are optimized using least squares in a generalized sense. The estimated Kriging model is essential in order to find additional design points which maximize the expected information gain of the boundary region. Mean $\hat{\mu}_\zeta(\xi)$ and variance $\tilde{\sigma}_\zeta^2(\xi)$

of the Kriging model can be used by a learning function such as the U-function [99] to estimate the probability of miss-classification of each additional candidate design point. Using the formalism of a U-function, the design points are selected from the set of candidates $\mathcal{S}$ using equation (5.22), where a is the limit state parameter specifying failure region points where the true response evaluates $\mathcal{M}(\xi) < a$. In a simple scenario involving a control system, the deterministic requirement may be that the overshoot of the system response must not exceed 15%. In a stochastic formulation, the requirement can be formulated such that the probability of exceeding 15% overshoot must not be larger than 10^{-3} in the defined ODD.

$$\xi^* = \arg\min_{\xi \in \mathcal{S}} \frac{|\,\hat{\mu}_\zeta(\xi) - a\,|}{\hat{\sigma}_\zeta(\xi)} \tag{5.22}$$

Figure 5.5 shows a failure boundary region computed by PCE-Kriging method using six iterations of adaptive re-sampling to accurately model the failure boundary region where the testing model response exceeds a (red region). The example features two-dimensional ODD uniformly distributed with the simple testing function modeling the limit state surface. This process is performed over all candidate controllers in our setting in order to evaluate

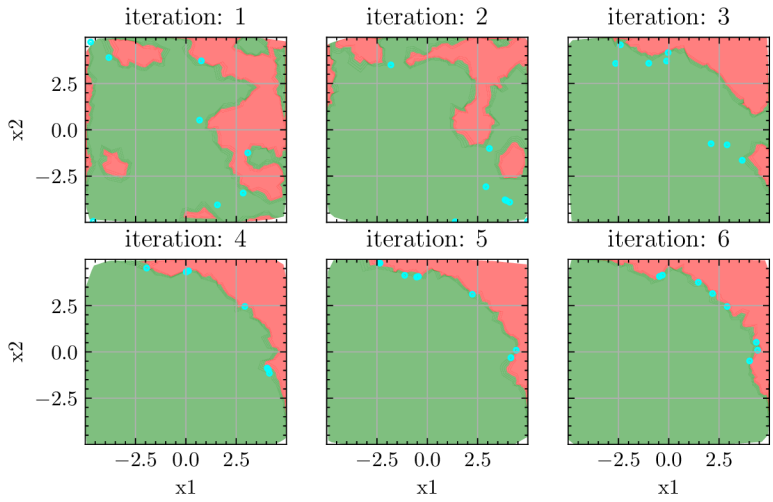

Figure 5.5: PCE-Kriging adaptive resampling for failure boundary refinement.

the probability of failure with respect to the ODD. This statistical approach is featured as the inner loop of Algorithm 5.1. The output of the inner loop is the probability of failure P_{fail}, which can be used by the optimizer to filter out solutions reaching a threshold requirement or augment the objective function to consider this metric in the candidate evaluation.

Our approach combines the CEM method for continuous optimization with reliability approximation of each sample using surrogate modeling to lower the computational requirements of the optimization. We initialize the algorithm using a feasible set of parameters stored in $\hat{\mu}_0^{CE}$, and defined initial covariance matrix $\hat{\sigma}_0^{CE}$. Following the steps of Algorithm 5.1 featuring the proposed RBCE, the optimized set of system parameters $\hat{\mu}^*$ can be retrieved while tracing the feasibility of the solution by estimating the probability of failure P_{fail} with respect to the stochastic nature of the ODD. This approach can be applied to simulations utilizing all types of models such as mathematical, quasi-physical, and digital twins. This thesis

uses the proposed RBCE algorithm in the context of GNC system optimization for the UAV, or the coupled system designated for aerial grasping tasks.

Algorithm 5.1: RBCE Method

Parameters N^{RB}, N^{CE}, ϵ^{CE}, α^{CE}.
Initialize $\hat{\mu}_0^{CE}$ and $\hat{\sigma}_0^{CE}$. Set $k = 1$.
while $\max_j(\hat{\sigma}_{k,j}) > \epsilon^{CE}$ **do**
    Generate candidates $\mathbf{z}_1, \ldots, \mathbf{z}_{N^{RB}}$ from the parameter distribution $\mathcal{N}(\hat{\mu}_k^{CE}, \hat{\sigma}_k^{CE})$.
    if $k > N^{RB}$ **then**
        Sample the ODD distribution (LHS) resulting in samples $\{\mathcal{X}^{PCE}, \mathcal{Y}^{PCE}\}$.
        Construct PCE surrogate model $\hat{\mathcal{M}}$ and meta model $\tilde{\mathcal{M}}$ using current mean $\hat{\mu}_k^{CE}$.
        for all candidates **do**
            Perform adaptive re-sampling to construct final models $\hat{\mathcal{M}}^*$ and $\tilde{\mathcal{M}}^*$.
            Compute P_{fail} of the candidate solution.
        end for
        Evaluate reliability objective function $\mathcal{J}_{RB}$ considering P_{fail} for all candidates.
    else
        Evaluate objective function $\mathcal{J}$ for all candidates.
    end if

    Assuming N^{CE} indices of best performing samples according to $\mathcal{J}$ or $\mathcal{J}^{RB}$ stored in $\mathcal{Q}$, update the sampling distribution:

$$\tilde{\mu}_{k,i}^{CE} = \sum_{j \in \mathcal{Q}} \frac{\mathbf{z}_{i,j}}{N^{CE}} \qquad \tilde{\sigma}_{k,i}^{CE} = \sum_{j \in \mathcal{Q}} \frac{(\mathbf{z}_{i,j} - \hat{\mu}_{k,i}^{CE})^2}{N^{CE}}$$

    Perform smoothing on the sample distribution statistical moments:

$$\hat{\mu}_k^{CE} = \alpha^{CE}\tilde{\mu}_k^{CE} + (1 - \alpha^{CE})\hat{\mu}_{k-1}^{CE} \qquad \hat{\sigma}_k^{CE} = \alpha^{CE}\tilde{\sigma}_k^{CE} + (1 - \alpha^{CE})\hat{\sigma}_{k-1}^{CE}$$

    Update the best solution: $\hat{\mu}_k^{CE,*} = \arg\min\{\mathcal{J}(\hat{\mu}_{k-1}^{CE,*}), \mathcal{J}(\mathbf{z}^*)\}$.
    Update iteration: $k = k + 1$.
end while
Return best solution $\hat{\mu}^{CE,*}$ with the corresponding estimated probability of failure $\hat{P}_{fail}^*$.

The proposed RBCE method initially minimizes the objective function/evaluation metric $\mathcal{J}$. After N^{RB} iterations, the algorithm switches to a mode minimizing the P_{fail} over elite samples. Each iteration initially constructs the surrogate model. For each candidate, a re-sampling is performed to find the correct P_{fail} for that sample. This ensures that the initial phase of optimization focuses on exploration and finding the optimal distribution of the candidates. The final phase increases the reliability by introducing ODD to the problem. The low number of samples required to construct PCE ensures that the computational cost becomes intractable.

6 NUMERICAL RESULTS AND EXPERIMENTS

This chapter provides numerical and experimental results based on the theoretical foundations and researched topics on optimal control with the utilization of the proposed RBCE algorithm for verification and validation of the system. The first part is dedicated to multirotor UAV as a standalone system. The second part examines a control system of the robotic arm manipulator as a standalone system. The third part examines the aerial grasping task assuming a coupled dynamics of a UAV equipped with a robotic manipulator. Tailored performance metrics for the aerial grasping tasks are developed such that they focus on specific mission needs related to the operation of the coupled system.

6.1 OPERATIONAL DESIGN DOMAIN SPECIFICATION

This section focuses on ODD specification in relation to the aerial grasping maneuver. In the context of ODD definition, we specify the probability density function of wind speed conditions, which fits a Weibull probability distribution as shown in Figure 6.1. The specified nominal wind speed then affects the wind turbulence model among others.

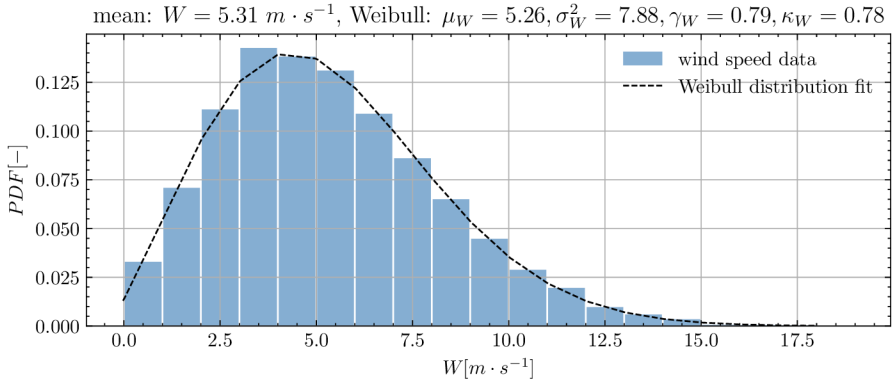

Figure 6.1: Count of measured wind speed fitted by Weibull distribution.

The wind turbulence and wind gusts depend on the nominal wind speed. Since the occurrence density of wind vortex and wind gusts are large enough to allow the control system to compensate it, we do not implement the occurrence density of wind gust and wind vortex to the ODD.

Variables such as time of day, illumination or fog density are typically a part of an ODD since they have a direct impact on computer vision algorithms onboard the vehicle. This thesis assumes involving these effects by forming a probability distribution of target/obstacle first detection slant range. First detection slant range has a direct impact on GNC workload such that the target reach or obstacle avoidance also depend on the maneuverability of the

vehicle. The assumed detection slant range denoted by R_s is assumed to be normally distributed according to Figure 6.2.

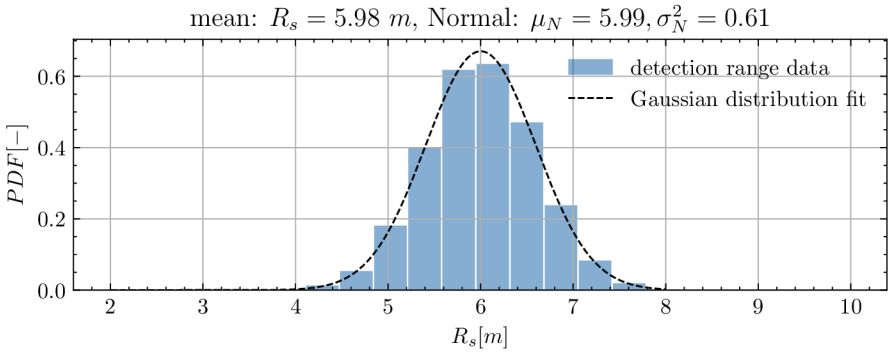

Figure 6.2: Count of target/obstacle detection based on slant range.

In the case of multirotor UAV testing, the detection slant range will be included in the scenarios involving obstacle avoidance, while in aerial grasping the slant range detection will involve the target being manipulated by the robotic arm enhanced UAV.

6.2 TESTING TRAJECTORY SPECIFICATION

Apart from operational domain conditions, conducted simulation experiments assume a set of testing reference trajectories which require varying levels of maneuverability associated to their path curvatures and the corresponding parametrization of the trajectory. Among the reference testing trajectories, this thesis assumes a spiral testing trajectory, lemniscate testing trajectory, diamond testing trajectory, and lobe testing trajectory. An example of each trajectory is shown in Figure 6.3.

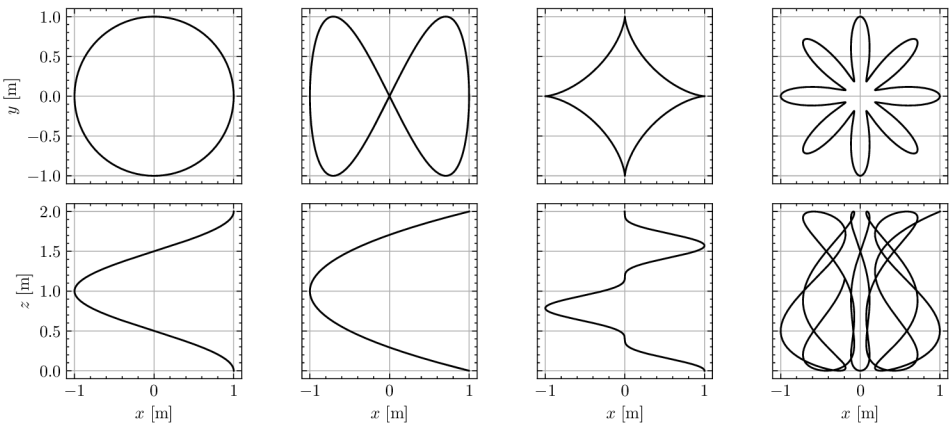

Figure 6.3: Considered testing trajectories projected to two planes.

The considered trajectories can be described by the following time-dependent equations. The spiral trajectory in equation (6.1) parametrized by set $\mathbf{a} = [a_1\ a_2\ a_3\ a_4]$.

$$\begin{aligned} \tilde{x} &= a_1 \cos(a_4 t) \\ \tilde{y} &= a_2 \sin(a_4 t) \\ \tilde{z} &= a_3 t \end{aligned} \tag{6.1}$$

The lemniscate trajectory in equation (6.2) has the following parametrization $\mathbf{a} = [a_1\ a_2\ a_3\ a_4\ a_5]$. Lemniscate trajectory in this form is a closed trajectory, meaning that the UAV can follow this trajectory in a loop.

$$\begin{aligned} \tilde{x} &= a_1 \cos(a_4 t) \\ \tilde{y} &= a_2 \frac{\sin(2a_4 t)}{a_5} \\ \tilde{z} &= \cos(a_3 t) \end{aligned} \tag{6.2}$$

The diamond trajectory in equation (6.3) has the following parametrization $\mathbf{a} = [a_1\ a_2\ a_3\ a_4]$. This formulation of diamond trajectory is open, meaning it has a starting point and an ending point.

$$\begin{aligned} \tilde{x} &= a_1 \cos(a_4 t)^3 \\ \tilde{y} &= a_2 \sin(a_4 t)^3 \\ \tilde{z} &= a_3 t \end{aligned} \tag{6.3}$$

The lobe trajectory in equation (6.4) has the following parametrization $\mathbf{a} = [a_1\ a_2\ a_3\ a_4\ a_5]$. Lobe trajectory in this form is a closed trajectory, meaning that the UAV can follow this trajectory in a loop.

$$\begin{aligned} \tilde{x} &= a_1 [a_2 + \cos(a_3 a_4 t)] \cos(a_4 t) \\ \tilde{y} &= a_1 [a_2 + \cos(a_3 a_4 t)] \sin(a_4 t) \\ \tilde{z} &= \cos(a_5 t) \end{aligned} \tag{6.4}$$

Figure 6.4 shows a simulation of a quadcopter following a specified lemniscate trajectory using a baseline MPC with fixed weighting parameters which were set manually. Naturally,

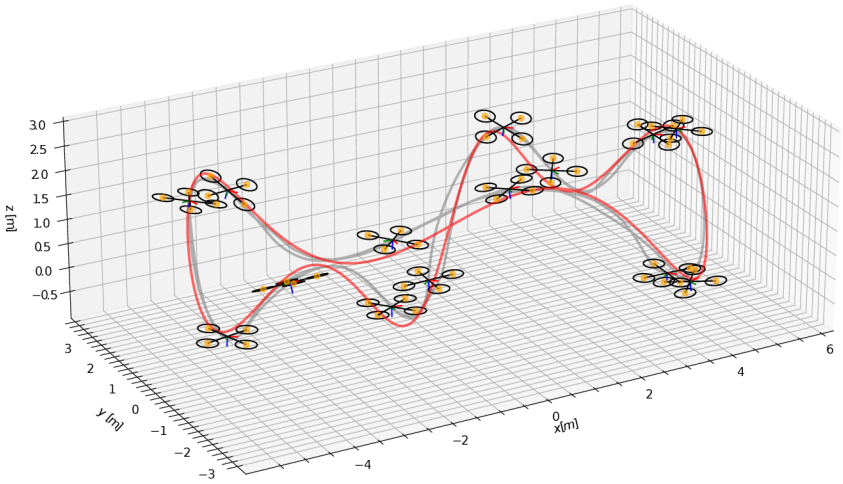

Figure 6.4: Lemniscate trajectory following by a quadcopter UAV.

changing the constants of the trajectory may easily lead to unfeasible trajectories which cannot be tracked with specified accuracy. The reference tracking accuracy, however, depends also on the present aleatory and epistemic uncertainties.

6.3 EVALUATION METRICS

In order to analyze the simulation results and perform the optimization process, evaluation metrics have to be defined. Predominantly, a metric called Root Mean Square Error (RMSE) is used within this work. It is given by equation (6.5) such that $N_s \in \mathbb{Z}^+$ is a number of evaluation points, $\mathbf{z}$ is the decision variable vector and α is the evaluated variable.

$$\mathcal{J}_\alpha(\mathbf{z}) = \sqrt{\frac{1}{N_s}\sum_{k=0}^{N_s}(\alpha_k(\mathbf{z}) - \tilde{\alpha}_k)^2} \tag{6.5}$$

The following variables in the scope of this work can be used as evaluation metrics either to UAV, robotic arm, or the coupled system:

- **Vehicle position** – Deviation of actual and reference position at time t.
- **Vehicle heading** – Deviation of actual and reference heading at time t.
- **End effector position** – Deviation of manipulator end-effector position at time t.
- **End effector orientation** – Deviation of manipulator end-effector rotation at time t.
- **End effector angular velocity** – Magnitude of end-effector angular velocity at time t.
- **End effector acceleration** – Magnitude of end-effector acceleration at time t.
- **Energy consumed** – Total energy consumed by the vehicle at the end of the mission.

The most intuitive metric to evaluate the grasping maneuver is the end-effector's position as introduced in equation (6.6) and orientation at end time t_f of the simulation. The L previous time steps assume a measurement step δt.

$$\mathcal{J}_1 = \sqrt{\sum_{j=0}^{L}\frac{\|\Delta_p(t_f - j\delta t)\|_2}{L}} + \sqrt{\sum_{j=0}^{L}\frac{\|\Delta_r(t_f - j\delta t)\|_2}{L}} \tag{6.6}$$

However, there are more options regarding evaluation metric modeling. An aerial grasping maneuver, such as the one shown in Figure 6.5, can be observed in natural environment performed by most birds. Studying aerial grasping in nature can help to specify evaluation metrics of the desired grasping maneuver. Apart from precision of the end-effector, the terminal phase of the grasping maneuver typically exhibits a deceleration of the bird to perform a soft contact with its claws, minimizing the force and torque required. Therefore, we consider a second considered evaluation metric defined by equation (6.7), where $\mathbf{a}^B_{C/N} \in \mathbb{R}^3$ is the end-effector acceleration with respect to inertial frame and $\boldsymbol{\omega}^B_{C/N} \in \mathbb{R}^3$ is the end-effector angular velocity.

$$\mathcal{J}_2 = \sqrt{\sum_{j=0}^{L}\frac{\|\mathbf{a}^B_{C/N}(t_f - j\delta t)\|_2}{L}} + \sqrt{\sum_{j=0}^{L}\frac{\|\boldsymbol{\omega}^B_{C/N}(t_f - j\delta t)\|_2}{L}} \tag{6.7}$$

Figure 6.5: Aerial grasping observed in nature. Source: [88]

In a third case, a total energy balance of the maneuver is considered. A total energy consumed by the UAV at time t can be derived using the motor torque $\tau(t) = k_m I(t)$ and the angular speed $\omega(t)$ forming the instantaneous power. The objective function (6.8) is then derived using the mechanical equation of the BLDC motor. In (6.8), J_m represents the motor moment of inertia, ρ is the air density, R_{pr} is the propeller radius, b_m is the viscous friction constant, and C_P is the power coefficient of the propeller.

$$\mathcal{J}_3 = \int_{t_0}^{t_f} \sum_{j=1}^{4} \left(J_m \dot{\omega}_j + \frac{4}{\pi^3} \rho R_{pr}^5 \omega_j^2 C_P + b_m \omega_j \right) \omega_j \, dt \tag{6.8}$$

Alternatively, any combination of these evaluation metrics can be used. This paper includes a weighted sum of all derived evaluation metrics as in equation (6.9) with weights w_1, w_2, and w_3 set to level the scaling of the individual terms. In the case of multi-objective evaluation metric $\mathcal{J}_4$, a pareto optimality is the preferred methodology to consider all the metrics properly.

$$\mathcal{J}_4 = w_1 \mathcal{J}_1 + w_2 \mathcal{J}_2 + w_3 \mathcal{J}_3 \tag{6.9}$$

6.3.1 *Probabilistic evaluation metrics*

In the context of RBCE algorithm outlined in the previous chapter, the following probabilistic requirement given by inequality (6.10) is specified, where $P_{th} \in \mathbb{R}$ is the threshold probability setting the maximum allowed probability of failure. The tolerance $\delta_\alpha \in \mathbb{R}$ is the maximum allowed RMSE deviation of the variable α across the simulation or simulation phase.

$$P\left[\mathcal{J}_\alpha(\mathbf{z}) \geq \delta_\alpha\right] < P_{th} \tag{6.10}$$

The probability to be estimated at the left-hand side of the inequality (6.10) is approximated using the PCE surrogate model approach. We assume the following tolerance and threshold parameters for the defined evaluation metrics as listed in table 6.1.

Table 6.1: Overview of tolerances and probabilistic thresholds.

Metric	Metric description	Tolerance δ_α	Threshold P_{th}
$\mathcal{J}_1$	End-effector accuracy	$3 \cdot 10^{-1}$	$3 \cdot 10^{-2}$
$\mathcal{J}_2$	Grasping softness	$2 \cdot 10^{-2}$	$5 \cdot 10^{-2}$
$\mathcal{J}_3$	Energy consumed	$4 \cdot 10^{1}$	$5 \cdot 10^{-2}$
$\mathcal{J}_4$	Combined metric	$7 \cdot 10^{-1}$	$5 \cdot 10^{-2}$

6.4 MULTIROTOR UNMANNED AERIAL VEHICLE

The following numerical experiments have been conducted assuming a standard quadcopter configuration. The model assumes a Holybro x500 airframe. A simulation using Simscape® allows for the visualization of the respective model as shown in Figure 6.6. Multiple viewpoints are set to observe the vehicle motion.

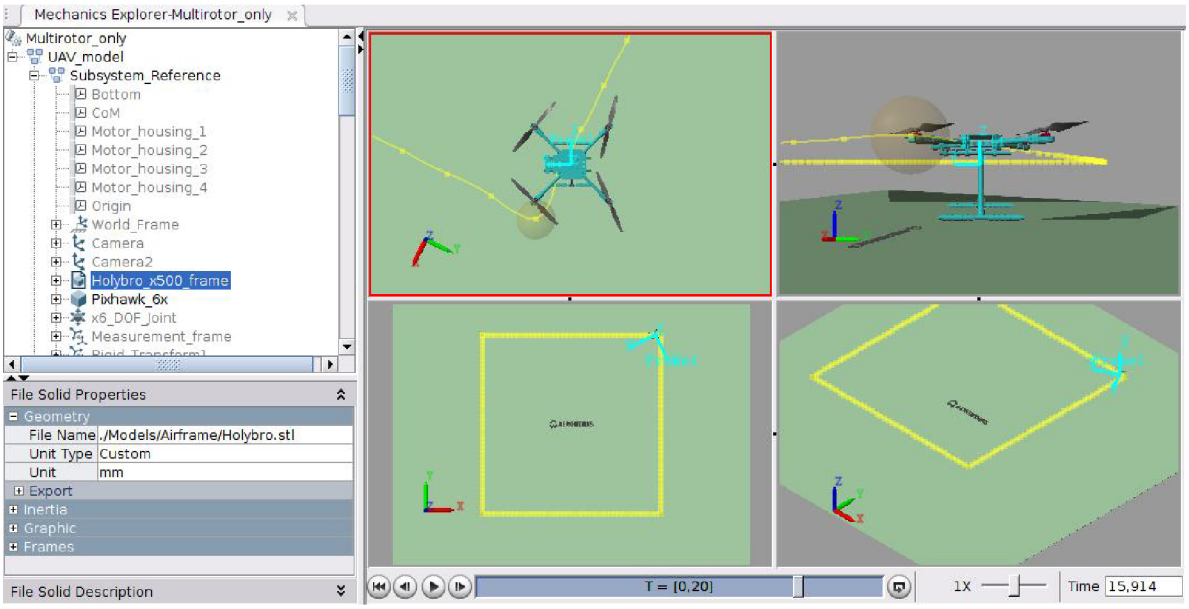

Figure 6.6: Trajectory following using quasi-physical model of a quadcopter.

Let us first introduce the physical parameters of the configuration with respective BLDC motor parameters and propeller parameters. A general set of physical properties of the mathematical model used within the MPC is given in Table 6.2 and includes the vehicle's moments of inertia, arm length, and thrust and power coefficients at nominal hover conditions. Note that the moments of inertia can be computed automatically by Simscape® if the airframe geometry is available.

Table 6.2: Overview of vehicle parameters used in the full nonlinear model.

Parameter	Description	Value	Unit
m	vehicle mass	1.1	kg
I_{xx}	moment of inertia	$3.64 \cdot 10^{-2}$	$kg \cdot m^2$
I_{yy}	moment of inertia	$3.64 \cdot 10^{-2}$	$kg \cdot m^2$
I_{zz}	moment of inertia	$4.86 \cdot 10^{-2}$	$kg \cdot m^2$
l	arm length	$2.00 \cdot 10^{-1}$	m
$C_{T,s}$	thrust coefficient simplified	$1.16 \cdot 10^{-1}$	–
$C_{P,s}$	power coefficient simplified	$3.80 \cdot 10^{-2}$	–

The BLDC motor parameters assumed in our configuration are in Table 6.3. They include the motor moment of inertia, inductance, resistance, electromagnetic force constant, motor torque constant, and viscous friction.

Table 6.3: Overview of BLDC motor parameters.

Parameter	Description	Value	Unit
J_m	motor moment of inertia	$1.25 \cdot 10^{-5}$	$kg \cdot m^2$
L	inductance	$0.5 \cdot 10^{-3}$	H
R	resistance	$0.5 \cdot 10^{-1}$	Ω
k_e	counter electromagnetic force constant	$4.19 \cdot 10^{-3}$	$V \cdot s \cdot 5rad^{-1}$
k_m	motor torque constant	$4.19 \cdot 10^{-3}$	$N \cdot m \cdot A^{-1}$
b_m	viscous friction	0.0	$N \cdot m \cdot s \cdot rad^{-1}$

The chosen propeller for the multirotor configuration was *APC 10x7 Thin Electric* with $N_b = 2$ blades. It has tip radius $R_{pr} = 0.127$ m and hub radius $R_h = 0.01905$ m. Its geometric characteristics along the blades span and the thrust and power coefficients in static conditions with respect to changing Revolutions Per Minute (RPM) are given by Figure 6.7. The mean drag coefficient of the rotor disk is given as $\bar{C}_D = 0.0363$. The mean chord $\bar{c} = 0.0186$ m can be derived. The lift curve slope a can be derived for a range of Reynold number values Re. The dynamic characteristics are given for a range of RPM values and changing free-stream velocity. The obtained characteristics are thus dependent on the advance ratio μ. Figure 6.8 shows the thrust coefficient $C_T(\mu)$ data, torque coefficient $C_Q(\mu)$ data, power coefficient $C_P(\mu) = 2\pi C_Q(\mu)$ data and the derived propulsive efficiency $\eta = \mu \frac{C_T}{C_P}$.

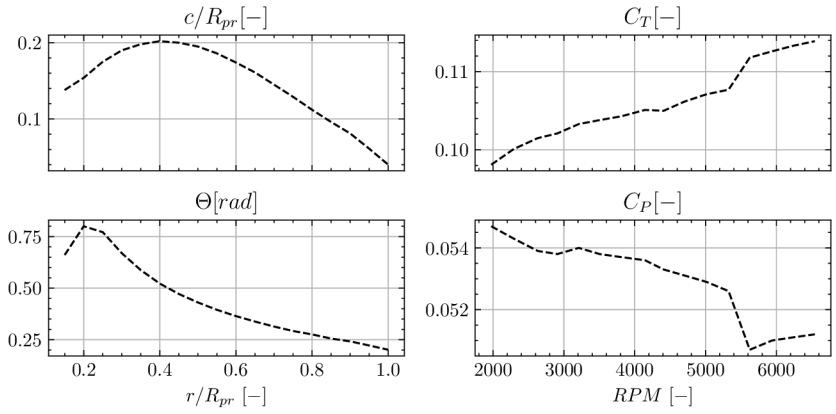

Figure 6.7: Geometric and static parameters of the selected propeller.

Next, the sensor system will be described briefly with the respective sensor noise characteristics introduced. With regards to the IMU model, the following noise power spectral density amplitudes are assumed for the accelerometers and gyroscopes (6.11) respectively.

$$\begin{aligned} \mathbf{N}_a &= \text{diag}([10^{-7} \;\; 10^{-7} \;\; 10^{-7}])\ m \cdot s^{-2} \cdot Hz^{-1} \\ \mathbf{N}_\omega &= \text{diag}([5^{-10} \;\; 5^{-10} \;\; 5^{-10}])\ rad \cdot s^{-1} \cdot Hz^{-1} \end{aligned} \tag{6.11}$$

The measurement of the acceleration $\mathbf{a}_m$ and angular velocity $\boldsymbol{\omega}_m$ in equations (6.12) is computed considering the scaling factors and misalignment matrices $\mathbf{A}_{sf}$ and $\mathbf{G}_{sf}$, the bias

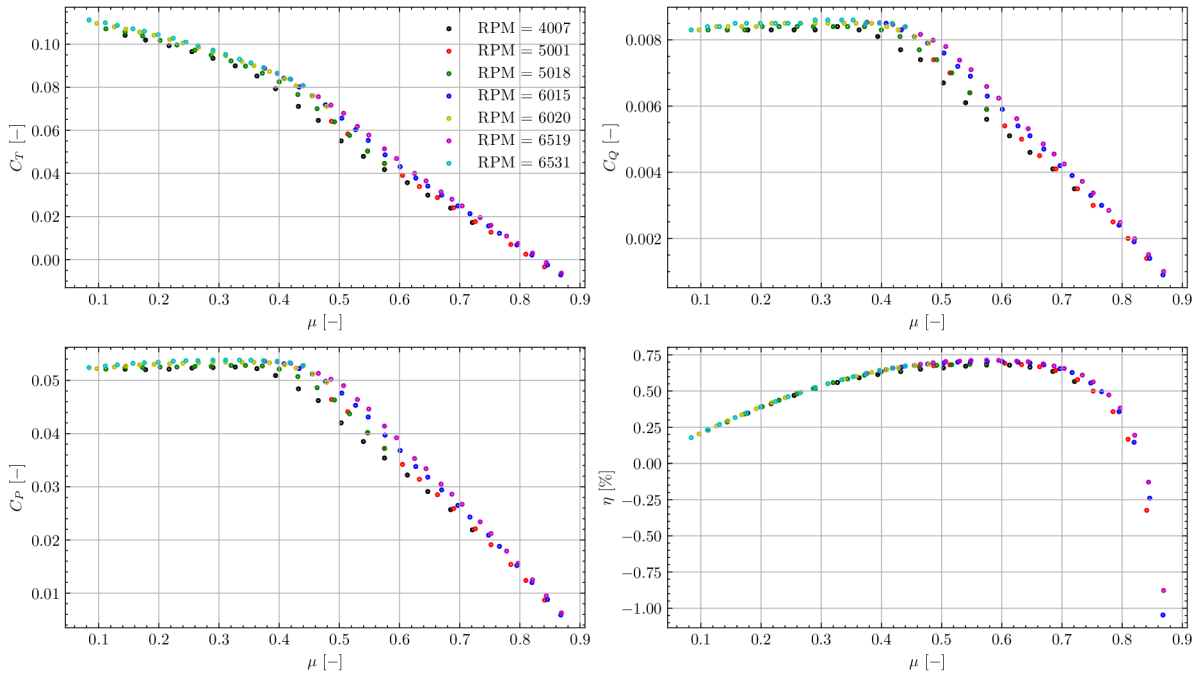

Figure 6.8: Aerodynamic characteristics of the selected propeller.

vectors $\mathbf{a}_b$ and $\mathbf{g}_b$, and the noise vectors $\mathbf{n}_a$ and $\mathbf{n}_\omega$. The true accelerations at CoG are $\mathbf{a}^B_{B/N}$ and angular velocities of the UAV are $\boldsymbol{\omega}^B_{B/N}$.

$$\begin{aligned} \mathbf{a}_m &= \mathbf{A}_{sf}\mathbf{a}^B_{B/N} + \mathbf{a}_b + \mathbf{n}_a \\ \boldsymbol{\omega}_m &= \mathbf{G}_{sf}\boldsymbol{\omega}^B_{B/N} + \mathbf{g}_b + \mathbf{n}_\omega \end{aligned} \tag{6.12}$$

UAV control of simplified mathematical model

The control system of a quadcopter vehicle is introduced by a simple demonstration case showing a minimum working example of developed control system on disturbance-free and measurement noise-free mathematical model matching the model integrated in the MPC system. The system is initialized with a position $\mathbf{p}_0 = [0\ 0\ 2]^T\text{m}$ and target positions $\tilde{\mathbf{p}}_1 = [1\ 1\ 3]^T\text{m}$ in a step change of the reference. The reference point for the quadcopter position is changed after $t = 2$ s.

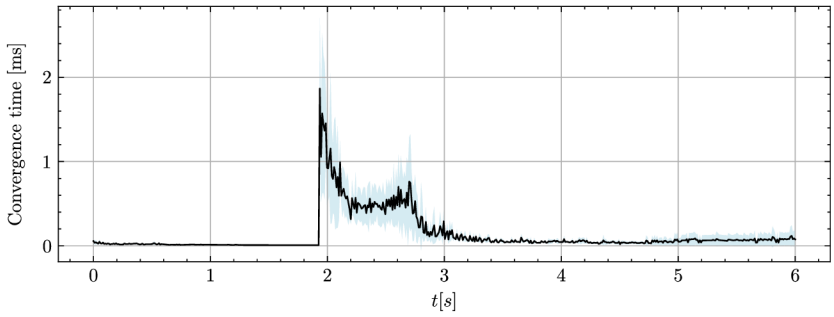

Figure 6.9: Convergence time statistics with respect to varying cost function gains.

The simulation is performed multiple times with varying cost function gains to observe the response and performance of the solver. Figure 6.9 shows a statistics (mean and standard deviation) of solver convergence time across the simulation. It can be seen that the step change of the reference rapidly increases the convergence time which decreases back to its nominal values in two seconds of the maneuver. The dynamic response of the UAV in three scenarios assuming different sets of objective function gains is shown in Figure 6.10. The three trajectories vary by time response characteristics such as overshoot or steady-state error. Two parametrizations also fail to keep the vehicle heading at a nominal value after the reference step change. Manual tuning of the controller gains can be rather inefficient assuming also the uncertainties that can affect the response due to modeling errors or environmental effects.

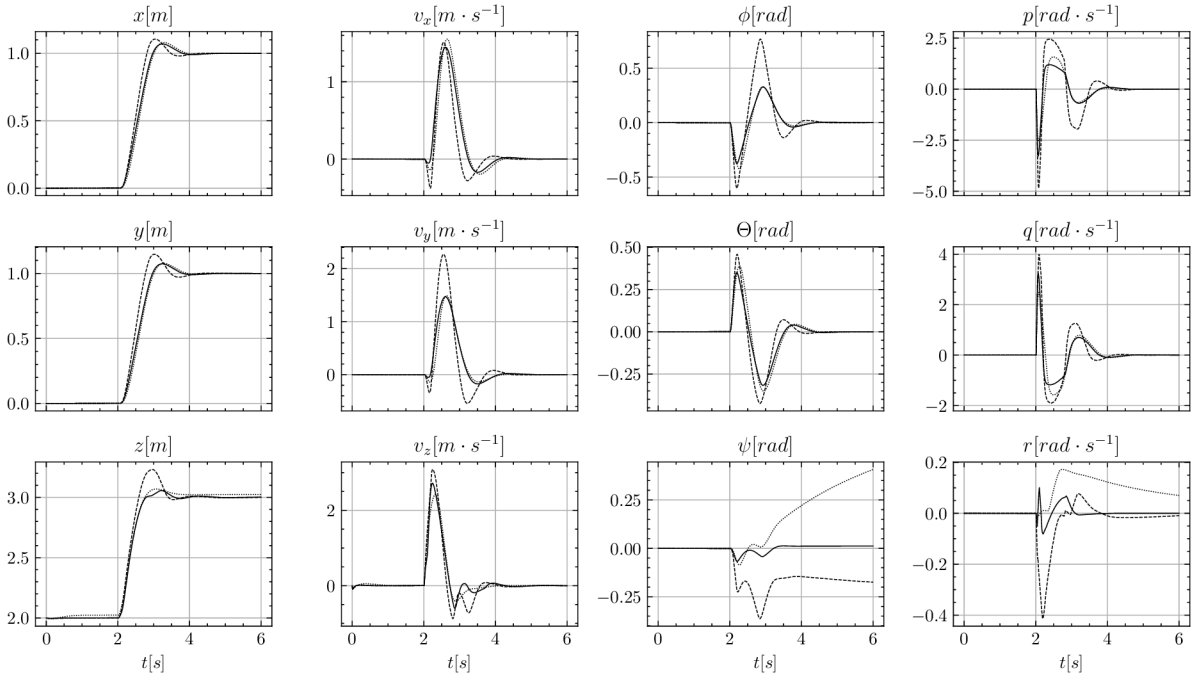

Figure 6.10: Response of the quadcopter dynamics to reference step change.

The control inputs in the form of rotor angular velocities are shown in Figure 6.11. One can see that the lower limit on the control input for MPC is reached during the maneuver. In all cases, however, the angular velocities spike after step change of the reference, abruptly steering the UAV to a new position. To conclude, the MPC controller was successfully tested on the UAV mathematical model identical to the internal model used by the MPC.

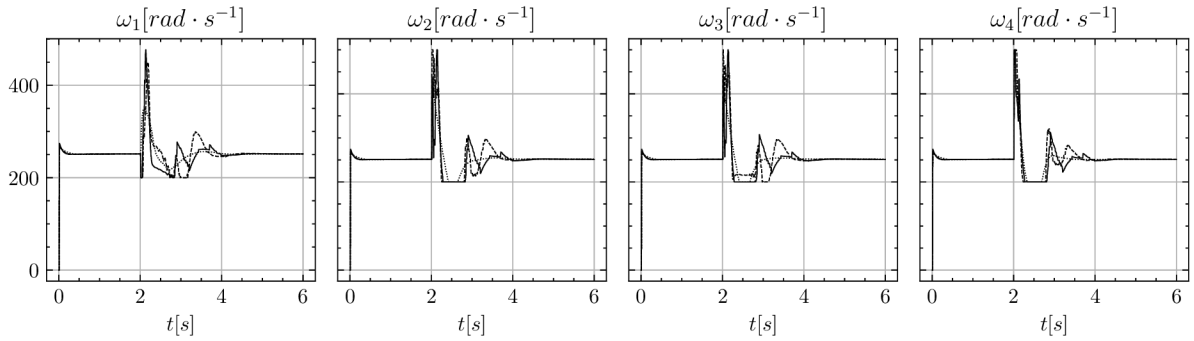

Figure 6.11: Angular velocity control inputs with lower and upper limits.

UAV control of quasi-physical model

Figure 6.12 shows the takeoff and landing of a quasi-physical model of a multirotor UAV with a modeled sensor system introducing respective measurement uncertainty and Von Kármán turbulence model with reference wind speed $W(h) = 2\ \mathrm{m} \cdot \mathrm{s}^{-1}$. It can be seen that the UAV is able to achieve the desired hover position and heading and land safely. In this case, the MPC controller is synchronized with the IMU and runs at 250 Hz. This update rate can be achieved in real-time on a personal computer with a solver warm-starting using the previous solution but not onboard a real vehicle. Therefore, different refresh rates of the MPC and

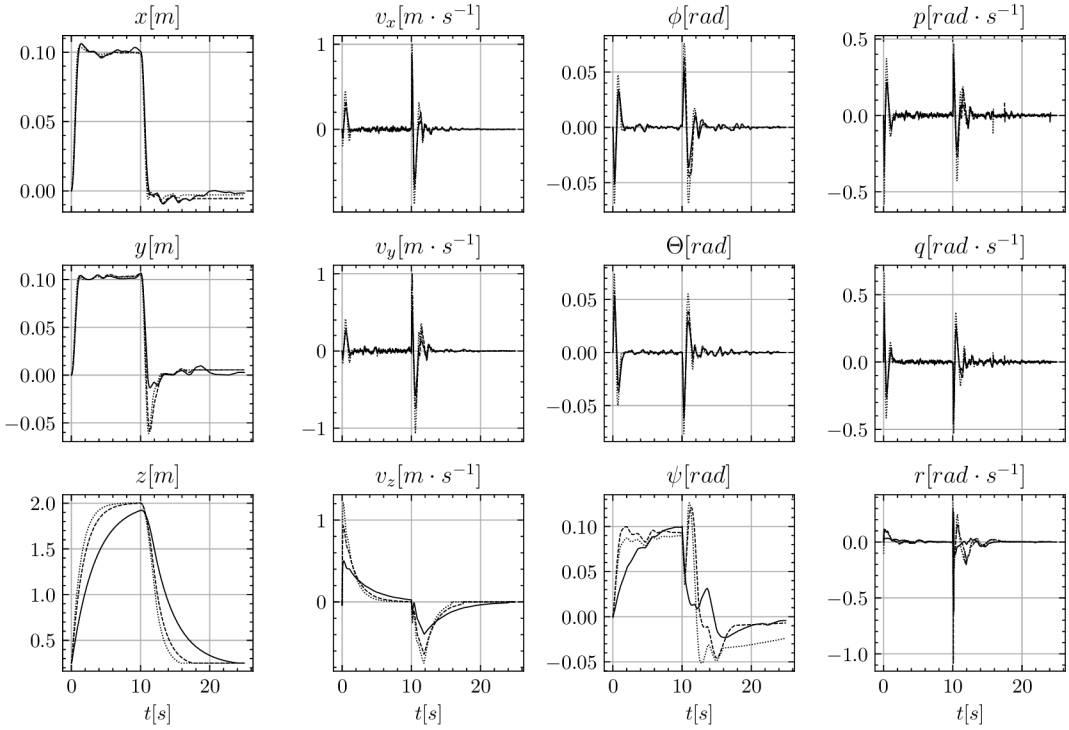

Figure 6.12: Quasi-physical model control using different weighting matrices.

the corresponding effect on achieving the desired state were studied to find a minimum required update rate of the MPC to consider integration onboard a real vehicle. Figure 6.13 shows that around 50 Hz of the update frequency, the reference tracking error spikes as the UAV struggles to stabilize.

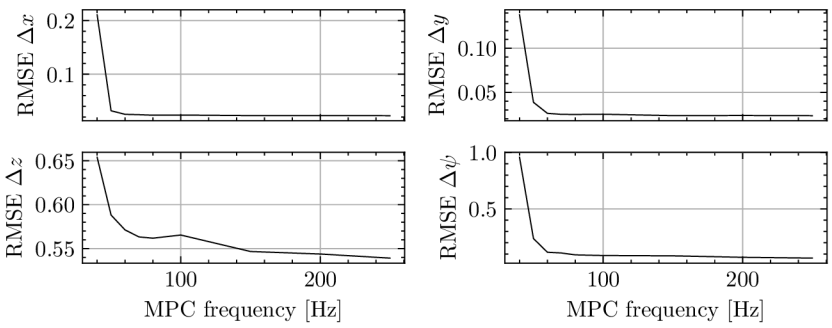

Figure 6.13: Position and heading RMSE with respect to changing MPC update rate.

Adaptive control loop evaluation and performance testing

In this section, the $\mathcal{L}_1$ adaptive augmentation loop is tested on the defined testing trajectories under the influence of uncertainties with regards to a UAV's mass and moments of inertia in order to emulate the conditions arising from the coupling effects of a UAV equipped with a robotic manipulator. Initially, the behavior of $\mathcal{L}_1$ adaptive augmentation loop is observed when changing mass difference Δm and inertia difference $\Delta \mathbf{I}_b$. The following manually-set parameters of the adaptation gain matrix and cut-off frequencies as in equation (6.13) were used in all the simulations in this section.

$$\begin{aligned} \mathbf{A}_s &= \text{diag}([-3.5 \ -3.5 \ -3.5 \ -3.5 \ -3.5 \ -3.5]) \\ \boldsymbol{\omega}_{co} &= [1.0 \ 1.0 \ 1.0 \ 1.0]^T \end{aligned} \tag{6.13}$$

Figure 6.14 shows the augmented control inputs computed by the $\mathcal{L}_1$ adaptive augmentation loop with varying additional mass which is not considered in the model propagated by the MPC.

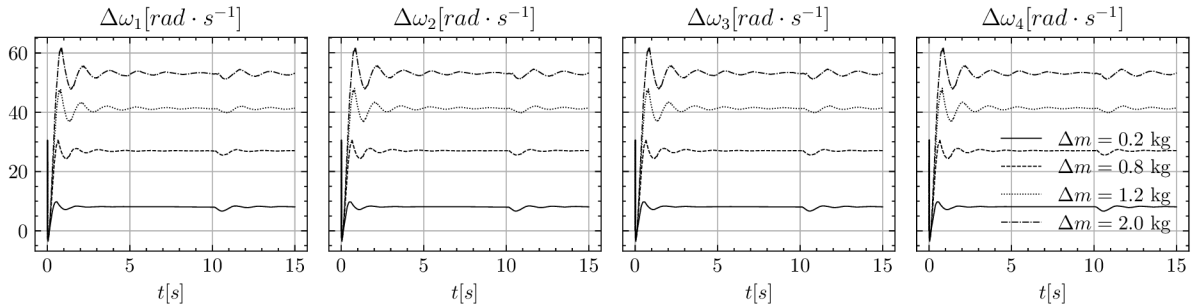

Figure 6.14: Adaptive augmentation loop inputs based on mass errors.

Following the same methodology, error in the moment of inertia matrix is introduced to the system and the $\mathcal{L}_1$ adaptive augmentation loop is activated to compensate for the incorrectly set moments of inertia propagating in the MPC. Figure 6.15 shows the compensation input of $\mathcal{L}_1$ under varying errors in moments of inertia. Increasing the error term further ultimately leads to instability even with the augmentation loop activated.

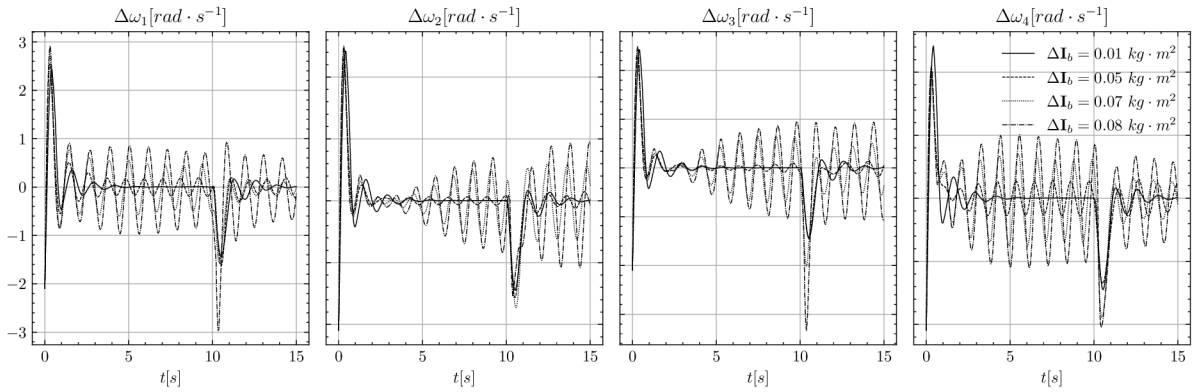

Figure 6.15: Adaptive augmentation loop inputs based on inertia errors.

Let us consider a hover change maneuver driven by either MPC or MPC augmented by $\mathcal{L}_1$ adaptive loop. Introducing mass and moment of inertia errors and observing the response characteristics and evaluation metrics can help to design a robust controller. Using the manually tuned controllers, the PCE-Kriging surrogate models are constructed for both cases evaluating the position of RMSE and setting the position error limit. The mass and inertia uncertainties are modeled to have Gaussian and uniform distribution as $\Delta m \sim \mathcal{N}(0.0, 0.1)$ and $\Delta I \sim \mathcal{U}(-0.01, 0.12)$. Figure 6.16 shows a failure boundary region with respect to RMSE position error tolerance that emerges as a result of mass and inertia error of nominal MPC internal model and the simulation model. In six iterations, the learning function selects enough additional design points to correctly shape the failure boundary region.

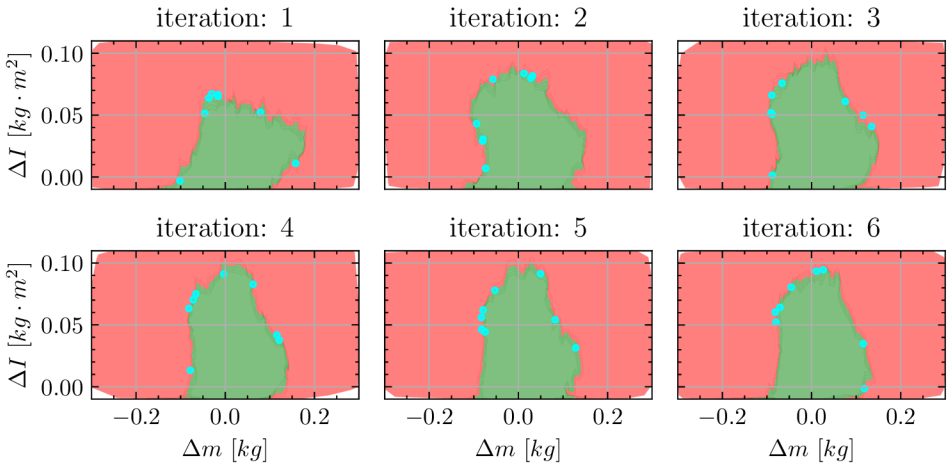

Figure 6.16: Nominal MPC evaluation using PCE-Kriging surrogate model.

Figure 6.17 shows a convergence of failure boundary on hover change scenarios driven by MPC with $\mathcal{L}_1$. One can see that, compared to nominal MPC, the region satisfying a specified RMSE limit extends, thus improving the robustness of the controller to the studied uncertainties. The surrogate models are constructed by sampling 20 design points by LHS to construct

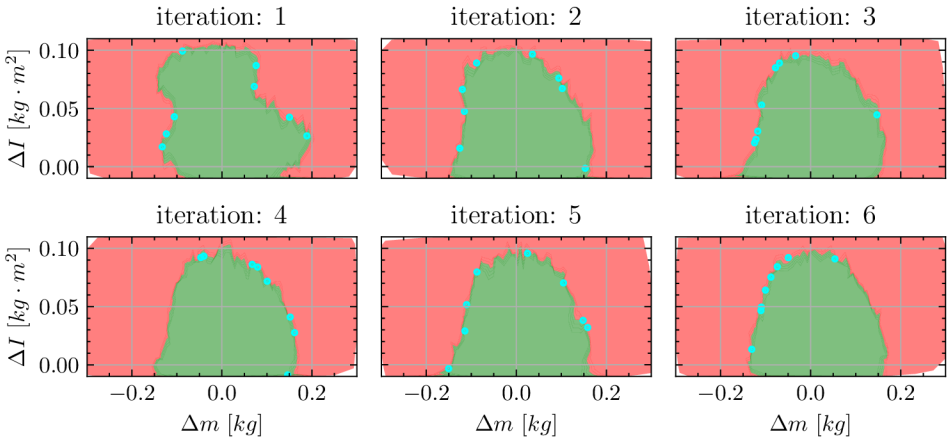

Figure 6.17: MPC with $\mathcal{L}_1$ evaluation using PCE-Kriging surrogate model.

the PCE model and consequently adding 8 simulations in each iteration of the failure boundary refinement process.

The following tables use the same technique to evaluate the controllers on a set of different flight trajectories computing the mean RMSE and the failure probability with respect to the probabilistic space. The UAV position error at time t is shown in Figure 6.18.

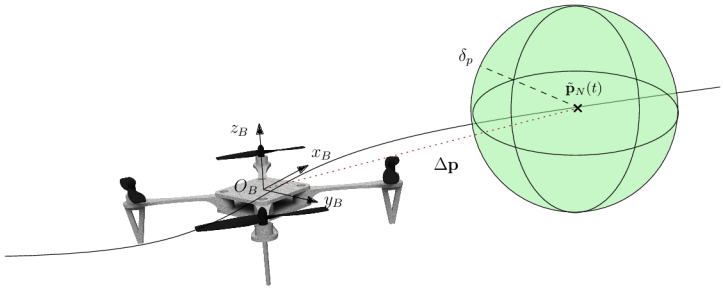

Figure 6.18: Position deviation with respect to time-varying reference.

Simultaneously, the results are compared with the same evaluations performed using the quasi-physical, which extends the mathematical model with accurate propeller aerodynamic, sensor, and actuator models. Thus, an overall lower robustness is expected at the quasi-physical model level since more uncertainties have an impact on the overall performance. The settings for constructing the PCE-Kriging surrogate models are shown in Table 6.4.

Table 6.4: Surrogate model settings used throughout the thesis.

Parameter of PCE	Symbol	Value	Parameter of Kriging	Symbol	Value
Number of samples	N^{PCE}	20	Meta-model samples	N^{KRIG}	1000
Max. polynomial order	p	4	Resampling iterations	M^{KRIG}	6
Failure eval. samples	N^{P}	1000	Resampling points	P^{KRIG}	8

The analysis of a mathematical model was initially performed. Table 6.5 shows the evaluation results of the nominal MPC in a number of considered scenarios following the defined trajectories. The respective failure rates highly depend on the trajectory parameters. Therefore, the trajectory parameters are kept constant when evaluating different controller settings and model types. The term failure does not by any means denote the crash of the UAV, but simply exceeding a given threshold of RMSE tracking error with a maximum allowed failure of 5 %.

Table 6.5: Evaluation of a mathematical model driven by nominal MPC.

Scenario	μ RMSE [m]	σ RMSE [m]	P_{fail} [%]	Passed P_{th}
Hover	$1.262 \cdot 10^{-1}$	$5.799 \cdot 10^{-3}$	1.612	True
Spiral	$2.798 \cdot 10^{-1}$	$1.613 \cdot 10^{-3}$	7.474	False
Lemniscate	$1.670 \cdot 10^{-1}$	$3.949 \cdot 10^{-3}$	4.825	True
Diamond	$1.195 \cdot 10^{-1}$	$5.737 \cdot 10^{-3}$	1.024	True
Lobe	$1.986 \cdot 10^{-1}$	$2.317 \cdot 10^{-3}$	6.612	False

Table 6.6 shows the evaluation of the nominal MPC augmented by $\mathcal{L}_1$ adaptive controller with respect to the considered scenarios. In all the studied cases, the $\mathcal{L}_1$ augmentation loop improved the overall error statistics and decreased the expected failure rate.

Table 6.6: Evaluation of a mathematical model using MPC and L1.

Scenario	μ RMSE [m]	σ RMSE [m]	P_{fail} [%]	Passed P_{th}
Hover	$9.404 \cdot 10^{-2}$	$2.155 \cdot 10^{-3}$	0.854	True
Spiral	$2.152 \cdot 10^{-1}$	$1.299 \cdot 10^{-3}$	6.877	False
Lemniscate	$1.364 \cdot 10^{-1}$	$1.768 \cdot 10^{-3}$	0.983	True
Diamond	$8.489 \cdot 10^{-2}$	$2.735 \cdot 10^{-3}$	0.711	True
Lobe	$1.581 \cdot 10^{-1}$	$1.160 \cdot 10^{-3}$	3.791	True

Similarly, the analysis of all the scenarios is performed using the quasi-physical model. Table 6.7 shows an evaluation of a quasi-physical model driven by the nominal MPC in all considered scenarios. Due to additional uncertainties emerging in the quasi-physical model, the failure rate slightly increased compared to the failure rate evaluation using a mathematical model.

Table 6.7: Evaluation of a quasi-physical model driven by nominal MPC.

Scenario	μ RMSE [m]	σ RMSE [m]	P_{fail} [%]	Passed P_{th}
Hover	$1.422 \cdot 10^{-1}$	$4.895 \cdot 10^{-3}$	1.933	True
Spiral	$2.914 \cdot 10^{-1}$	$1.877 \cdot 10^{-3}$	8.177	False
Lemniscate	$1.909 \cdot 10^{-1}$	$3.287 \cdot 10^{-3}$	5.134	False
Diamond	$1.313 \cdot 10^{-1}$	$5.185 \cdot 10^{-3}$	4.148	True
Lobe	$2.240 \cdot 10^{-1}$	$3.899 \cdot 10^{-3}$	7.052	False

Table 6.8 shows an evaluation of a quasi-physical model driven by the nominal MPC augmented by $\mathcal{L}_1$ adaptive controller with respect to the considered scenarios. Again, the $\mathcal{L}_1$ augmentation loop significantly lowers the failure probability compared to nominal MPC.

Table 6.8: Evaluation of a quasi-physical model using MPC and L1.

Scenario	μ RMSE [m]	σ RMSE [m]	P_{fail} [%]	Passed P_{th}
Hover	$1.047 \cdot 10^{-1}$	$2.418 \cdot 10^{-3}$	1.057	True
Spiral	$2.266 \cdot 10^{-1}$	$1.217 \cdot 10^{-3}$	5.847	False
Lemniscate	$1.809 \cdot 10^{-1}$	$2.274 \cdot 10^{-3}$	4.437	True
Diamond	$9.752 \cdot 10^{-2}$	$2.528 \cdot 10^{-3}$	0.896	True
Lobe	$2.180 \cdot 10^{-1}$	$6.223 \cdot 10^{-3}$	6.952	False

Without automatic tuning, the surrogate modeling technique was used to evaluate failure probability of a dedicated UAV on a range of intricate trajectories considering position RMSE across the simulation as the primary evaluation factor.

Weight Optimization using Cross Entropy Method

In order to improve the overall performance of the control mechanism, CEM is employed to optimize the weighting parameters of the MPC. The spiral trajectory was chosen for this example. Several parameters regarding the CEM solver and MPC such as a number of prediction samples and the sampling interval were fixed and are shown in Table 6.9.

Table 6.9: Fixed parameter settings of the CEM and MPC algorithms.

Parameter of CEM	Symbol	Value	Parameter of MPC	Symbol	Value
Iteration sample size	M^{CE}	40	Solver tolerance	ϵ_s	$1e^{-5}$
Elite sample size	N^{CE}	8	Prediction samples	K	20
Smoothing factor	α^{CE}	0.8	Maximum duration	t_{max}	3 ms
Stopping criterion	ϵ^{CE}	$1e^{-5}$	Penalty weight	γ	20
Maximum iterations	K^{CE}	20	Sampling interval	δt	0.05

For demonstration purpose, Figure 6.19 shows two scenarios of spiral trajectory tracking, where the trajectory on the left was generated using the initial manually-set weighting parameters, while the trajectory on the right was generated by using the best performing set of weights computed using 20 iterations of CEM on the mathematical model.

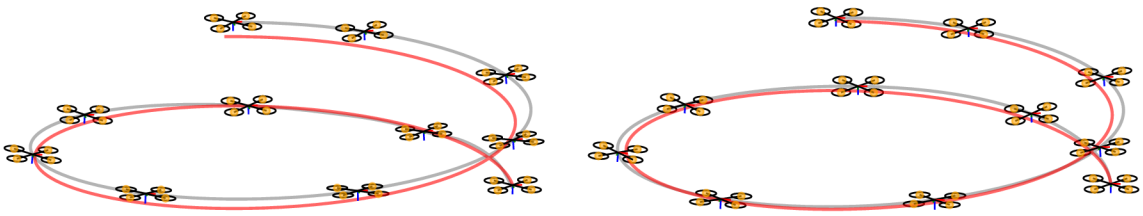

Figure 6.19: Spiral trajectory tracking optimization.

The CEM weight optimization was utilized for both mathematical and quasi-physical models to observe the effect of discrepancies on the achieved evaluation metric. Figure 6.20 shows a comparison of RMSE convergence during the optimization process. It can be observed that the optimized solution yields higher RMSE for the quasi-physical model compared to a mathematical model.

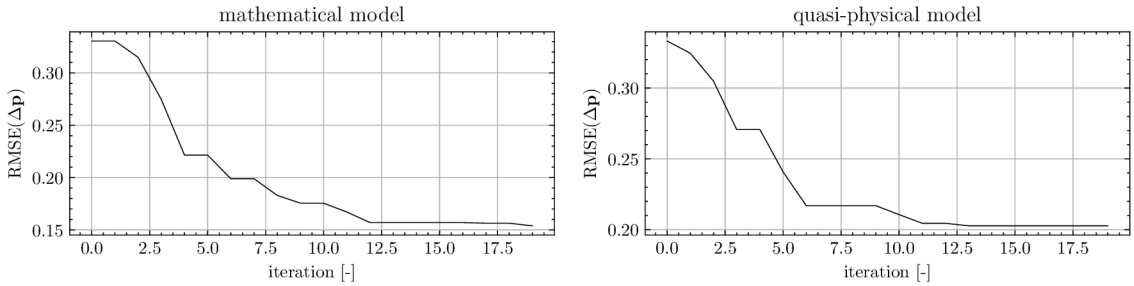

Figure 6.20: Optimizing MPC parameters for two types of models.

Probabilistic Weight Optimization using Reliability-based Cross Entropy

Compared to standard CEM, the RBCE method switches to the failure probability minimization task with respect to the defined uncertainties after a fixed number of iterations. The surrogate model settings with CEM settings were combined while specifying the switching iteration $N^{RB} = 15$. Figure 6.21 shows the refined failure boundary obtained by 6 resampling iterations of the optimized MPC. The last 5 of 20 CEM iterations were focused on reducing the failure probability given the model of uncertainties presented earlier. It can be seen that by optimizing the controller, the safe operational envelope given the uncertainties increased substantially compared to the nominal design tuned manually. Table 6.10 shows the result-

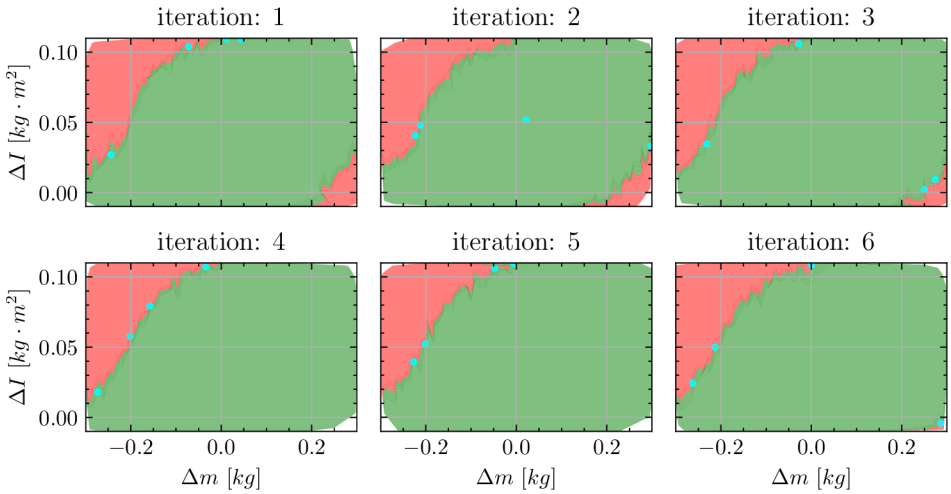

Figure 6.21: Optimized MPC evaluation using PCE kriging surrogate model.

ing failure probabilities evaluated for the mathematical and quasi-physical model using the optimized MPC augmented by $\mathcal{L}_1$. For both models, the failure rate decreased to a satisfactory value after the optimization finished. To conclude this set of examples, the use of

Table 6.10: Evaluation of failure probability for the spiral trajectory.

Model	μ RMSE [m]	σ RMSE [m]	P_{fail} [%]	Passed P_{th}
Mathematical	$1.627 \cdot 10^{-1}$	$5.533 \cdot 10^{-3}$	3.741	True
Quasi-physical	$2.026 \cdot 10^{-1}$	$4.257 \cdot 10^{-3}$	4.862	True

PCE-Kriging surrogate model for failure probability estimation was shown, adaptive control loop performance testing on a range of analytically specified trajectories was performed, and an optimization of MPC weighting parameters was shown to substantially increase the performance of the controller. The methodology will later be used on the coupled system of UAV with an integrated robotic arm while considering the stochastic ODD and the specified evaluation metrics.

6.5 SERIAL ROBOTIC MANIPULATOR

Following the methodology applied for the multirotor UAV numerical experiments section, this section is dedicated to serial manipulator numerical testing. The simulation of a robotic arm with six movable joints is composed using Simscape®. Figure 6.22 shows a robotic arm performing box picking driven by baseline MPC controller of the arm. The MPC has been developed using a kinematic mathematical model and then applied on the quasi-physical model which has inner inverse dynamics control loop that provide torques to counteract joint motions that would result by acting gravitational acceleration.

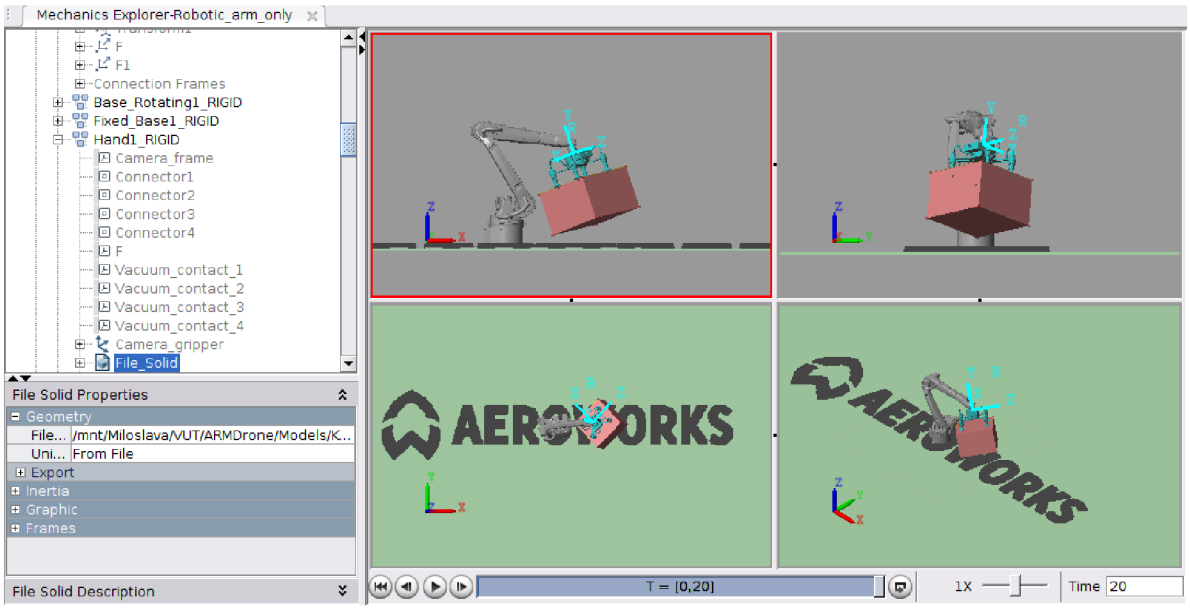

Figure 6.22: Box pick and place simulation of manipulator quasi-physical model.

The geometry and physical parameters of each robotic arm joint are now specified. The kinematic chain specifying the serial robot of choice featuring six rotational joints is described by a set of DH parameters, which are listed in Table 6.11 for each of the links of the robot along with the mass of the associated arm part.

Table 6.11: DH parameters of the implemented serial robot and mass of each part.

-	$l_i[m]$	$d_i[m]$	$\alpha_i[rad]$	$\Theta_i^0[rad]$	$m_i[kg]$
Link 1	$6.69 \cdot 10^{-2}$	$1.67 \cdot 10^{-1}$	$\pi/2$	0.00	0.10
Link 2	$3.18 \cdot 10^{-1}$	0.00	0.00	$\pi/2$	0.12
Link 3	$7.87 \cdot 10^{-3}$	0.00	$\pi/2$	0.00	0.08
Link 4	0.00	$3.38 \cdot 10^{-1}$	$\pi/2$	0.00	0.05
Link 5	0.00	0.00	$-\pi/2$	$\pi/2$	0.03
Link 6	0.00	$3.93 \cdot 10^{-2}$	0.00	0.00	0.02

The quasi-physical model has joints subject to rotational friction featuring parameters of viscous friction coefficient C_{vf}, coulomb friction torque τ_{cf}, breakaway friction torque τ_{brk}, and breakaway friction velocity ω_{brk}. Table 6.12 shows all the friction parameters for each

joint with lower and upper bounds on the joint angular position. These bounds can be used as state constraints in the MPC formulation.

Table 6.12: Rotational friction and rotational limit parameters of the joints.

-	$C_{vf}[-]$	$\tau_{brk}[N \cdot m]$	$\tau_{cf}[N \cdot m]$	$\omega_{brk}[rad/s]$	$\Theta_{lb}[deg]$	$\Theta_{ub}[deg]$
Link 1	$1.00 \cdot 10^{-3}$	$6.50 \cdot 10^{-2}$	$6.00 \cdot 10^{-2}$	$1.00 \cdot 10^{-2}$	−170	170
Link 2	$1.00 \cdot 10^{-3}$	$1.60 \cdot 10^{-2}$	$1.50 \cdot 10^{-2}$	$1.00 \cdot 10^{-2}$	−185	65
Link 3	$1.00 \cdot 10^{-3}$	$3.00 \cdot 10^{-2}$	$2.40 \cdot 10^{-2}$	$1.00 \cdot 10^{-2}$	−137	163
Link 4	$1.00 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	$5.00 \cdot 10^{-4}$	$1.00 \cdot 10^{-2}$	−185	185
Link 5	$1.00 \cdot 10^{-3}$	$2.50 \cdot 10^{-4}$	$2.00 \cdot 10^{-4}$	$1.00 \cdot 10^{-2}$	−350	350
Link 6	$1.00 \cdot 10^{-3}$	$2.50 \cdot 10^{-4}$	$2.00 \cdot 10^{-4}$	$1.00 \cdot 10^{-2}$	−120	120

Serial manipulator control of mathematical model

The control system will be introduced by a simple demonstration case showing a minimum working example of the developed control system on a disturbance-free and measurement noise-free mathematical model matching the model integrated in the MPC system. The reference target position $\tilde{\mathbf{p}}_1 = [0.48\ 0.35\ 0.25]^T m$ and target rotation $\tilde{\mathbf{q}}_1 = [0.0\ 1.0\ 0.0\ 0.0]^T$ is processed by the optimizer to optimally drive the arm joints, such that the reference is achieved. There are two possibilities that have an effect on the controller ability to correctly drive the end-effector to its reference pose. The OCP can be composed such that the reference is included in the objective function and is minimized along the path, or it is forced by imposing a constraint at the end of the prediction horizon. In the following example, the reference is achieved by augmenting the objective function along the path by end-effector position and orientation cost. However, a statistical evaluation of solver convergence assumes cases with active position constraint or orientation constraint at the end of the prediction horizon. Figure 6.23 shows the end-effector position trajectories) with regards to the reference (dashed line). All three considered MPC parametrizations resulted in convergence of the end-effector position to the referenced location.

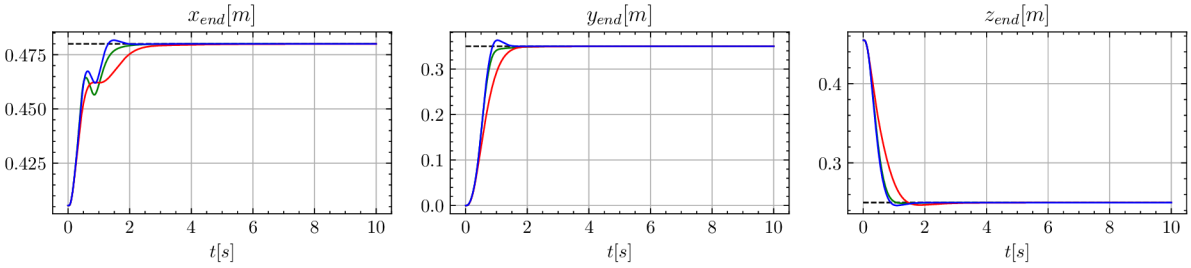

Figure 6.23: End-effector position control with reference target values.

The tuning of the control loop assuming the objective parameters or solver parameters including the tolerance or maximum duration may however result in a better performance

of the controller. The optimization of the arm weighting parameters by CEM or RBCE will be assumed in case of coupled dynamics of UAV and robotic arm.

The end-effector orientation is shown shown in Figure 6.24 in a quaternion form. The orientation initially deviates from the reference and continually converges back. Once again, the overshoot suggests that the controller might need some tuning of parameters, bounds, or solver settings to improve the controller performance.

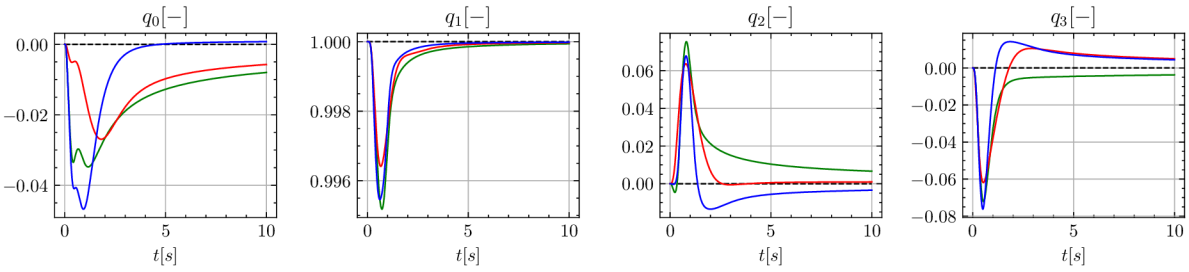

Figure 6.24: End-effector orientation reference tracking.

A further analysis of the solver performance with regards to solver convergence time across varying prediction horizons and applying end-effector position constraint at end of the horizon (A1) or end-effector orientation constraint at end of the horizon (A2). As shown in Figure 6.25, applying position or orientation constraint results in similar increase in computational time required. The computational time assuming a maximum of $K = 20$ prediction steps does not exceed 10 ms with or without constraints indicating possibility to operate all controller configurations at up to 100 Hz.

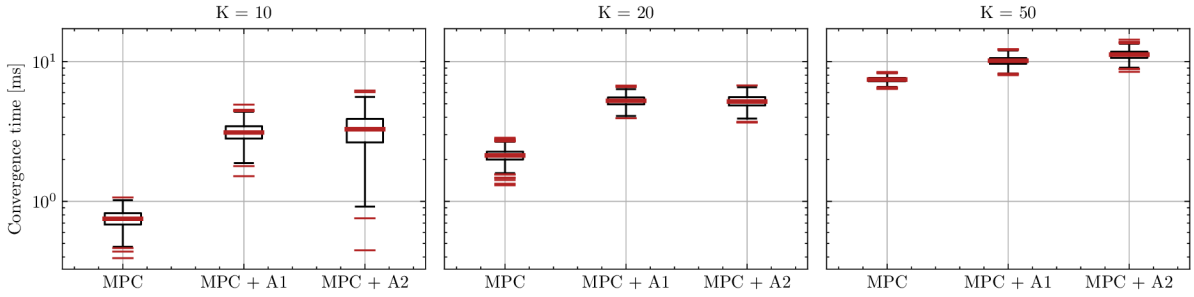

Figure 6.25: Statistical evaluation of solver convergence on robotic arm MPC.

Figure 6.26 shows angular acceleration commands for the manipulator joints in the assumed demonstration cases. The sampling time of the controller was set to $\delta t = 0.05$ s. The control input can also be limited to prevent aggressive motion by tuning the lower and upper bounds of the control inputs. The plots show that the control bounds have been reached in several cases. The acceleration converge to zero when the transition to a new pose is achieved.

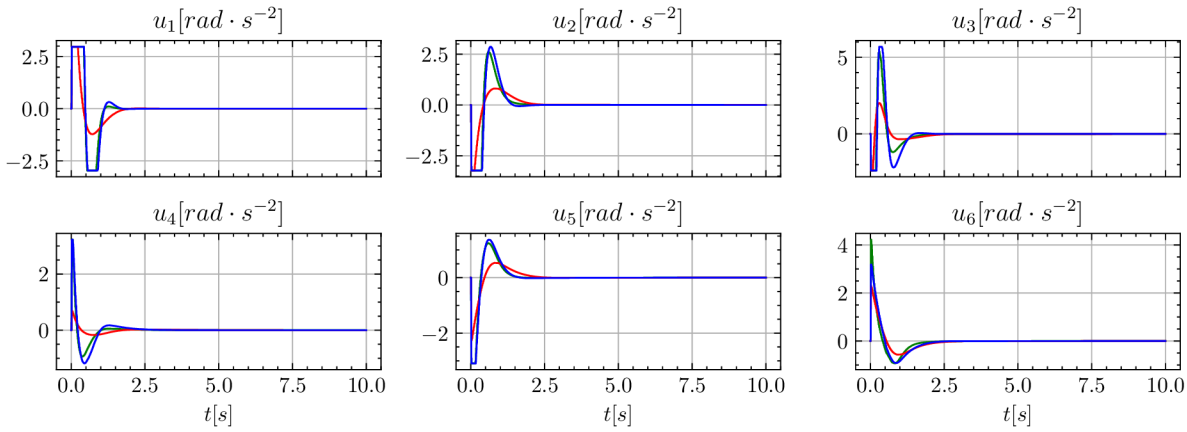

Figure 6.26: Joint angle acceleration control inputs generated by MPC.

Figure 6.27 shows the change of manipulator joint angle positions and the respective angular velocities during the demonstration simulations. The effect of different weighting parameters can be clearly seen by observing the joint angular velocities. For example, the angular velocity weighting factors were set lowest for the blue-colored trajectory resulting in highest joint velocities. All joint angular velocities ultimately converge to zero as the end-effector reaches the reference position and orientation.

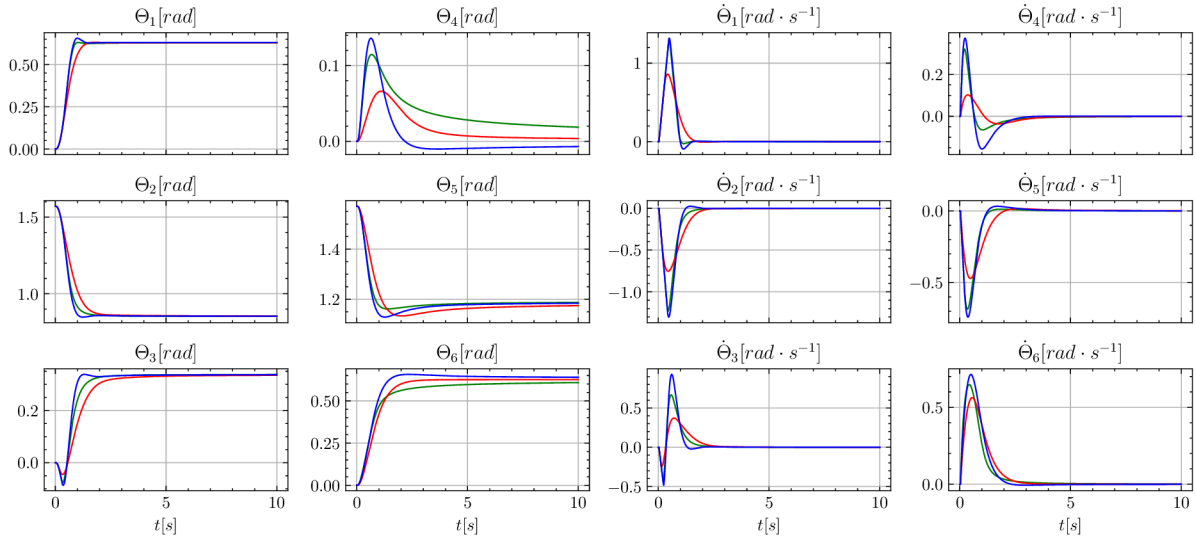

Figure 6.27: Joint angle position and angular velocities in demonstration case.

This section has demonstrated the developed MPC for six-joint robotic arm. The next section focuses on merging the two control loops into a single coupled MPC that computes the control action for UAV and the manipulator simultaneously within a single optimization problem that would allow coordinated motion allowing for complex aerial grasping maneuvers.

6.6 COUPLED SYSTEM FOR AERIAL GRASPING

The complexities arising in formulating and solving the coupled problem for aerial grasping have several important aspects to be considered. First, the computational complexity may render the controller unfeasible with regards to the required update rate. Forcing a stopping criterion based on computation time may then produce a sub-optimal or even unstable behavior of the MPC. Secondly, the motion reference is set to both UAV CoG and end-effector position and rotation. This can cause conflicting objectives resulting in a sub-optimal behavior of the system. Weighting the motion of each subsystem such that the system behaves as expected, therefore, poses a challenge. As an example of a varying behavior of the controller based on parameter setting is shown in Figure 6.28. Some of the simulations are weighted to prioritize the minimization of the end-effector position and orientation error. As a result, the UAV performs an aggressive maneuver in order to minimize the cost function as quickly as possible. On the other hand, some simulations have been weighted such that the UAV follows a continuous trajectory without a sharp change of attitude and joint angles. It can be observed that the OCP does not consider collision constraints with the UAV platform. The bottom right trajectory shows a collision of the end-effector with the UAV when performing the maneuver. This shows the importance of setting the collision cone constraints on the end-effector position relative to the UAV BFF.

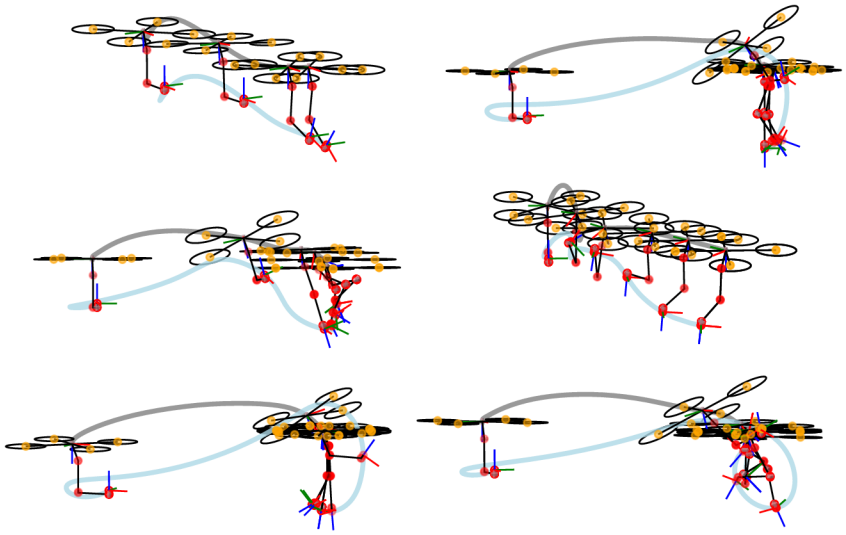

Figure 6.28: Effect of coupled problem objective weighting on the simulation result.

The selected simulations also show the diversity of the motion with the same structure of OCP considered. The weighting matrices seem to be essential for shaping the maneuver according to the mission needs. Initially, the cost function weighting was manually set in a sub-optimal way. In addition, a computational time limit was set to 3 ms, which substantially reduced the controller robustness as the solution did not converge well enough to control the coupled system. The motion of the arm is highly chaotic in some of the simulations with significant overshoot with respect to the target.

A set of simulations using varying objective weight setting has shown that the solution quality issue persists in many possible cases. As Figure 6.29 shows, a reference position of the UAV is achieved, but the transition shows oscillatory behavior, which is not satisfactory for the aerial grasping task. Furthermore, the heading of the vehicle changes very rapidly in the first few seconds of the simulation, thus rendering the yaw motion of the UAV unstable.

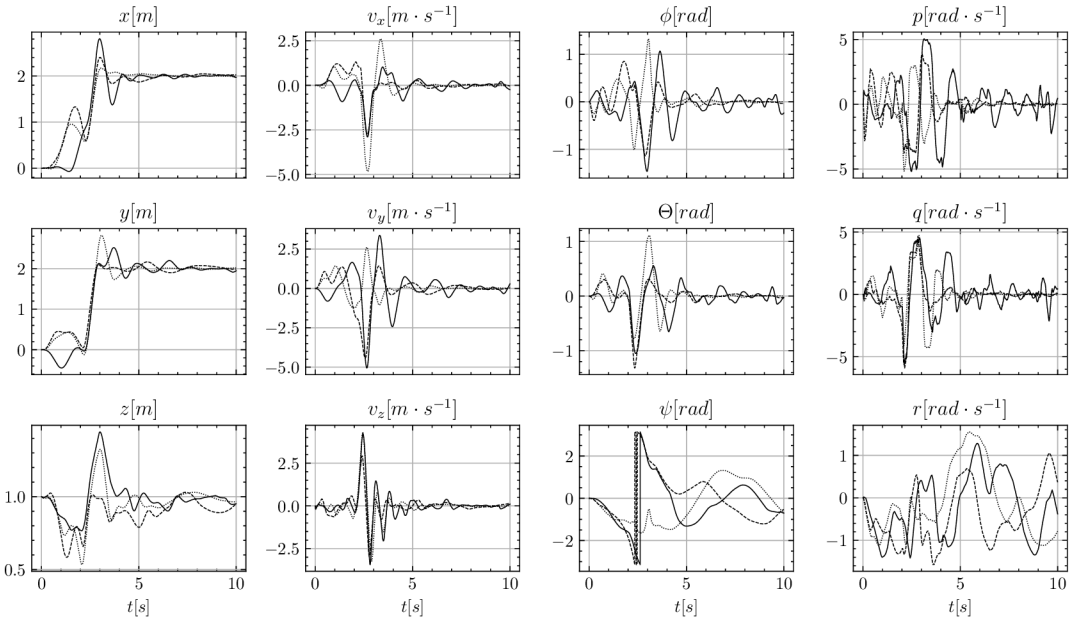

Figure 6.29: Quadcopter state evolution using initial settings.

A similar phenomenon is observed at joint angle motion in Figure 6.30, where the joint velocities oscillate in all tested cases. The joint angles diverge as the result of poor weighting of the manipulator joint velocities and accelerations. No bounds were imposed on the joint angles since the problem was formulated as a single shooting problem integrating the dynamics forward in the objective function. In such case, the states are not parameters of the optimization

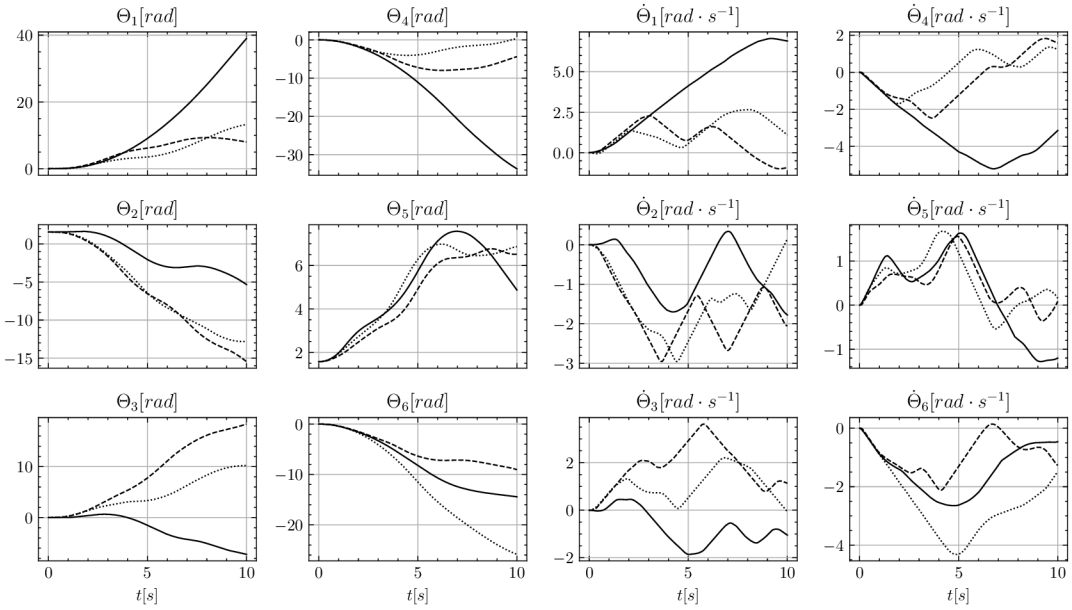

Figure 6.30: Joint angle position and angular velocities in demonstration case.

problem and the problem is typically simpler to compute. The UAV control inputs applied in the initial scenarios are shown in Figure 6.31. It can be seen that the lower limit on angular velocities is frequently hit as the solution oscillates due to convergence issues.

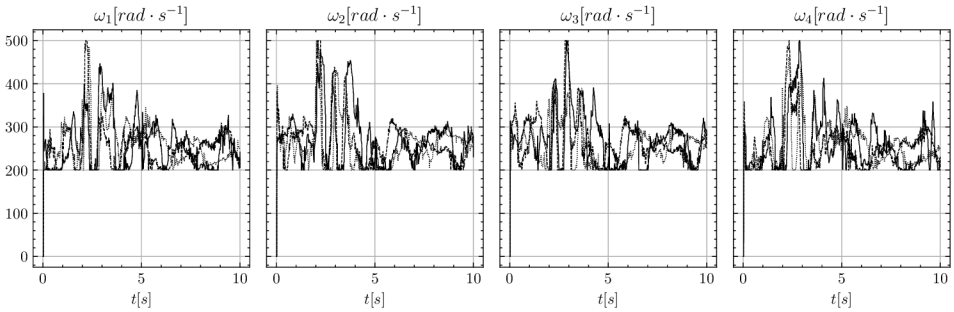

Figure 6.31: Rotor angular velocities in initial testing of coupled MPC.

The same applies to manipulator control inputs, which oscillate while frequently reaching joint acceleration limits, which are enforced by the OCP as shown in Figure 6.32.

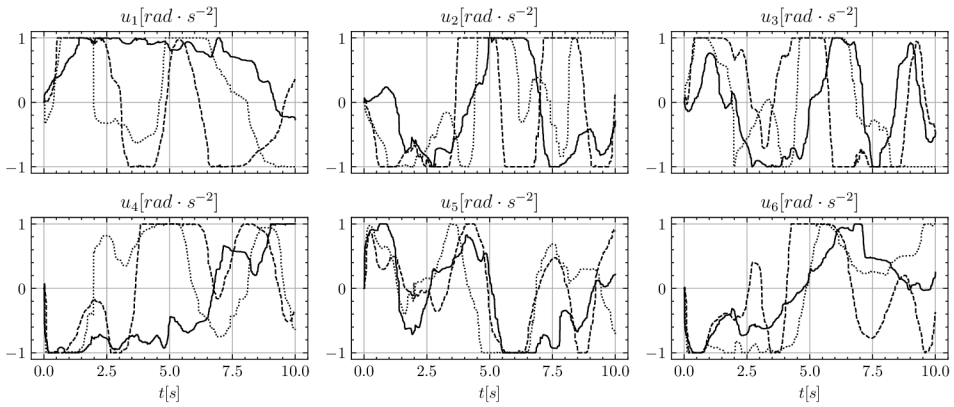

Figure 6.32: Joint angle acceleration inputs in initial testing of coupled MPC.

In the next testing phase, the computational time limit is increased to 10 ms to allow the solver to perform more iterations, thus reaching solutions closer to the optimum.

Since the convergence speed is critical when implementing the MPC, the next step was to evaluate the convergence times of different structures of the problem developed for aerial grasping. The compared structures yield a baseline MPC with no active additional constraints, MPC with operational constraints active (C1) and MPC with perception constraints active (C2). Figure 6.33 shows solver convergence time statistics across multiple simulations with varying the prediction horizon steps. The results show that applying the operational or perception constraints comes at the cost of larger convergence times. Note that simply reducing number of prediction horizon steps is not a good solution since the controller becomes unstable for low number prediction horizon steps. Best performing controllers for quadcopter setting had typically around $K = 20$ prediction steps.

This paper assumes the following initial weight distribution listed in equations (6.14) and (6.15), which are used to initialize CEM optimization, where $\mathbf{Q}^{q}_{MPC}$ is the UAV's state error

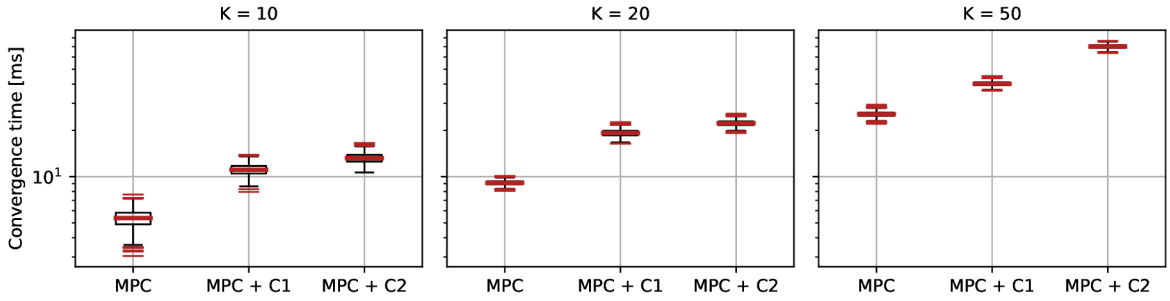

Figure 6.33: Solver convergence for different problem structures.

weighting matrix, $\mathbf{R}^{q}_{MPC}$ is the UAV's control effort weighting matrix, $\mathbf{Q}^{m}_{MPC}$ is the manipulator's state error weighting matrix, and $\mathbf{R}^{m}_{MPC}$ is the manipulator's control input weighting matrix. Matrices $\mathbf{Q}^{q}_{\sigma}, \dots \mathbf{G}_{\sigma}$ denote the standard deviations of the corresponding weighting components that are used to initialize CEM.

$$\hat{\mu}^{CE}_{0} = \left[\text{vec}(\mathbf{Q}^{q}_{MPC})\ \text{vec}(\mathbf{Q}^{m}_{MPC})\ \text{vec}(\mathbf{R}^{q}_{MPC})\ \text{vec}(\mathbf{R}^{m}_{MPC})\ \text{vec}(\mathbf{G}_{MPC})\right] \tag{6.14}$$

$$\hat{\sigma}^{CE}_{0} = \begin{bmatrix} \mathbf{Q}^{q}_{\sigma} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{Q}^{m}_{\sigma} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{R}^{q}_{\sigma} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{R}^{m}_{\sigma} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{G}_{\sigma} \end{bmatrix} \tag{6.15}$$

The initial mean weighting parameters are listed by equations (6.16). Sampling different weighting for the coupled MPC controller yields different responses of the UAV and robotic manipulator.

$$\begin{aligned} \mathbf{Q}^{q}_{MPC} &= \text{diag}\left([2\ 2\ 2\ 1e^{-2}\ 1e^{-2}\ 1e^{-2}\ 0\ 0\ 0\ 0\ 1e^{-1}\ 1e^{-1}\ 1e^{-1}]\right) \\ \mathbf{R}^{q}_{MPC} &= \text{diag}\left([1e^{-2}\ 1e^{-2}\ 1e^{-2}\ 1e^{-2}]\right) \\ \mathbf{Q}^{m}_{MPC} &= \text{diag}\left([0\ 0\ 0\ 0\ 0\ 0\ 1e^{-2}\ 1e^{-2}\ 1e^{-2}\ 1e^{-2}\ 1e^{-2}\ 1e^{-2}]\right) \\ \mathbf{R}^{m}_{MPC} &= \text{diag}\left([1e^{-3}\ 1e^{-3}\ 1e^{-3}\ 1e^{-3}\ 1e^{-3}\ 1e^{-3}]\right) \\ \mathbf{G}_{MPC} &= \text{diag}\left([1\ 1\ 1\ 0.2\ 0.2\ 0.2]\right) \end{aligned} \tag{6.16}$$

The effect of varying parametrization is shown in Figure 6.34 for the evolving UAV states, Figure 6.35 in terms of drone control inputs, Figure 6.36 of the evolving joint states, and Figure 6.37 in terms of robotic arm control inputs. It may be observed that the resulting maneuver highly depends on the selected weighting factors. For example, the dotted trajectory performs a much slower maneuver while maintaining the altitude of the UAV close to the initial one and not successfully achieving a given reference for the end-effector.

On the other hand, increasing the weights too much may lead to overshoot and high relative motion of the end-effector with respect to its target in the terminal phase. Thus, the manual tuning and exploration of the controller behavior suggests that the proposed

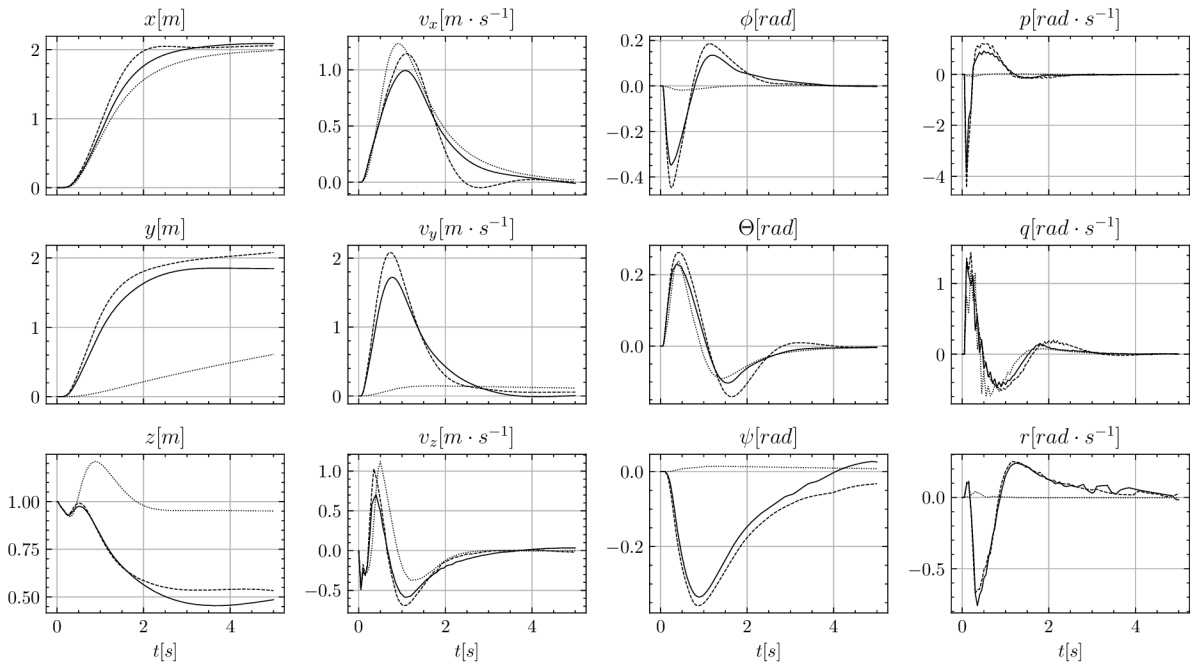

Figure 6.34: Evolving states of a UAV performing a grasping maneuver.

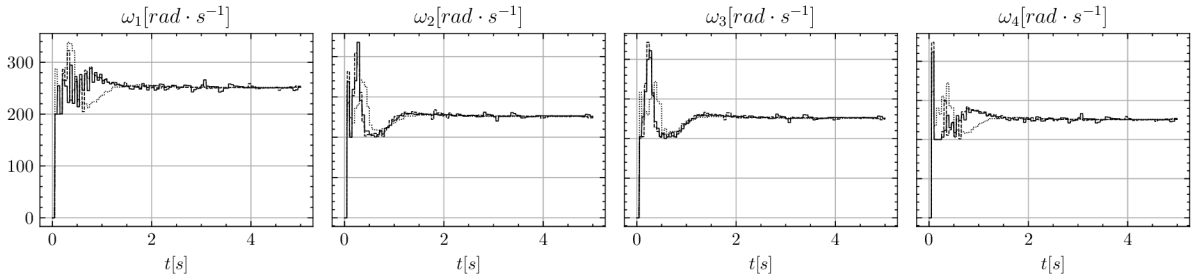

Figure 6.35: Evolving control inputs of a UAV performing a grasping maneuver.

weight-varying approach might be a suitable solution with the aim of adapting the controller behavior during the mission and achieving a better performance with regards to the defined evaluation metrics.

Observing the control inputs of the arm joints, the imposed bounds are often triggered after the start of the maneuver. Since the objective weights penalizing the arm motion were typically set an order of magnitude lower, the arm seems to be incentivised to change its position and orientation and move towards the target. Although the joint accelerations typically converge back to zero at the end of the maneuver, the behavior of the arm could be optimized not to reach actuation limits so often. In the following pages, both fixed-weighting and varying-weighting optimization approaches will be examined to tune the controller. In the case of tuning for a quasi-physical model, tuning the parameters of the adaptive augmentation loop will also be examined. The parameter optimization will be considered in both deterministic and probabilistic sense with respect to the defined ODD, quantifying the probability that the objectives formulated with regards to the grasping maneuver is not met.

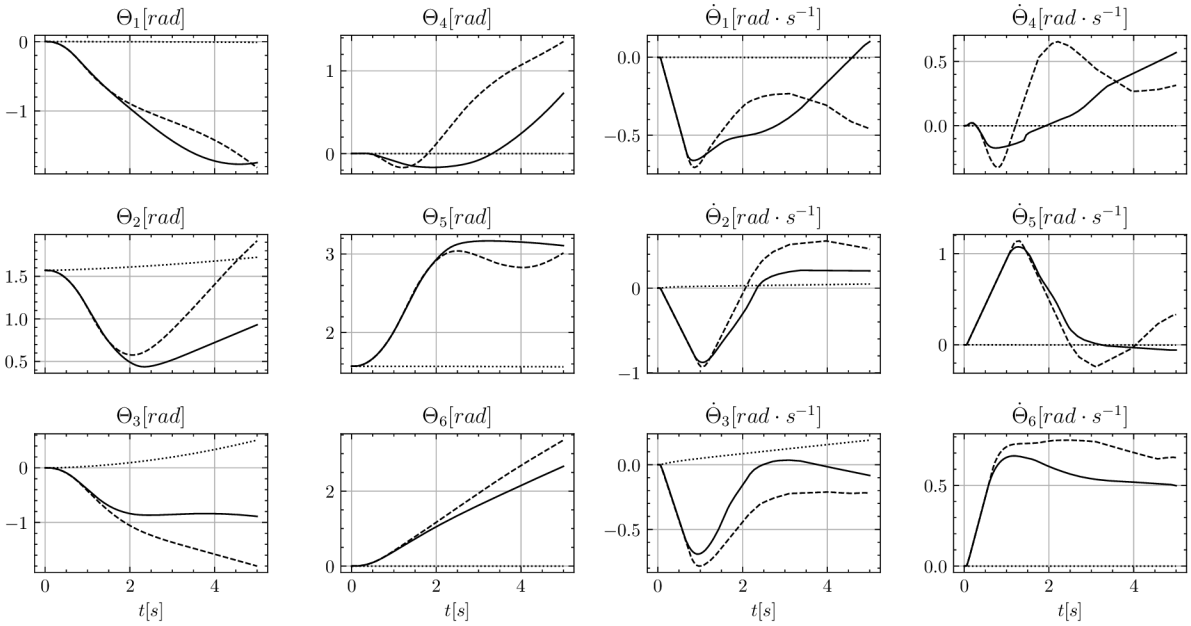

Figure 6.36: Evolving joint states of an arm performing a grasping maneuver.

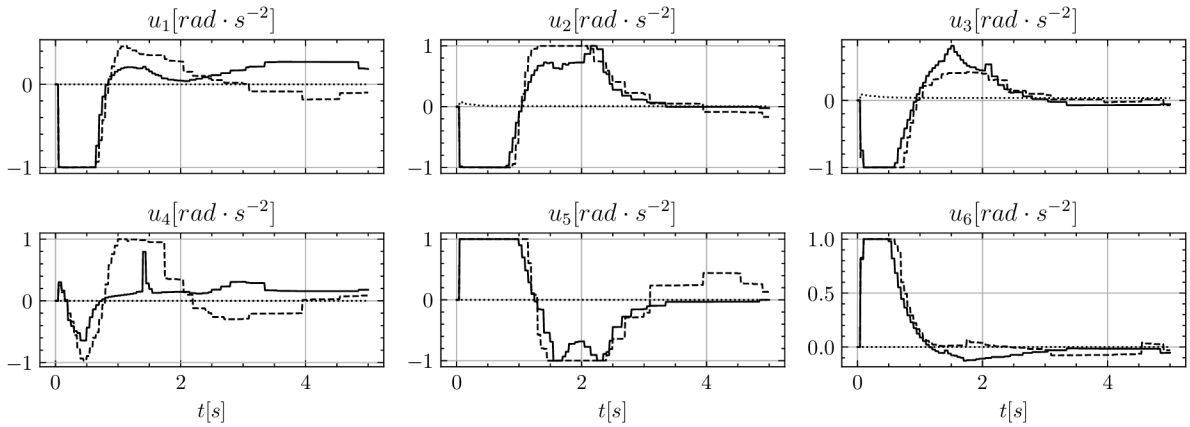

Figure 6.37: Evolving control inputs of an arm performing a grasping maneuver.

Before focusing on the control system optimization process, the effects of imposed safety operational constraints will be shown. Let us consider a case study which assumes the following conflicting reference setting. The UAV target position references are coincident with the end-effector position reference. In terms of MPC formulation, this is not a problem, since both the UAV and the robotic manipulator can kinematically achieve the desired position. However, from a practical standpoint, a collision of the end-effector with the airframe might result in damage or failure of the whole system. Therefore, imposing the safety constraints ensures that the end-effector position cannot exit a defined cone relative to the UAV BFF. The effect of imposing the safety constraints may be seen in Figure 6.38. While the trajectory on the left was computed without the safety constraint, the trajectory on the right has the safety constraints active. It may be seen that despite the reference for the UAV position, the UAV increases the hovering altitude such that the safety constraint is satisfied and the end-effector reaches its reference position.

Based on MPC weighting, the platform can prioritize achieving the end-effector reference, or the UAV position reference but never both at the same time since the second-order cone constraint does not allow coincidence of the end-effector and UAV's body. For identical weights, the UAV nor the end-effector typically reach the desired position as the preference is not clearly set.

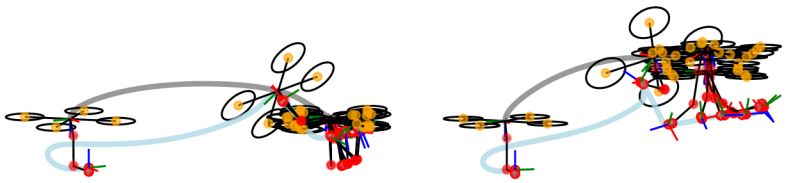

Figure 6.38: Effect of imposing safety operational constraints.

Coupled control of quasi-physical model

The coupled control approach will now be utilized to perform aerial grasping maneuvers using the quasi-physical model, which includes measurement noise, wind turbulence and coupling effects of the UAV and robotic arm. It has shown to be difficult to achieve the desired end-effector position and orientation and maintain it throughout the simulation without minor movements while the system is being subjected to the mentioned uncertainties. A minor improvement was observed by increasing the end-effector weighting factor. The respective visualizations of three maneuvers considering a change of the weighting factor are shown in Figure 6.39.

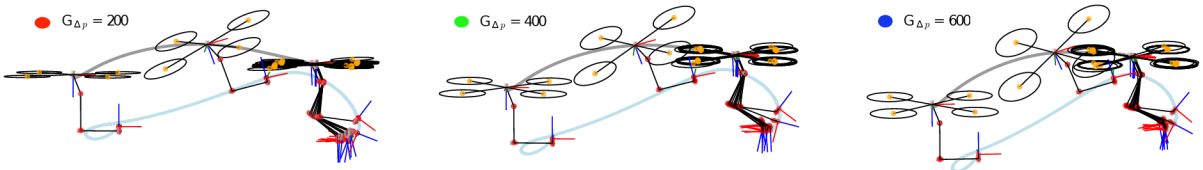

Figure 6.39: Visualization of the aerial grasping maneuver.

Figure 6.40 shows the evolving end-effector position in inertial frame and the respective commanded position denoted by a dashed line. Steady-state error of the position can be observed, which is to be attributed to model discrepancies with regards to the internal MPC model and the quasi-physical model or insufficiently tuned weights of the control system.

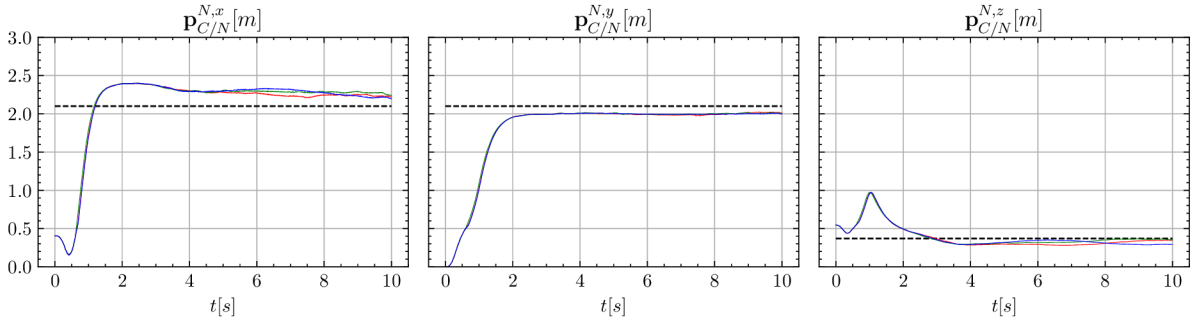

Figure 6.40: End-effector position with respect to commanded reference.

Figure 6.41 shows the evolving end-effector orientation represented by a quaternion and the respective commanded orientation denoted by a dashed line. It can be seen that the orientation is changed and oscillates around the commanded orientation. The goal of optimizing the weighting factors and including the $\mathcal{L}_1$ adaptation law is to suppress the emerging effects and perform the maneuver as smoothly as possible.

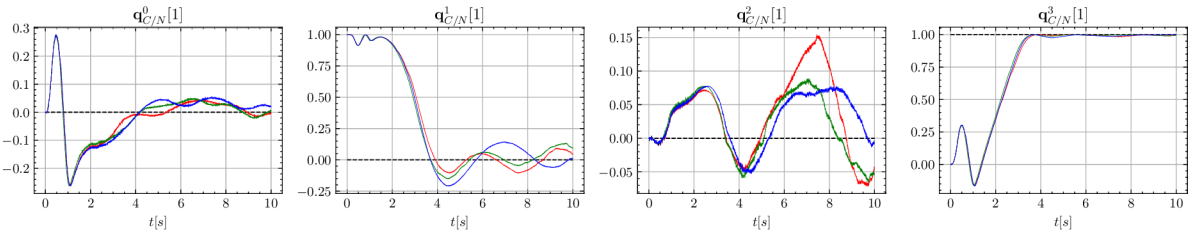

Figure 6.41: End-effector orientation in the inertial frame.

Figure 6.42 shows the respective end-effector acceleration components in the inertial frame. It can be observed that after the initial hover change required to achieve proximity with the target, the end-effector acceleration drops significantly in the final phase of the maneuver.

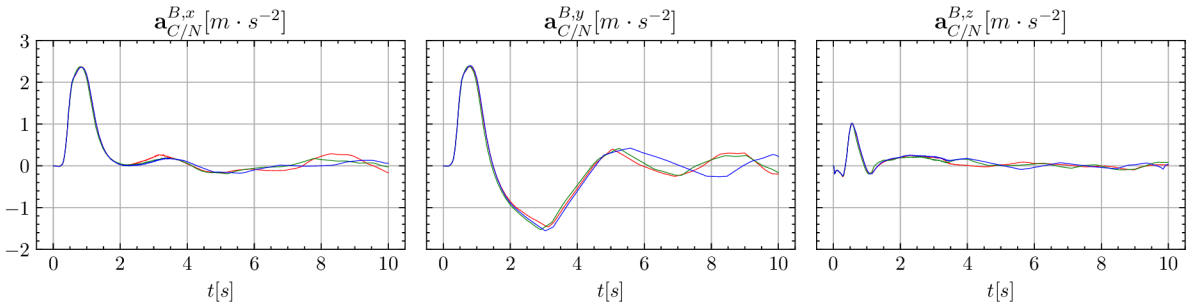

Figure 6.42: End-effector acceleration in the inertial frame.

Similarly, the end-effector angular velocity shown in Figure 6.43 shows that after the initial change of end-effector orientation with respect to the inertial frame, the angular velocity components oscillate around zero. Testing the coupled MPC on a quasi-physical model

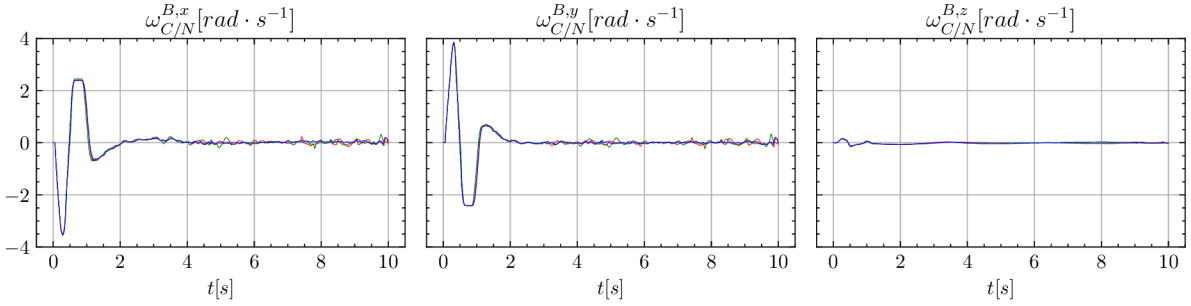

Figure 6.43: End-effector angular velocity in the inertial frame.

showed that the control mechanism is applicable given the discrepancy of the mathematical and quasi-physical model and measurement noise. Optimizing the weighting factors by the CEM or RBCE and evaluating the RMSE of the specified metrics for aerial grasping has the potential to improve the execution of the maneuver.

Figures 6.44 and 6.45 show the evolving states and control inputs of the UAV respectively. In all cases, the UAV achieves its desired position and stabilizes in 3 seconds of the maneuver while the arm gradually changes its pose. Given that the UAV manages to stabilize accurately, the oscillations of the end-effector seem to be induced by the joint movement rather than UAV attitude changes. This motivates further examination of the arm control weighting factors and consideration with regards to the multiobjective characteristics of the problem formulation.

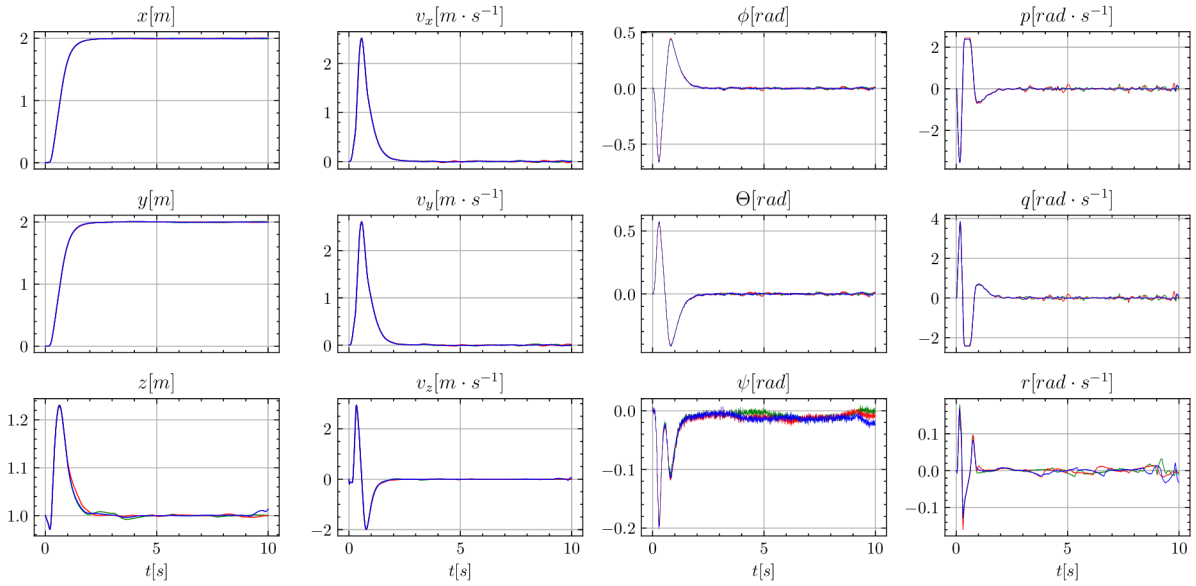

Figure 6.44: Quadcopter states in the considered scenarios.

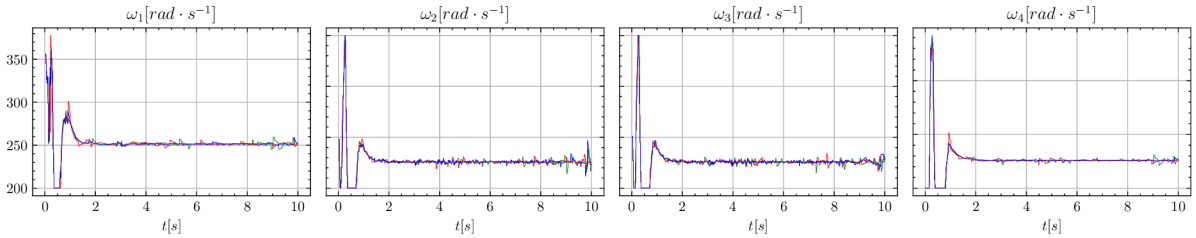

Figure 6.45: Quadcopter motor angular velocities in the considered scenarios.

Table 6.13 shows numerical values of the evaluated kinematic quantities for the end-phase of the maneuver evaluated using $L = 500$ final steps. Note that the value of L has impact on the RMSE values and has to be kept constant in all cases, so that the results can be compared. The sampling time is also kept constant so that the RMSE is computed over the same time section of the simulation. It can be seen that increasing the position weight above a certain threshold does not further improve the RMSE of the end-effector position error.

Table 6.13: Evaluation of the end-effector kinematic errors.

scenario	Δ_p RMSE	Δ_r RMSE	$\|\mathbf{a}^B_{C/N}\|$ RMSE	$\|\boldsymbol{\omega}^B_{C/N}\|$ RMSE
$\mathbf{G}_{\Delta p} = 200$	$1.873 \cdot 10^{-1}$	$6.712 \cdot 10^{-3}$	$1.219 \cdot 10^{-1}$	$1.696 \cdot 10^{-1}$
$\mathbf{G}_{\Delta p} = 400$	$1.555 \cdot 10^{-1}$	$5.187 \cdot 10^{-3}$	$1.867 \cdot 10^{-1}$	$1.555 \cdot 10^{-1}$
$\mathbf{G}_{\Delta p} = 600$	$1.671 \cdot 10^{-1}$	$3.854 \cdot 10^{-3}$	$2.587 \cdot 10^{-1}$	$1.248 \cdot 10^{-1}$

Fixed-Weighting Optimization on Mathematical Model

The first scenario assumes optimizing a set of fixed weights during the CEM. The initial distribution of weighting parameters was set empirically based on the expected range providing valid results. The optimization showed a clear performance improvement with respect to all specified evaluation metrics. The results of optimizing fixed weights are shown in Figure 6.46 including the mean and standard deviation of the CEM convergence with respect to three optimization runs for each evaluation metric using the same initial weight distribution.

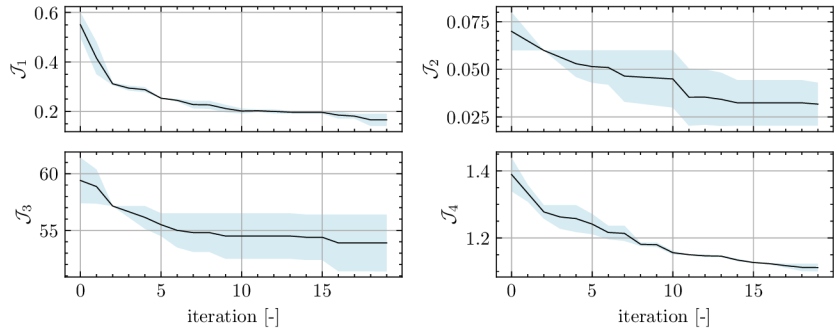

Figure 6.46: Convergence of CEM in fixed-weight problem configuration.

The best fixed weight solution for end-effector accuracy was $\mathcal{J}_1 = 0.17425$. The optimization with respect to acceleration and angular velocity evaluation metric was $\mathcal{J}_2 = 0.0224$. The total energy evaluation metric was optimized to $\mathcal{J}_3 = 54.7809$ J. The combined metric yielded the best result of $\mathcal{J}_4 = 1.1249$ assuming scaling weights $w_1 = 1$, $w_2 = 1$ and $w_3 = 0.01$.

Varying-Weighting Optimization on Mathematical Model

The second optimization scenario assumes a weight-varying MPC driven by ANN. The ANN weights are trained using an RL approach and utilizing CEM for continuous optimization. The effectiveness of this method to RL problems has been demonstrated numerous times, such as in [110]. The proposed strategy utilizes a nominal set of weights $\mathbf{z}_w^{nom}$, which are configured for each evaluation metric individually as the best set of weights obtained by fixed-weight optimization. The CEM manages to optimize the ANN weights and improve the

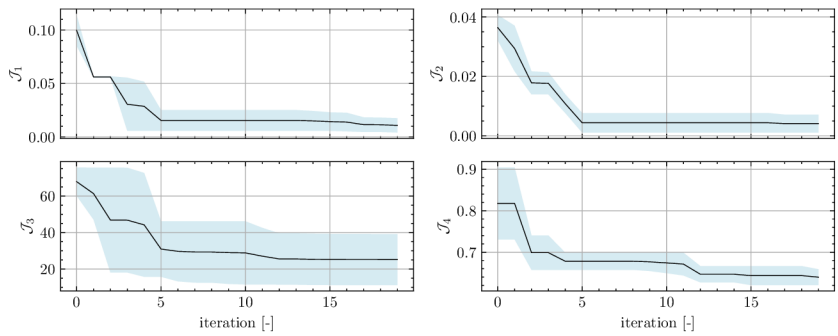

Figure 6.47: Convergence of CEM in weight-varying problem configuration.

overall performance with respect to the designed metrics. Figure 6.47 shows the mean and standard deviation of three training runs for a weight-varying controller. It can be seen that

the weight-varying controller outperforms the fixed-weight controller in all training runs. The best performance in terms of end-effector accuracy was $\mathcal{J}_1 = 4.090 \cdot 10^{-3}$. In the case of end-effector acceleration and angular velocity metric, the best evaluation metric value was $\mathcal{J}_2 = 1.099 \cdot 10^{-3}$, outperforming the fixed-weight controller even more significantly. The energy evaluation metric was optimized to $\mathcal{J}_3 = 1.117 \cdot 10^{1}$ J. Finally, the combined evaluation metric was optimized to $\mathcal{J}_4 = 6.250 \cdot 10^{-1}$. The improvements may be attributed to controller adaptability with respect to the ANN input features.

Fixed-Weighting Optimization on Quasi-physical Model

The third optimization scenario assumes a fixed weight MPC control of the quasi-physical model augmented by the $\mathcal{L}_1$ augmentation loop. Figure 6.48 shows the CEM convergence assuming all evaluation metrics. Compared to the mathematical model, the evaluation

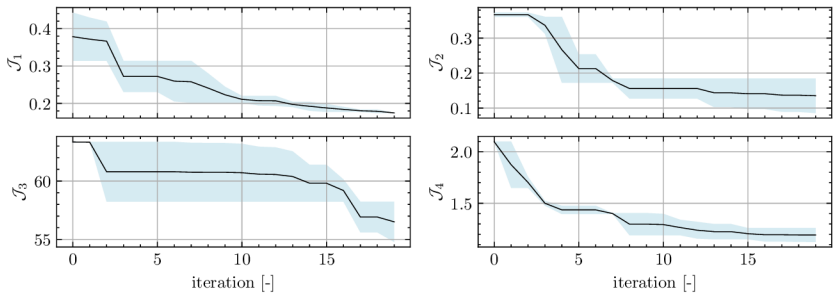

Figure 6.48: Convergence of CEM in fixed-weight problem configuration.

metric values are slightly higher due to discrepancies such as coupling effects since the $\mathcal{L}_1$ augmentation loop is not perfect in canceling out these effects.

Varying-Weighting Optimization on Quasi-physical Model

The fourth optimization scenario assumes a weight-varying MPC control of the quasi-physical model augmented by the $\mathcal{L}_1$ augmentation loop. Figure 6.49 shows the CEM convergence assuming all evaluation metrics. The standard deviations for metrics $\mathcal{J}_3$ and $\mathcal{J}_4$ are significantly

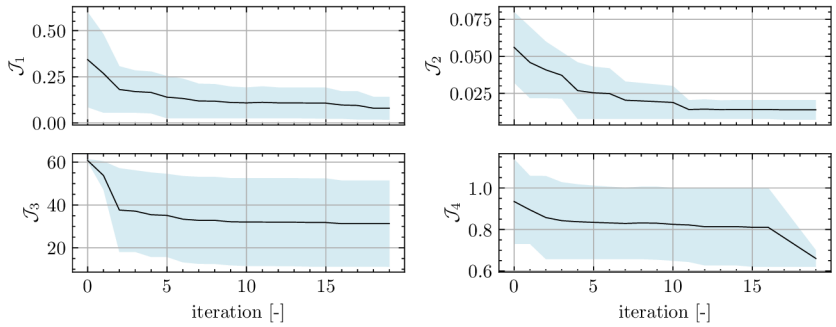

Figure 6.49: Convergence of CEM with weight-varying problem configuration.

higher compared to other scenarios. This can be the result of convergence of the algorithm to local optima. The ANN manages to adapt the weights so that the metric values are lower compared to the fixed-weight scenarios.

Optimized Controller Validation

A validation of selected optimized controllers with respect to the quasi-physical model was performed. Specifically, the weight varying optimized controllers for the end-effector accuracy (red), acceleration and angular velocity (green), and the combined metric (blue) were used in the presented validation simulations. The evaluation is performed using the quasi-physical model in the control setting augmenting the MPC by $\mathcal{L}_1$ adaptation loop.

Figure 6.50 shows that the end-effector achieves the desired position reference in all three cases with no notable oscillations once the end-effector achieves the position.

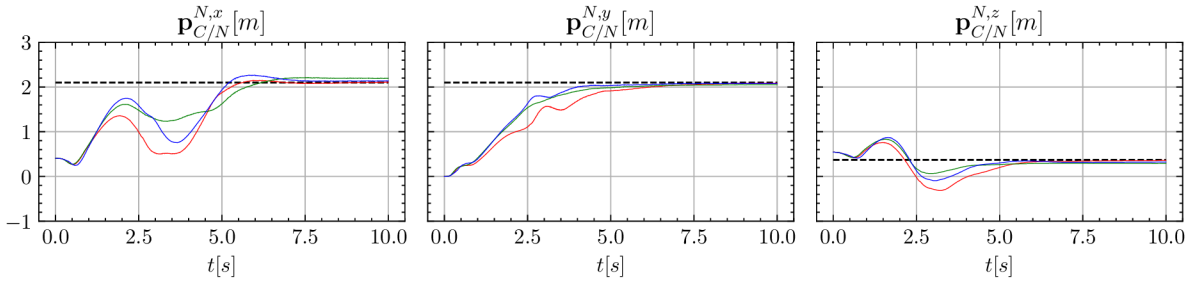

Figure 6.50: End-effector position considering the optimized controllers.

Figure 6.51 shows the change of the end-effector orientation in the selected validation scenarios. The observed error was evaluated to be within acceptable margins as will be shown by computing RMSE error metric.

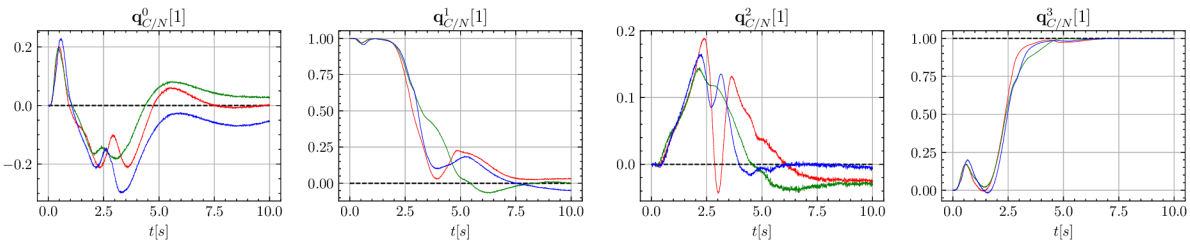

Figure 6.51: End-effector position considering the optimized controllers.

One of the issues encountered when manually tuning the fixed-weight controller was a residual motion of the arm after transitioning to the desired position. This resulted in measuring changes of the end-effector acceleration and angular velocity. Figure 6.52 shows the evolving end-effector acceleration converging to zero after the initial transition phase.

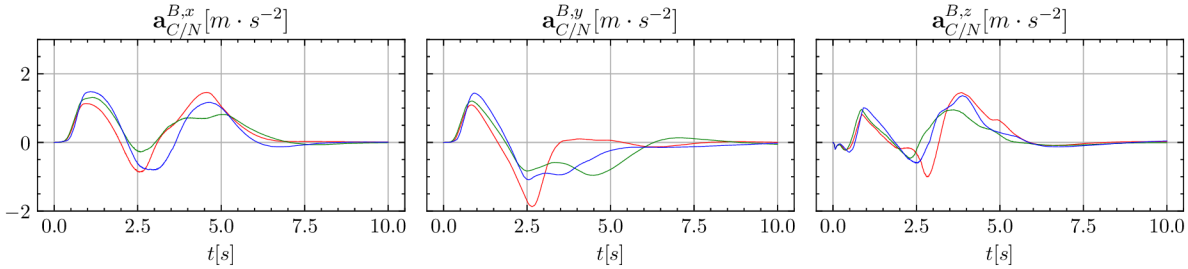

Figure 6.52: End-effector acceleration considering the optimized controllers.

Similarly, the end-effector angular accelerations with respect to the inertial frame are shown in Figure 6.53 and converge to zero after reaching the steady-state.

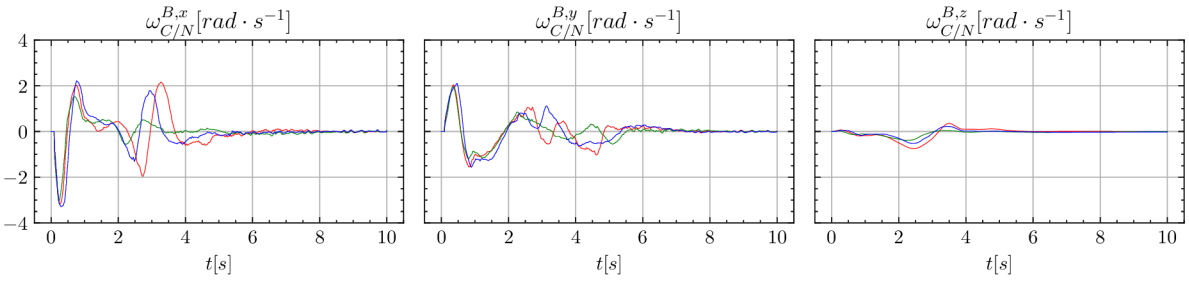

Figure 6.53: End-effector angular velocities considering the optimized controllers.

Figure 6.54 shows the evolving UAV states. It can be seen that the transition maneuver is initiated sharply, and subsequently reaches the steady state. The UAV altitude is not always maintained according to reference since the optimization metrics focused primarily on the minimization of the end-effector motion.

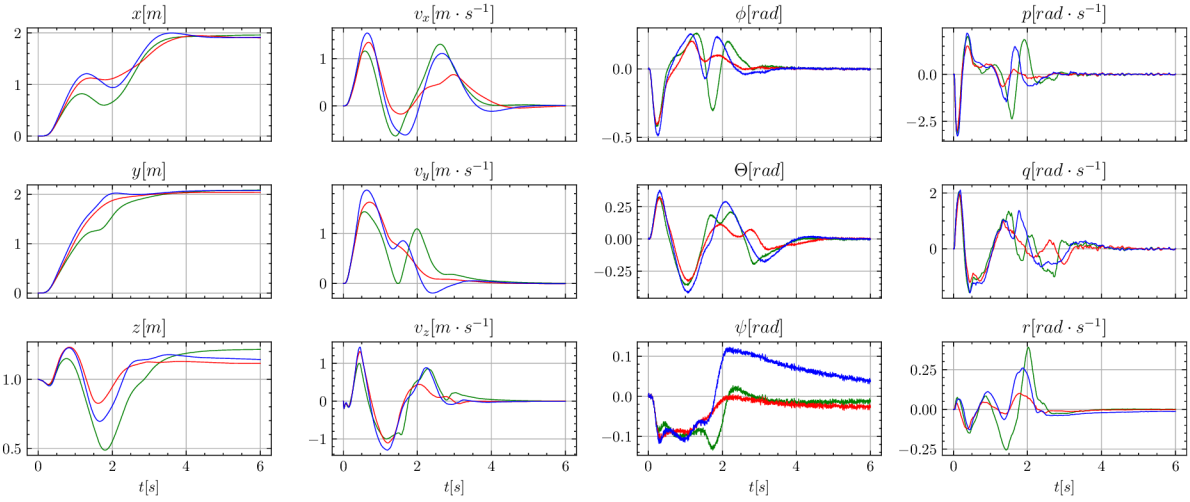

Figure 6.54: Evolving UAV states considering the optimized controllers.

Figure 6.55 shows the UAV control inputs for the selected validation scenarios. One can see that the rotor angular velocities slightly oscillate to maintain the UAV in steady-state with minimum impact on the end-effector motion.

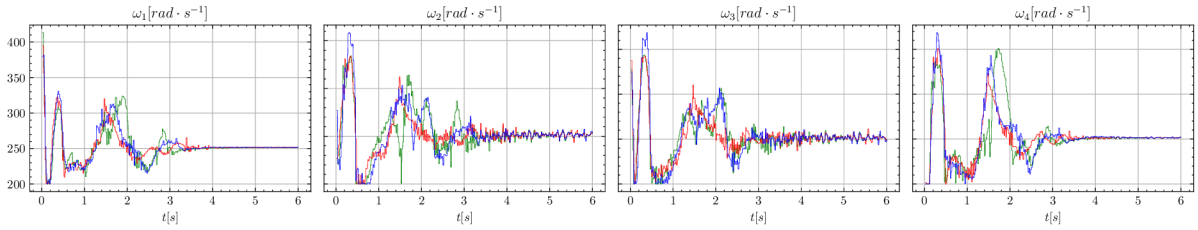

Figure 6.55: UAV control inputs considering the optimized controllers.

Figure 6.56 shows the robotic arm joint states involving the joint angular positions and joint angular velocities. Applying each of the optimized controllers resulted in a different final joint angular position showing that the arm allows for more state configurations that result in reaching the same end-effector state.

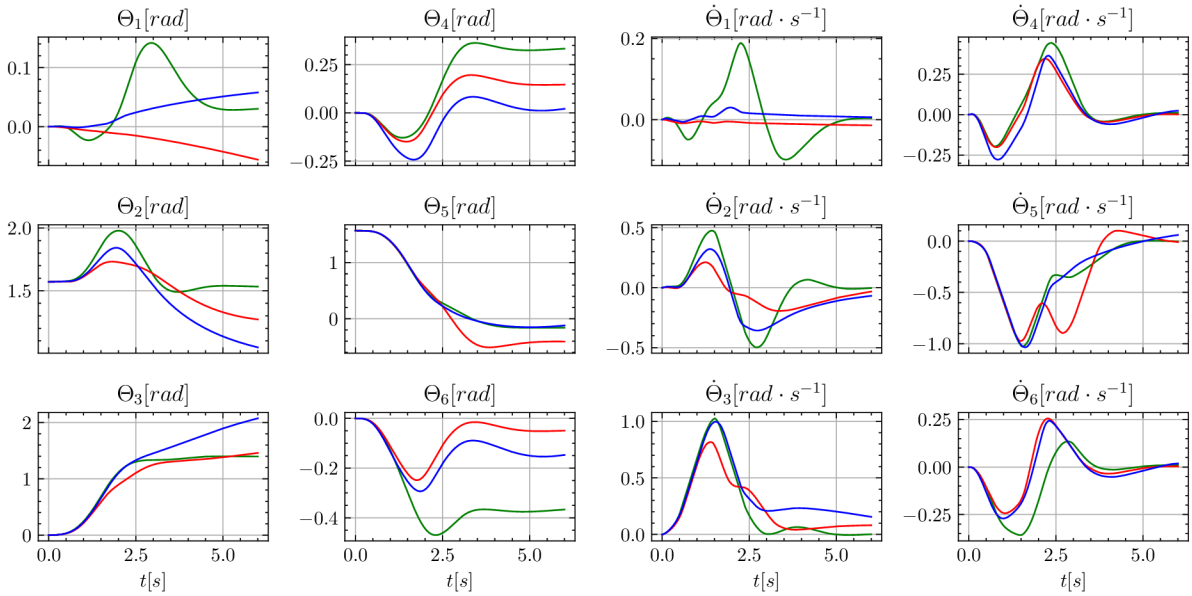

Figure 6.56: Evolving joint states considering the optimized controllers.

Figure 6.57 shows the commanded joint accelerations of the robotic arm. The commanded accelerations converge to zero after the target has been successfully reached. Control action limits were reached for the second, third, and fifth joint during the transition maneuver.

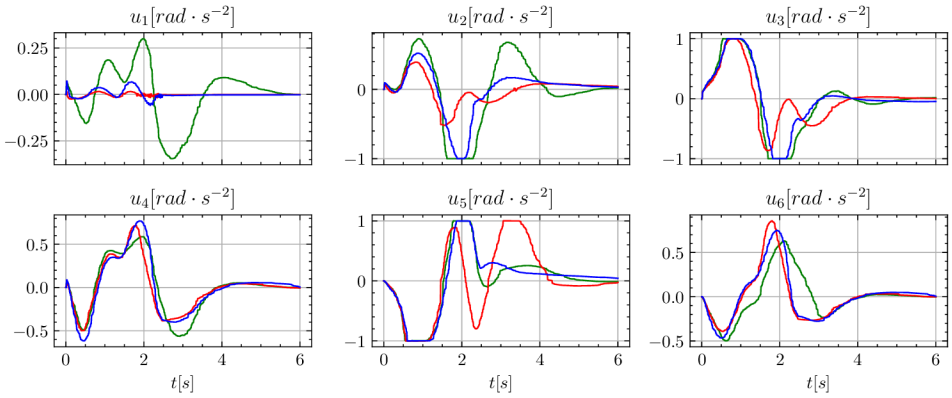

Figure 6.57: Robotic arm control inputs considering the optimized controllers.

Table 6.14 shows numerical values of the evaluated kinematic quantities for the end-phase of the maneuver considering the three optimized controllers. The controller optimized for consumed energy $\mathcal{J}_3^*$ was not included as it did not perform the grasping maneuver sufficiently well.

Table 6.14: Kinematic errors evaluation on optimized controllers.

scenario	Δ_p RMSE	Δ_r RMSE	$\|\mathbf{a}^B_{C/N}\|$ RMSE	$\|\boldsymbol{\omega}^B_{C/N}\|$ RMSE
$\mathcal{J}_1^*$	$2.342 \cdot 10^{-2}$	$4.117 \cdot 10^{-3}$	$3.003 \cdot 10^{-2}$	$2.736 \cdot 10^{-2}$
$\mathcal{J}_2^*$	$8.216 \cdot 10^{-2}$	$7.288 \cdot 10^{-3}$	$2.306 \cdot 10^{-2}$	$1.850 \cdot 10^{-1}$
$\mathcal{J}_4^*$	$6.558 \cdot 10^{-2}$	$5.854 \cdot 10^{-3}$	$5.151 \cdot 10^{-2}$	$4.154 \cdot 10^{-2}$

Grasping maneuver execution using the quasi-physical model

The maneuver is investigated considering three points for the descent initialization. The side view of the resulting simulated trajectories using the quasi-physical model is shown in Figure 6.58 for the $\Delta x = 1$ m, Figure 6.59 for the descent start at $\Delta x = 3$ m, and Figure 6.60 for the descent start at $\Delta x = 6$ m. In this example, optimized fixed weights of the MPC are used. However, the grasping maneuver is more complex as it is not just hover change with arm extension, but a three-phase maneuver.

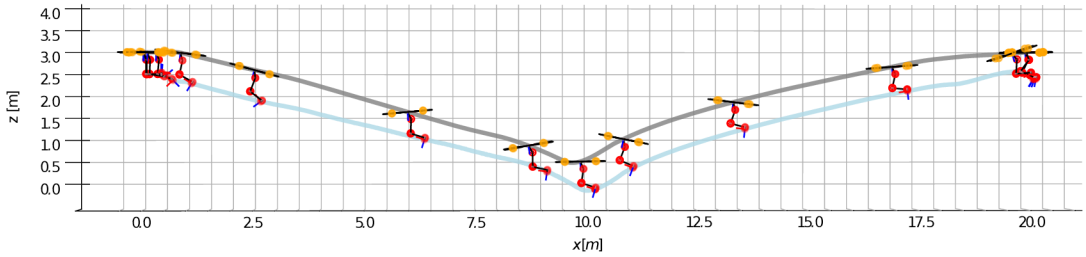

Figure 6.58: Coupled UAV performing shallow descent to target.

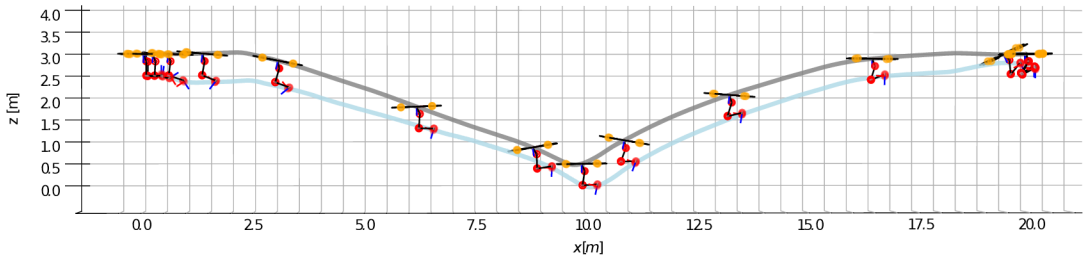

Figure 6.59: Coupled UAV performing medium rate descent to target.

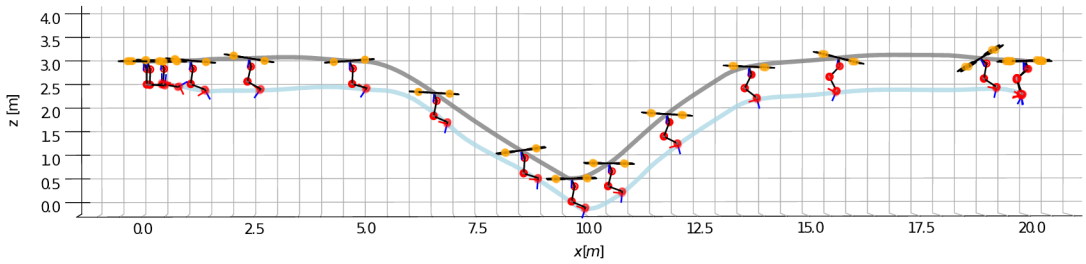

Figure 6.60: Coupled UAV performing sharp descent to target.

In all three cases, the UAV is initialized at the position $\mathbf{p}^{N}_{B/N}(t_0) = [0 \;\; 0 \;\; 3]$ m in a hovering mode with the arm orientation changed in the first seconds of the simulation. The operational constraints ensure that the end-effector position stays within the operational envelope. At the end of the maneuver, the UAV decelerates to establish a new hovering position at $\mathbf{p}^{N}_{B/N}(t_f) = [20 \;\; 0 \;\; 3]$ m. The payload is considered to be located at $\mathbf{p}^{N}_{P/N} = [10 \;\; 0 \;\; 0]$ m.

The state and control time histories for the three considered simulations featuring a quasi-physical model are shown in the shallow descent green-colored maneuver, medium descent rate red-colored maneuver, and the sharp in the blue-colored maneuver. Figure 6.61 shows the evolving state histories of the UAV. Although the trajectory required to perform the grasping maneuver does not consider any offset in the y axis, small deviations due to model and

environment uncertainties may be observed as well as the measurement noise introduced into the model.

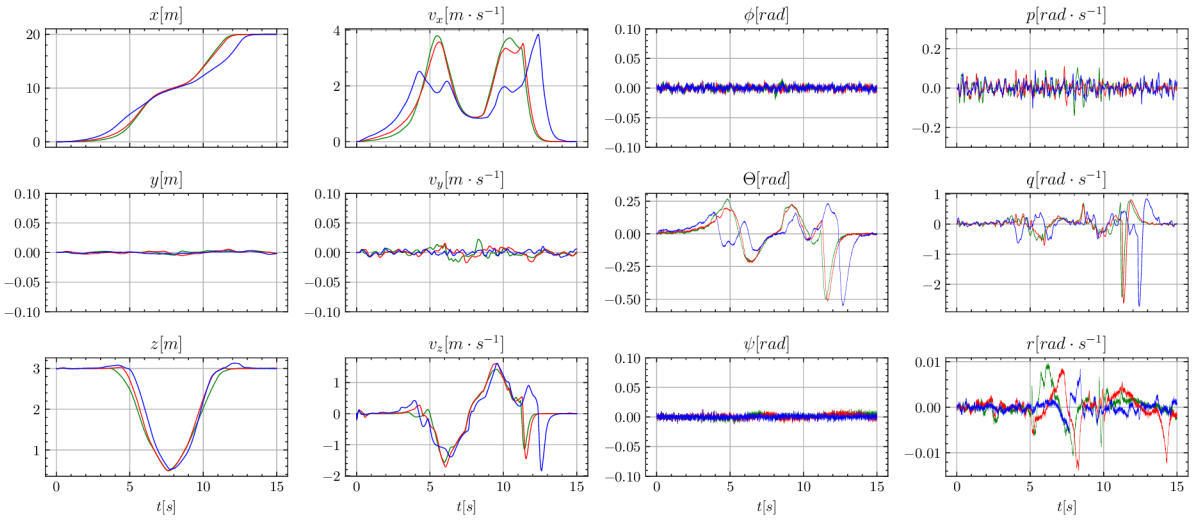

Figure 6.61: Quasi-physical model state trajectories in considered maneuvers.

Figure 6.62 shows the evolving manipulator joint angles and joint velocities across all three simulations. Figure 6.63 shows the UAV propeller angular rates generated by the BLDC motors.

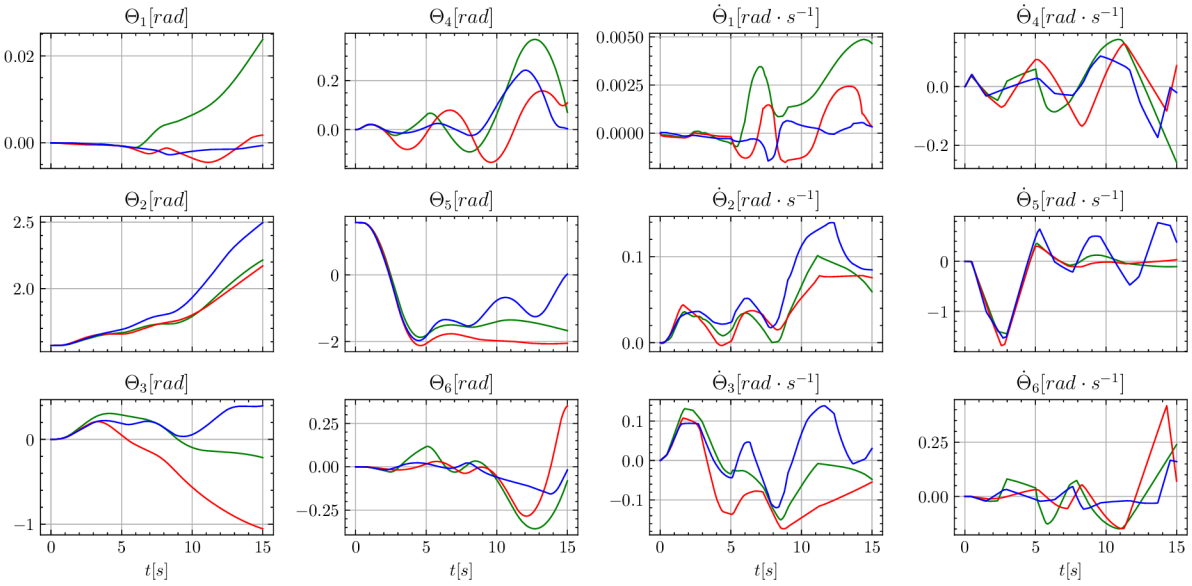

Figure 6.62: Manipulator joint angles and angular velocities.

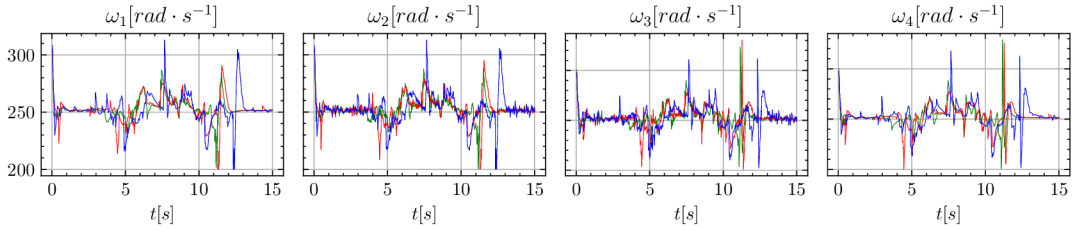

Figure 6.63: UAV motor's angular rates in the considered maneuvers.

To conclude, the optimized controller has been able to perform more complex maneuver than it was initially optimized for. However, errors in end-effector orientation have been

observed as a result of continual movement of the UAV in the grasping phase. For this reason, CEM is utilized again to improve the end-effector's accuracy.

Figure 6.64 shows three simulations showcasing aerial grasping maneuver using different weight parametrizations computed by three consecutive iterations of CEM. The blue-colored trajectory yields the results of the re-optimized controller in the final iteration. The UAV

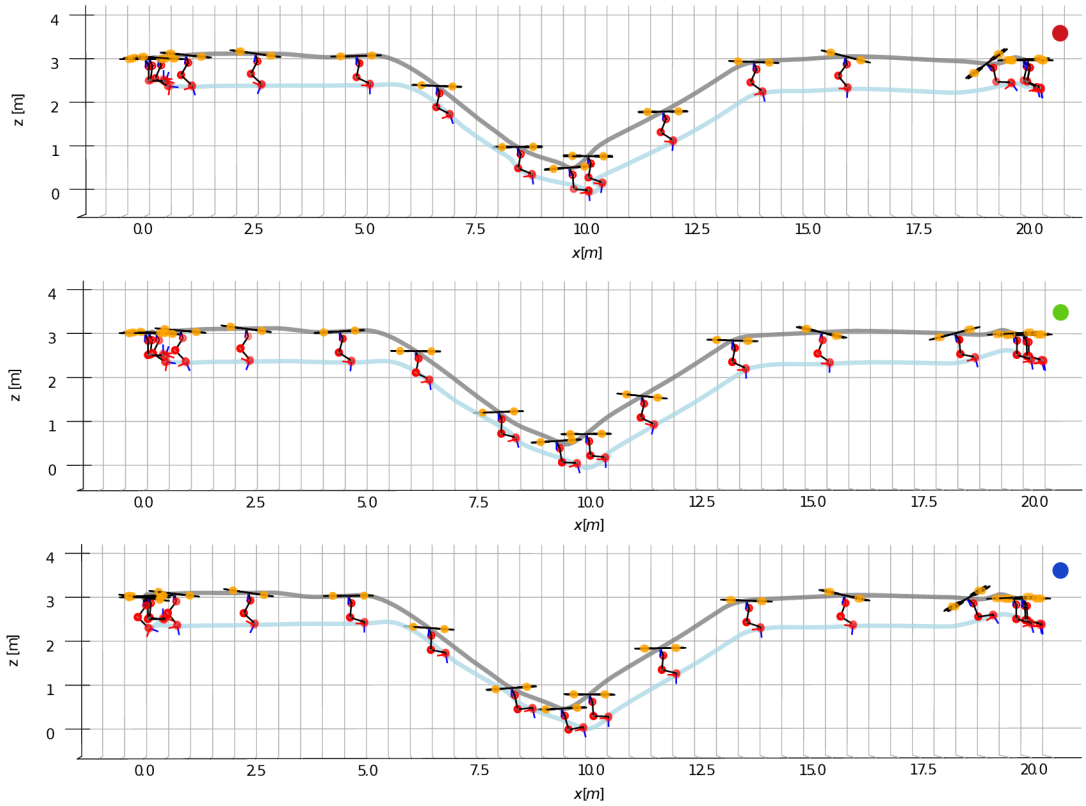

Figure 6.64: Autonomous aerial grasping simulations with varying MPC weights.

begins by deploying it's arm to correct orientation and initiates its approach to target. Subsequently, it decreases its velocity to execute a grasping maneuver. The arm is subsequently retracted during the departure phase of the maneuver. Figure 6.66 shows the evolving UAV states for the three different controller parametrizations. It can be seen that the UAV trajectory has been minimally affected in the last iterations of CEM compared to the arm trajectory. Arm

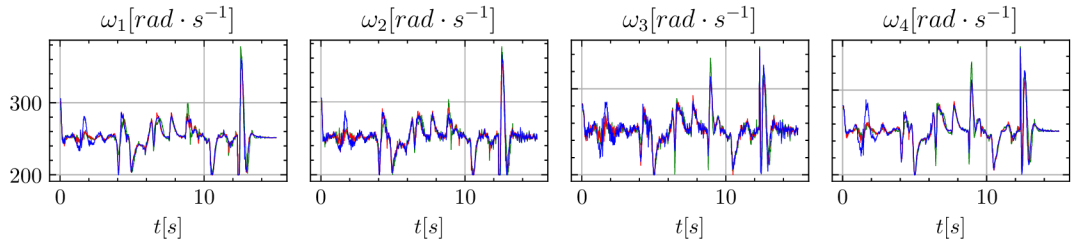

Figure 6.65: Evolving UAV control inputs in the considered scenarios.

joint angular positions and velocities shown in Figure 6.67 evolve differently in each scenario. This is to be attributed to change of weighting factors for the arm and the end-effector as they have the most impact on the resulting evaluation metric value. The applied control inputs computed by MPC are composed of propeller angular velocities shown in Figure 6.65 and

arm joint accelerations shown in Figure 6.68. It can be seen that movement of joints 1 and 4 are very small due to minimal offset of the target in y_N-axis.

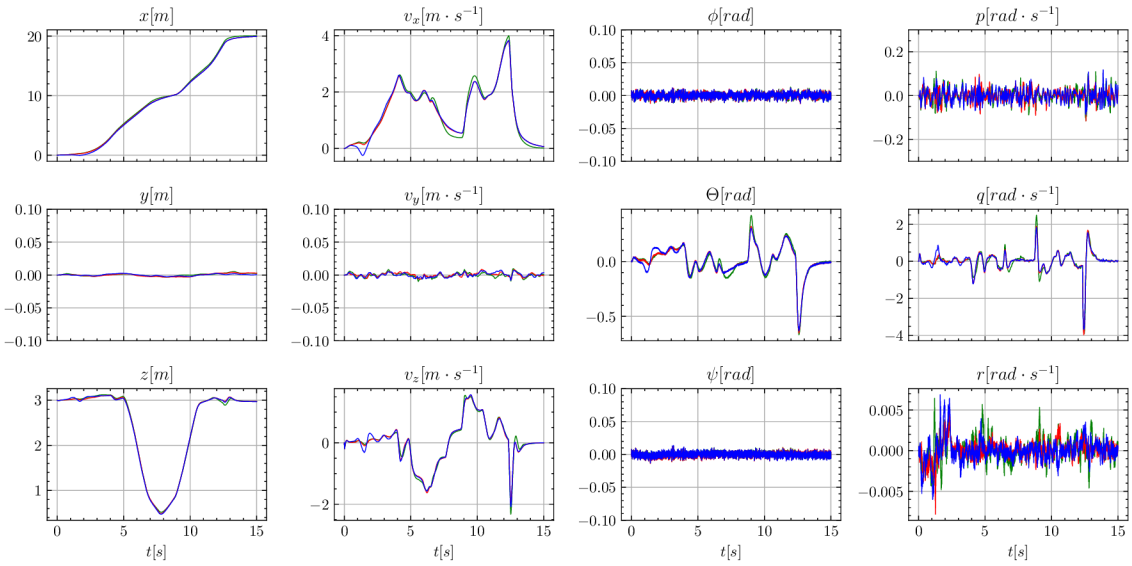

Figure 6.66: Evolving UAV states in considered scenarios.

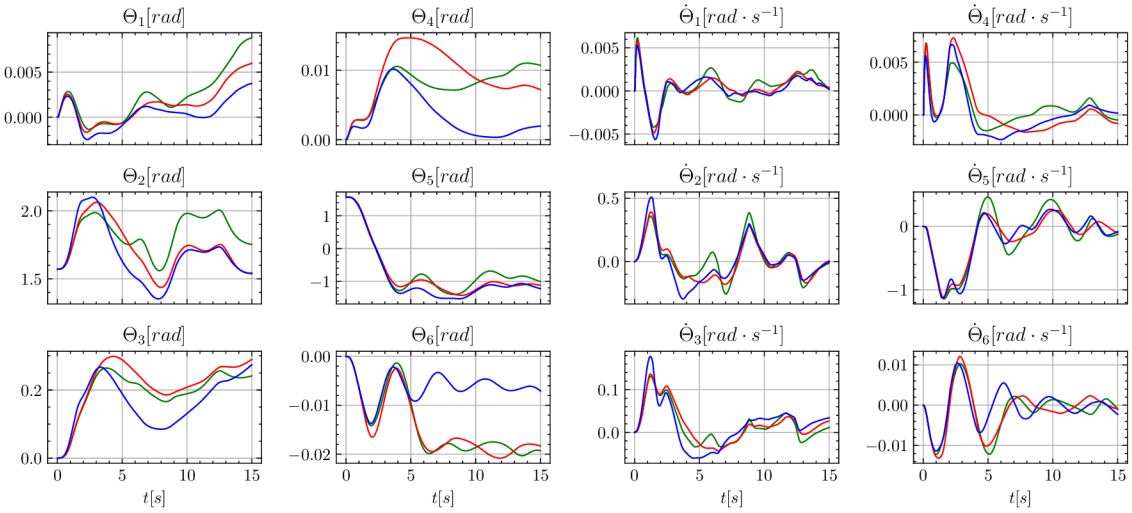

Figure 6.67: Evolving robotic arm states in the considered scenarios.

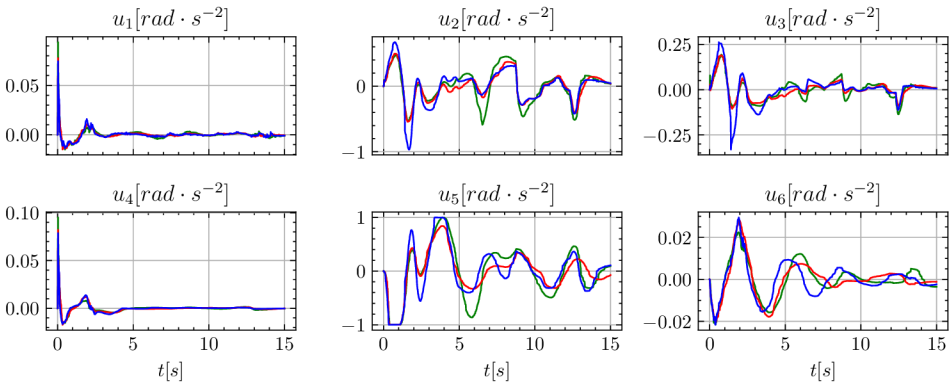

Figure 6.68: Evolving robotic arm control inputs in the considered scenarios.

6.6.1 *Reliability Based Optimization of Aerial Grasping*

Let us now consider the ODD specified earlier, which is composed of probability distributions of wind speed variable and detection slant range variable respectively. The failure probability has been first estimated for the initial design parameters as shown in Figure 6.69. The failure boundary is refined in six consecutive steps. In this example, the considered evaluation metric is the weighted sum of end-effector position and orientation $\mathcal{J}_1$. The resulting surrogate model shows the failure region located primarily in the region of lower detection slant range. This forces the UAV to perform sharper descent resulting in larger error in the end-effector accuracy. Higher wind speeds also lead to unsatisfactory end-effector accuracy in the grasping maneuver.

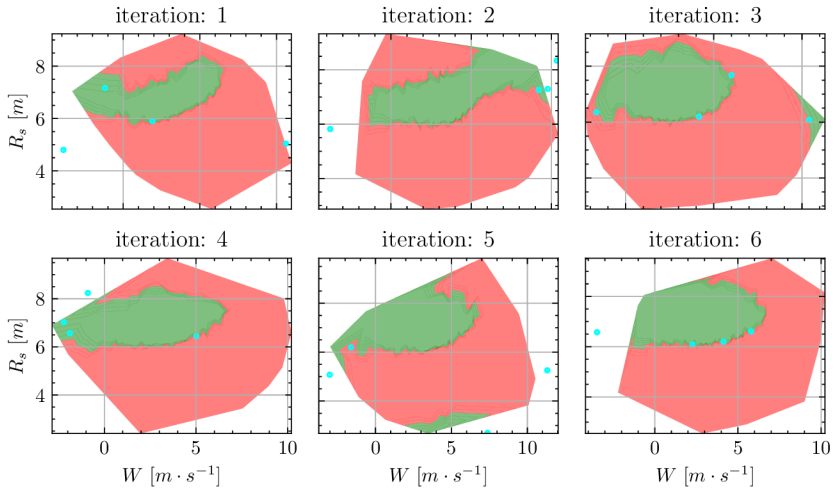

Figure 6.69: Failure boundary before optimization.

The design parameters of the UAV and robotic arm have been subsequently optimized by the RBCE algorithm. The failure probability estimation has been performed using the optimized design parameters resulting in failure boundary region shown in Figures 6.70 and 6.71 in the first and last step of RBCE respectively. It can be seen that the failure region has shrunk significantly compared to surrogate model valuated on the initial design parameters.

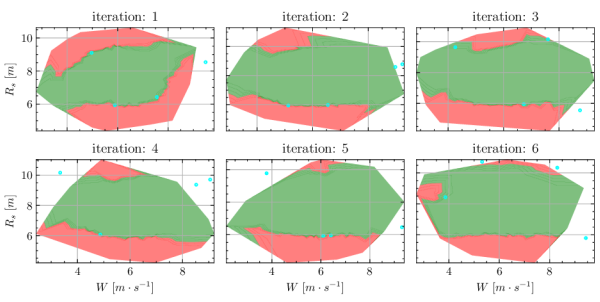

Figure 6.70: Failure region RBCE iteration 1.

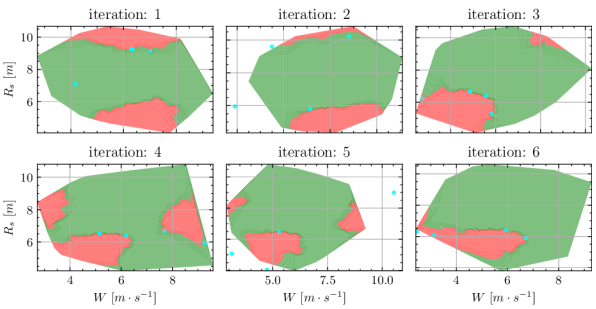

Figure 6.71: Failure region RBCE iteration 6.

Table 6.15 shows performance evaluation of all considered scenarios given the initial design parameters. The considered aerial grasping maneuver is fixed for all scenarios. However, the evaluation metric is computed using simulations of mathematical and quasi-physical model

considering baseline MPC and MPC augmented by adaptive control loop. In all the considered cases, controller has not been able to satisfy the probabilistic requirement.

Table 6.15: Aerial grasping accuracy with initial design parameters.

Scenario	μ RMSE [1]	σ RMSE [1]	P_{fail} [%]	Passed P_{th}
Mathematical model MPC	$4.658 \cdot 10^{-1}$	$9.782 \cdot 10^{-2}$	27.685	False
Quasi-physical model MPC	$4.989 \cdot 10^{-1}$	$1.933 \cdot 10^{-2}$	32.647	False
Mathematical model MPC + $\mathcal{L}_1$	$4.263 \cdot 10^{-1}$	$1.511 \cdot 10^{-2}$	22.144	False
Quasi-physical model MPC + $\mathcal{L}_1$	$4.456 \cdot 10^{-1}$	$1.882 \cdot 10^{-2}$	25.258	False

Table 6.16 shows performance evaluation of the optimized design parameters with respect to modeled uncertainties. In three of the considered cases, the controller is able to pass the imposed probabilistic requirements.

Table 6.16: Aerial grasping accuracy with optimized design parameters.

Scenario	μ RMSE [1]	σ RMSE [1]	P_{fail} [%]	Passed P_{th}
Mathematical model MPC	$1.641 \cdot 10^{-1}$	$1.587 \cdot 10^{-2}$	1.113	True
Quasi-physical model MPC	$2.753 \cdot 10^{-1}$	$5.896 \cdot 10^{-2}$	6.486	False
Mathematical model MPC + $\mathcal{L}_1$	$2.152 \cdot 10^{-1}$	$2.751 \cdot 10^{-2}$	0.853	True
Quasi-physical model MPC + $\mathcal{L}_1$	$2.266 \cdot 10^{-1}$	$4.533 \cdot 10^{-2}$	2.558	True

Figure 6.72 shows the optimized approach to the target yielding a best RMSE error $\mathcal{J}_1 = 0.0247$. Instead of flying directly to the target location, the UAV initially climbs and then descents with lower horizontal velocity to the location of payload. The RBCE algorithm has shown that this approach trajectory yields the highest probability of success with regards to meeting the end-effector accuracy requirement. This approach also seems to have a positive effect in cases where the detection system detects the payload on lower range as the MPC weights force the UAV to reduce horizontal velocity substantially.

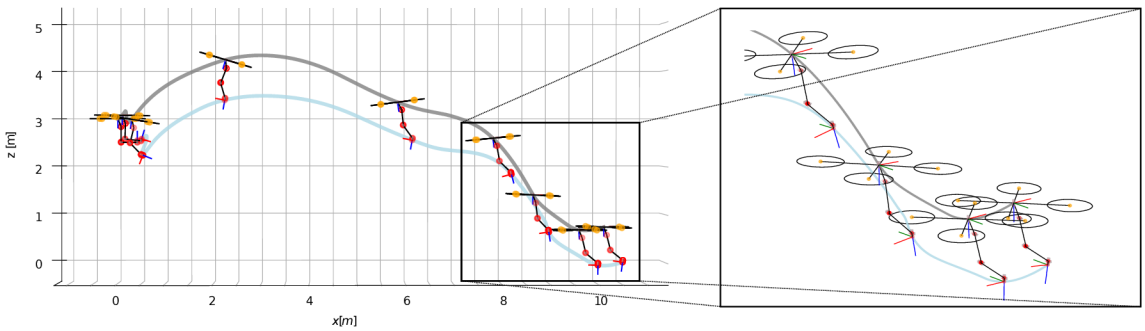

Figure 6.72: Optimized aerial grasping maneuver robust to uncertainties.

6.7 INTEGRATION TO PHYSICAL SYSTEM

To demonstrate MPC capabilities and feasibility in real-world settings, the experimental test case considering an aerial refueling task showcasing the integrated movable arm on the vehicle was conducted. The proposed experiments test the maneuvering capabilities of a multirotor UAV platform with the task of establishing contact with a static or dynamic target and subsequently transferring a liquid onboard a vehicle to the target platform.

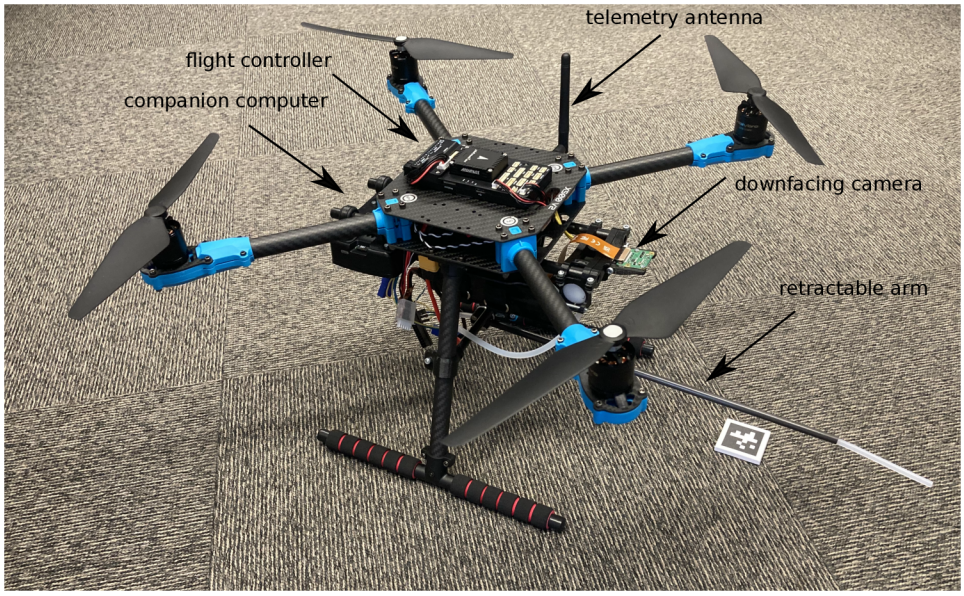

Figure 6.73: Assembled drone platform with an integrated movable arm.

The full set of hardware components which were used to assemble the autonomous drone platform enabling transfer from or to a liquid container of the drone through the integrated robotic arm extension is listed in Appendix D.

The flight software is composed of three major parts running on two computational platforms. The Raspberry pi computer runs a compiled Rust code, efficiently solving a given OCP at up to 150 Hz. A second thread captures camera frames to detect the reference position and orientation with respect to the target AprilTag. The flight management code on the Raspberry Pi combines the data received from the AprilTag detector and the current attitude received from the Pixhawk board's IMU, and inputs the state estimate to the MPC. The Raspberry pi flight management software also integrates the control of the arm servo and the peristaltic pump. The software includes a basic *hover* mode maintaining a fixed relative position to the AprilTag, a *takeoff* mode, *landing* mode, and *capture* mode, which commands the UAV to approach a liquid container and transfer the liquid by the extended arm. A simple block diagram of respective components is shown in Figure 6.74.

Figure 6.75 shows a Simulink® block diagram of the flight controller. The hardware support for the Pixhawk gives access to both raw IMU data and the estimated attitude of the attitude EKF implemented in the baseline firmware. The implemented control logic is utilized

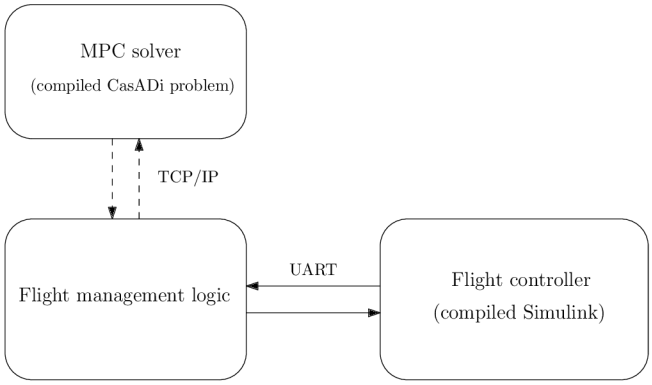

Figure 6.74: Block diagram of the interconnected software components.

when arming the PWM through the radio controller. The failsafe mode can be triggered when exceeding the maximum allowed pitch or roll angle of the vehicle. A connected Raspberry pi device feeds the computed MPC outputs to this controller's logic. In case the autonomous mode with the MPC fails, the controller switches attitude control to a baseline PID controller and turns to manual control by the radio controller. The estimated attitude is sent to the Raspberry pi, which uses the information to compute the OCP at a given time. The controller operates at 250 Hz. The companion computer runs three independent threads.

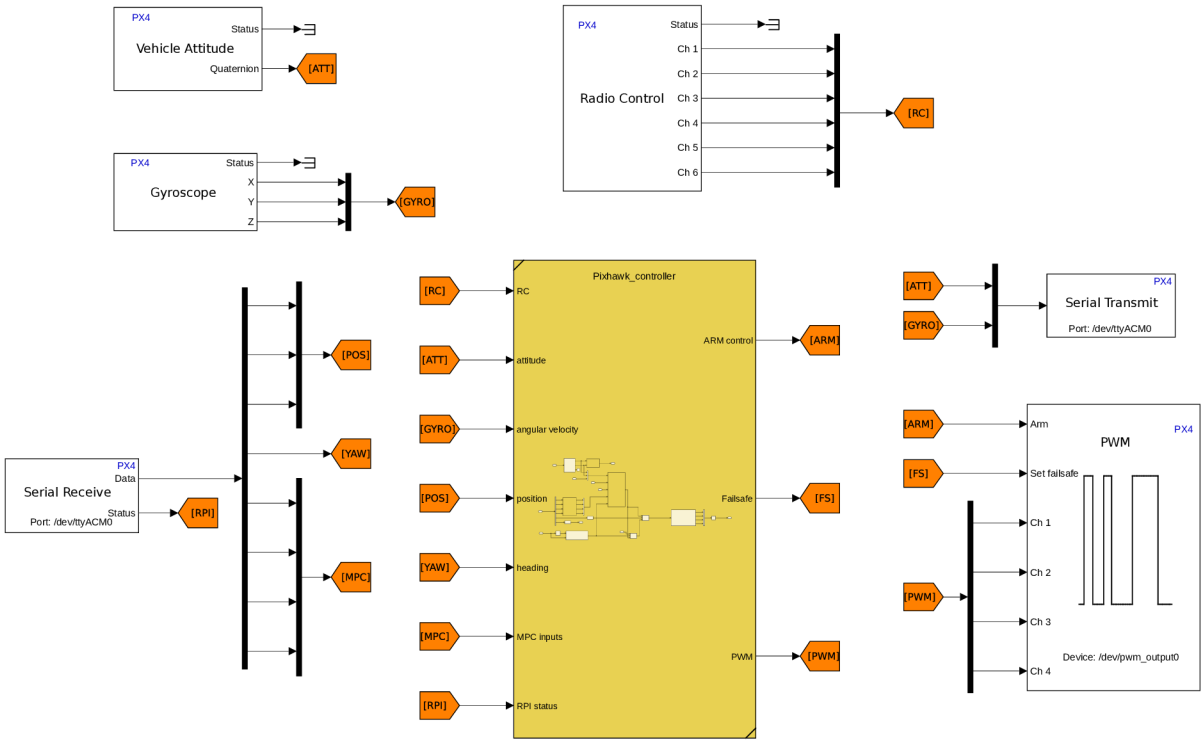

Figure 6.75: Block diagram of the flight controller used in the modified firmware.

The main thread accepts commands from the operator which include number of testing routines and flight management commands. The second thread runs MPC thus continually updating the commanded control inputs whenever the vehicle becomes armed and ready to fly. The third thread runs the computer vision algorithm to estimate vehicle pose based on a detected AprilTag. The pitch and roll of the vehicle estimated by fusion of accelerometer and gyroscope measurements is fused with the image-based pose at lower sample rate. The

relative heading of the vehicle is based on AprilTag's orientation. In case no AprilTag is detected, the UAV automatically transitions to a hover mode. In case the operator activates mission mode, the UAV references a relative position and orientation to the AprilTag and moves the arm to a desired location relative to the AprilTag.

Figure 6.76 shows two zoomed AprilTag images captured by the downfacing camera of the quadcopter with the estimated pose drawn on top of the AprilTag image. The detection algorithm is adopted from [80] and [119]. While the image on the left shows the AprilTag in a relative heading around ≈ 135 deg, the image on the right shows an aligned AprilTag with relative heading around ≈ 0 deg, showing the end-effector retracted to the target.

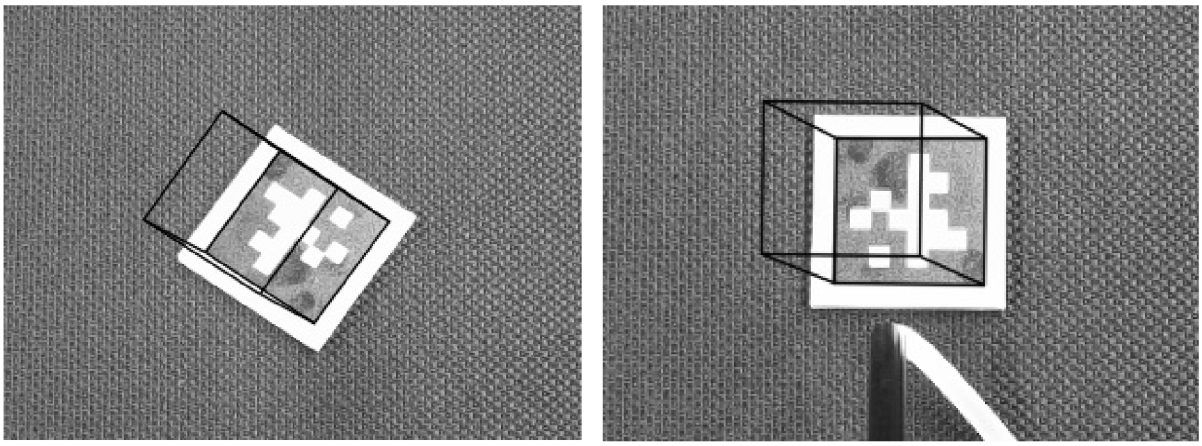

Figure 6.76: AprilTag detection using the downfacing camera.

The final step of integration to physical drone constituted of performing system and subsystem testing routines. They are listed as follows:

1. *Pixhawk receiver test* – Record received sensor data from Pixhawk and store it on Raspberry Pi device. Check the validity of incoming data and update frequency.

2. *Computer vision test* – Generate a sequence of images using the down-facing camera at different relative distances and orientations of the AprilTag. Find the limits of used detection algorithm and adapt the flight management software accordingly.

3. *Attitude fusion test* – Perform fusion of attitude estimated by Pixhawk and camera based on AprilTag detection. The heading relative to AprilTag is used as a primary component driving the yaw motion. Store the results on Raspberry Pi device.

4. *Nonlinear controller test* – Run the control algorithm and send the commands to motors with no propellers on. Change attitude of the drone manually or using a Stewart's platform and validate the commands computed by nonlinear solver compared to expected control allocation.

5. *Manual control test* – In case no position reference is provided by a camera, a relative position error can be set manually to control the hover position. This test can be used to tune scaling factors with respect to manual control inputs on a joystick or radio control.

After successfully performing the specified tests, the UAV is ready to perform first experimental missions. Final goal is to perform coordinated maneuver involving the UAV and robotic arm in a dynamic target scenario described below. The considered missions are ordered accordingly based on complexity.

1. *Takeoff and Landing* – Perform takeoff and landing while minimizing change of horizontal position. This flight test provides insight in MPC model accuracy and appropriate weighting. The UAV's position is referenced relative to the AprilTag.

2. *Hover with arm movement* – Coordinate the integrated arm movement in a hovering mode. Test that the coupling effects are negligible with the considered configuration.

3. *Follow AprilTag* – Let the drone keep a constant relative position and orientation with respect to the AprilTag. The AprilTag is placed on movable platform such that the drone's maneuverabilty is tested.

4. *Static liquid transfer mission* – Approach a target location in vicinity of liquid container. Transfer liquid from the target container to the on-board container and transition back to original hover position.

5. *Dynamic liquid transfer mission* – Approach a target location in vicinity of liquid container placed on movable platform. Transfer liquid from the target container as it continually changes it's position. Transition back to original hover position.

7 CONCLUSION AND PERSPECTIVE

Increasing the need for vehicle autonomy poses an important challenge with regards to the safe deployment of GNC to predominantly uncertain environments. With the rising potential of ML algorithm deployment for computer vision-based situational awareness or vehicle guidance or control tasks, the need for novel verification and validation methodologies is crucial. This thesis addressed the black-box verification and validation of Optimal Control based on-board algorithms and adaptive control systems of underactuated UAVs with increasing levels of complexity up to coupled control for aerial grasping UAV missions. Aerial grasping serves multiple application domains such as autonomous transportation or infrastructure repair such that it enables physical interaction with the environment.

The following sections conclude the achieved results, describe potential future research opportunities, and summarize the author's publications submitted during his work on this thesis. At the end, the author reflects his involvement in scientific/industrial projects as a research group member.

SUMMARY OF ACHIEVED RESULTS

This thesis addressed the challenges of the safe deployment of optimization-based and adaptive algorithms for UAVs with varying levels of complexity. The work presented a UAV platform, a robotic arm platform, and a coupled platform of UAV equipped with a robotic arm tailored for missions involving aerial grasping or other tasks requiring physical interaction with the UAV platform. The work showed, that solving the control tasks for such systems as coupled ones has the benefit of an increased coordination and efficiency with respect to the mission tasks. For nonlinear MPC, the achieved convergence speeds on a personal computer have enabled running a control system algorithm at the maximum of 500 Hz for solving OCP for the 6-DoF quadcopter, and the maximum of 100 Hz for a quadcopter equipped with a six-joint robotic arm. On embedded devices, the convergence speeds achieved enabled control at 150 Hz for the quadcopter case. Unfortunately, convergence of a coupled system has not been sufficiently high to perform a real-time control on this multi-degree of freedom system, reaching update rate around 40 Hz. However, augmenting the adaptive augmentation loop enables higher frequency at the control input level. This concludes the hypothesis with regards to the viability of nonlinear optimal control for the deployment on autonomous systems.

Initial numerical experiments of the coupled system simplified the problem by assuming only the arm kinematics without considering the coupling effects arising from the arm movement, the changing moments of inertia, and the changing UAV's mass when grasping

an object. Therefore, the work subsequently studied the effect of the model mismatch on the performance of nonlinear MPC. The $\mathcal{L}_1$ adaptive augmentation loop was introduced to the control system to increase the robustness with respect to such uncertainties. Furthermore, the adaptive augmentation loop is able to operate at a much higher frequency, enabling UAV attitude stabilization even when baseline MPC update rate is not satisfactory such as in the case of solving OCP for the coupled system. The surrogate model results showed an increased robustness with respect to the assumed uncertainties. This concludes the second hypothesis presented in this work.

The coupled control of the multi-degree of freedom system was shown to be highly dependent on the objective function parameters. In this thesis, a neural network architecture driving the MPC parameters was proposed. Training this network was achieved in RL setting utilizing the CEM optimization. This allows for training the network to prioritize the desired behavior of the control system based on the defined evaluation metrics. This work assumed evaluation metrics with respect to the grasping maneuver that consider the end-effector acceleration and angular velocity with respect to the inertial frame, the achieved accuracy of the end-effector position, and the orientation and consumed energy during the maneuver. The weight-varying MPC can be regarded as more robust than RL trained directly at control input level, since the proven nonlinear optimization solver is used to find optimal control input values instead of a black-box neural network model.

Furthermore, the thesis focused on application of surrogate models to estimate the failure probability of optimization-based and adaptive control systems. To estimate a failure probability, the ODD has to be specified in terms of probability distributions over the considered conditions. To minimize the number of simulated evaluations, a PCE surrogate model with Kriging meta-model was utilized. The thesis proposed a RBCE method to simultaneously estimate the failure probability and accordingly optimize the controller parameters. The thesis shows the use of this method optimizing control system parameters using high-fidelity UAV simulations that involve complex aerodynamic effects, measurement noise, and actuator dynamics. Since such simulations become computationally expensive, surrogate models help to lower the number of required simulations when estimating the failure probability. The author notes that the ODD distributions, evaluation metric thresholds, and maximum allowed failure probabilities were set empirically and may be subject to change based on application.

To summarize, this work addressed key aspects of the GNC development process for autonomous systems utilizing optimal control and ML methods. Safety verification aspects play a crucial role as new optimization-based, adaptive, or ML-based GNC algorithms are to be deployed on safety-critical systems such as UAVs. A specialized set of autonomous systems termed as coupled were introduced. MPC was utilized to perform tasks such as aerial grasping maneuvers which would be difficult to achieve by standard control methods that do not consider coupling of the multi-degree of freedom system's motion.

FUTURE WORK

The author has addressed several current topics which have emerged in the control theory community and machine learning community. The connection of model-based optimization and learning has gained a lot of interest with regards to research, increasing the autonomy of CPS. Furthermore, the aspect of certification of black-box systems has also been in the forefront of applied machine learning in the aerospace engineering community. As the importance and capabilities of ML systems grow, the demand for verification and validation of data-based systems will grow and require new mathematical concepts and structures surpassing the current surrogate model-based or formal methods. The author's research will tackle these topics in more depth in his follow-up research.

Based on the topics encountered in this dissertation, a novel branch of research concerning bio-inspired GNC seems to be taking shape as a promising direction of future research. The author has encountered OCPs which have been observed in natural environment such as dynamic soaring or aerial grasping. Mastering such techniques could widely extend the operational capabilities of both multirotor or fixed-wing UAVs. Bio-inspired GNC could promote new configurations, structures, and, in consequence, the use of alternative materials. Flapping wing miniature UAVs are a good example of emerging bio-inspired aerial robots already in active development. Beyond aerial platforms, bio-inspired GNC could shape the future of the emerging soft robotics, multi-environment robotic platforms, and robotic applications in medicine.

With regards to aerial grasping described in this thesis, a multitude of aspects can be studied further. The UAV configuration may be adapted for specific tasks of aerial grasping studying varying airframe structures and robotic arm configurations. The use of flexible grippers or jamming grippers can also be studied in the future. A perching mechanism could extend the potential range of autonomous UAVs by the possibility of recharging at the top of trees using on-board solar arrays.

The conducted research concerning MPC for aerial platforms also offers interesting research directions. Apart from the studied weight-varying approach for aerial grasping, the emerging safe MPC architectures based on control barrier functions are currently studied. A direct inclusion of sensing information to MPC also poses a challenge. On-board optimization is still limited for safety-critical applications mainly to convex optimization problems due to the proven convergence guarantees. This is a great challenge proving that the requirements for systems utilizing nonlinear solvers are always met. The research concerning efficient NLP solvers is also adapting to the current needs very quickly, focusing on a low memory cost and limiting to operations which can be efficiently performed on field-programmable arrays or other modern computational platforms. Although proximal methods have shown to be very effective solvers, learning-based solutions may overtake state-of-the-art MPC formulations in the future.

LIST OF PUBLICATIONS

The author has published numerous articles in topics involving Optimal Control for autonomous UAVs involving a coupled system of a quadcopter vehicle and a robotic arm manipulator designed for aerial grasping tasks, surrogate modeling methods for black-box validation of safety-critical systems, and derivative-free combinatorial and continuous optimization. The publications are split into papers accepted for publication at international conferences/journals and published articles.

Accepted for publication:

- **NOVÁK Jiří**, HANÁK Jiří a CHUDÝ Peter Weight-varying Model Predictive Control for Coupled Cyber-Physical Systems: Aerial Grasping Study. In: Machine Learning, Optimization, and Data Science. Springer-Nature Lecture Notes in Computer Science . Grasmere: Springer Nature Switzerland AG, 2024
- **NOVÁK Jiří**, HANÁK Jiří a CHUDÝ Peter Reliability-Based Control System Optimization in Uncertain Conditions. In: AIAA Aviation Forum, 2024.
- HANÁK Jiří, **NOVÁK Jiří** a CHUDÝ Peter Tactical Scenario Adaptation for Pilot Training. Accepted to: 2024 IEEE/AIAA 43th Digital Avionics Systems Conference (DASC).
- HANÁK Jiří, **NOVÁK Jiří** a CHUDÝ Peter Cognitive Modeling Approach for Generating Authentic Tactical Agent Behavior. Accepted to: 2024 IEEE/AIAA 43th Digital Avionics Systems Conference (DASC).
- **NOVÁK Jiří**, HANÁK Jiří and CHUDÝ Peter. Predictive Control Driven Tactical Maneuvering. Accepted to: ICAS 2024, Florence: International Council of the Aeronautical Sciences.
- **NOVÁK Jiří** a CHUDÝ Peter. Model Predictive Control Driven Aerial Grasping with Soft Operational Constraints. Accepted to: ICAS 2024, Florence: International Council of the Aeronautical Sciences.
- **NOVÁK Jiří** a CHUDÝ Peter. Hybrid Modeling Approach for Optimization Based Control of Multirotor Unmanned Aerial Vehicles. Accepted to: ICAS 2024, Florence: International Council of the Aeronautical Sciences.
- HANÁK Jiří, **NOVÁK Jiří**, CHUDÝ Peter a BEN-ASHER Joseph Z. The Cross-Entropy Method for Laser Defense Applications. Journal of Aerospace Information Systems, 2023, s. 14. ISSN 2327-3097.

Published articles:

- **NOVÁK Jiří** a CHUDÝ Peter. Dynamic Soaring in Uncertain Wind Conditions: Polynomial Chaos Expansion Approach. In: Machine Learning, Optimization, and Data Science. Springer-Nature Lecture Notes in Computer Science . Grasmere: Springer Nature Switzerland AG, 2023, s. 104-115. ISBN 978-3-031-53969-5.
- **NOVÁK Jiří** a CHUDÝ Peter. Surrogate Modeling of Optimal Control Based Collision Avoidance System for Multirotor Unmanned Aerial Vehicles. In: 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC) Proceedings. Barcelona: Institute of Electrical and Electronics Engineers, 2023, s. 1-7. ISBN 979-8-3503-3357-2.
- **NOVÁK Jiří** a CHUDÝ Peter. Unmanned Aerial Vehicle's Trajectory Optimization in Constrained Environments. In: ICAS PROCEEDINGS. Stockholm: International Council of the Aeronautical Sciences, 2022, s. 1-15. ISSN 2958-4647.

PROJECT PARTICIPATION

The author has participated in scientific/industrial projects as a member of the AEROWORKS research group at the Faculty of Information Technology, Brno University of Technology. During the project participation, the author was a part of the development team of a Sensor Fusion Toolbox and Mission Design and Optimization Tool. The author was part of the following projects:

- **Laser sensor for autonomous driving of trucks** project granted by Technology Agency of the Czech Republic in period 2021-2023 Project ID. CK02000106

  The aim of the project is to develop the innovative laser sensor (flashing LiDAR) for autonomous driving of trucks. The sensor will be developed according to automotive development and safety standards (ASIL, A-SPICE) and the development will cover laser optics, mechanical design, HW and SW. Together with the sensor we will also develop four functions - autonomous driving in traffic jam, automated safety moving off the truck, function of safety lane change maneuver and safety turn of maneuver. These functions will not use only data from new laser sensor, but we will implement a fusion with other sensors (radar, camera) and the functions will be implemented as SW packages for higher ECU in the truck. The sensor and the functions will be tested on the real trucks (test polygon / open road).

- **Nanoradar for autonomous driving of trucks and its industrialization 4.0** project granted by Technology Agency of the Czech Republic in period 2021-2023 Project ID. FW03010614

  The aim of the project is to research and develop an innovative nanoradar sensor for autonomous driving of trucks. The research and development will encompass respective fields in radar technology, mechanical design, HW and SW aspects, automation and control, machine learning, signal processing, validation and testing. Part of the radar development will contain research and development of two autonomous driving functions, which will be used for customer demonstration purposes. The radar itself will be designed considering the Industry 4.0 principles. A digital twin of the product, serving the virtual design of the manufacturing plant and a virtual adaptation of associated manufacturing processes, will be developed. The digital twin will allow for efficiency improvements in product customization, leading to time and cost savings throughout the implementation process.

- **Framework of key enabling technologies for safe and autonomous drones' applications** project granted by ECSEL Joint Undertaking in period of 2019 - 2022. Project ID. 8A19003, 826610

  UAVs/drones can perform air operations that manned aircrafts struggle with, and their use brings significant economic savings and environmental benefits whilst reducing the risk to human life. UAV-based service and product innovation, as driven by increased levels of connectivity and automation, is curtailed by the growing dependence on poorly interoperable proprietary technologies and the risks posed to people, to other vehicles and to property. The COMP4DRONES project complements SESAR JU efforts with particular focus on safe software and hardware drone architectures.

- **Artificial Intelligence Driven Autonomy** project granted by the National Competence Center for Aeronautics and Space funded by TACR in period of 2019 - 2020. Project ID. TN01000029

  Pilot training belongs to time-wise and financially demanding activities. Its concept reflects not only the technical and technological advancement in aviation, but also a high standard of social expectations. In context of advanced Cyber-Physical Systems, Machine Learning and Artificial Intelligence , modern pilot training represents a significant scientific, commercial and social challenge. The main project goal is to develop a set of software tools for AI driven autonomy in agents for modern synthetic pilot training platforms.

A REFERENCE FRAMES AND TRANSFORMATIONS

For the proper definition of physical quantities and their transformation, this thesis uses a number of reference frames introduced in this section. The used reference frames are adopted from the literature, and their respective dependencies are graphically shown in Figure A.1. They form a necessary background commonly used in the aerospace engineering literature. The commonly used coordinate frame transformations in the form of a rotation matrix and quaternion formulation are also mentioned.

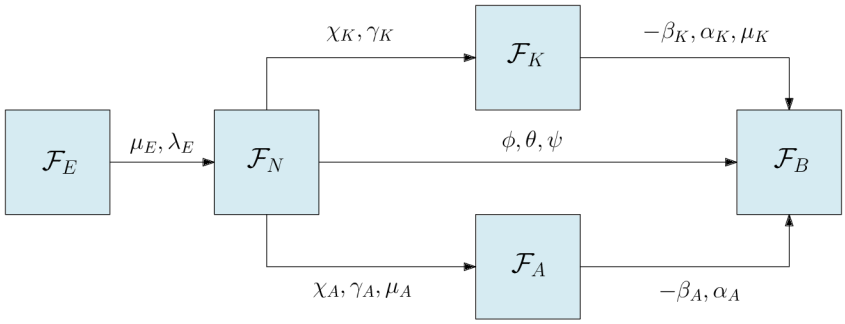

Figure A.1: Schematics of coordinate systems and their transformations.

EARTH-CENTERED, EARTH-FIXED FRAME

The Earth-Centered, Earth-Fixed (ECEF) frame is a cartesian coordinate frame referencing positional data on the Earth. This thesis uses the mathematical notation $\mathcal{F}_E = (x_E, y_E, z_E, O_E)$. The origin is located in the Earth's center, the x_E-axis points towards the Greenwich Meridian and the z_E-axis coincides with the Earth's rotational axis. In practice, the ECEF frame is used in an alternative formulation of World Geodetic System 1984 (WGS84), which is used for the Global Positioning System (GPS). This system defines the object's position using the geodetic latitude μ_E, geodetic longitude λ_E, and geodetic height h_E. The geodetic latitude is an angle measured between the zero meridian plane and the meridian plane of position point P. The geodetic latitude measures the angle between the Equatorial plane and the surface normal of the point P in the meridian plane of point P.

BODY-FIXED FRAME

The BFF is a reference frame that translates and rotates with the aircraft. It is commonly used to express body-related quantities such as forces and moments acting on the aircraft or state variables (velocities, acceleration, etc.). The frame denoted by $\mathcal{F}_B = (x_B, y_B, z_B, O_B)$ has its origin fixed in the aircraft reference point (commonly CoG). The x_B-axis points towards the aircraft nose, the y_B-axis points towards the right wing, and the z_B-axis points downwards

in the aircraft's plane of symmetry. Figure A.2 illustrates the frame with respect to the aircraft.

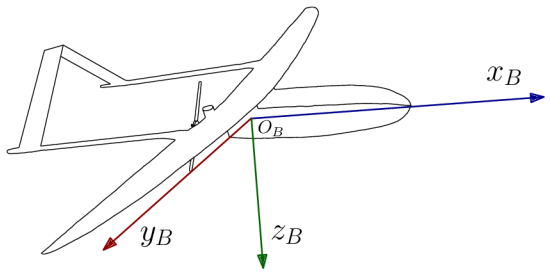

Figure A.2: Body-Fixed frame visualization.

AERODYNAMIC FRAME

The aerodynamic frame defines the motion and attitude of the aircraft according to aerodynamic flow. It is essential for expressing the aerodynamic forces known as lift, drag, and lateral force. The frame $\mathcal{F}_A = (x_A, y_A, z_A, O_A)$ has its origin in the aircraft's reference point. The frame 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 transformation from BFF to the aerodynamic frame is performed using AoA with notation α_A and sideslip angle β_A. Figure A.3 illustrates the aerodynamic frame and its characteristics.

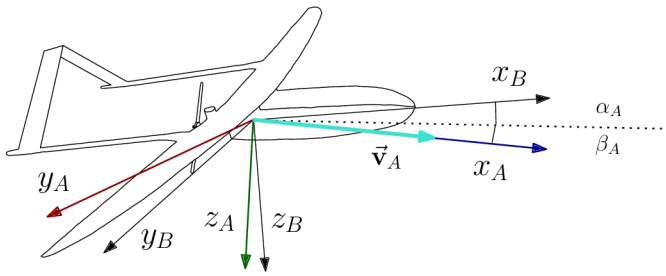

Figure A.3: Aerodynamic frame visualization.

KINEMATIC FRAME

The kinematic frame, sometimes referred to as the trajectory frame, is a frame of reference for the aircraft trajectory. The kinematic frame is also fixed in the aircraft reference point while rotating with respect to the kinematic velocity. The notation $\mathcal{F}_K = (x_K, y_K, z_K, O_K)$ used to describe the right-handed coordinate system defines the x_K-axis aligned to the kinematic velocity, and the z_K-axis pointing downwards in the symmetric plane of the aircraft. Figure A.4 shows the kinematic frame and its characteristics.

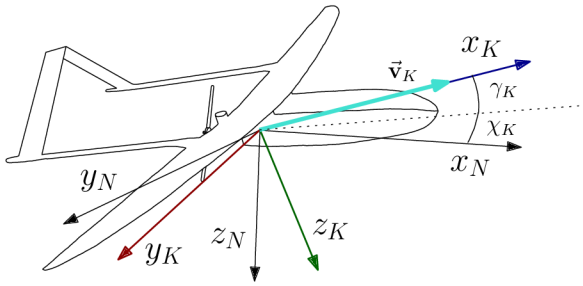

Figure A.4: Kinematic frame visualization.

NORTH-EAST-DOWN FRAME

The NED frame is defined to denote the orientation of the aircraft. It is fixed with the aircraft reference point and translates with the aircraft. The NED frame denoted by $\mathcal{F}_N = (x_N, y_N, z_N, O_N)$ has its x_N-axis pointing towards the geographic North and it is parallel to the local geoid surface. The y_N-axis points towards the geographic East and it is also parallel to the local geoid surface. The z_N-axis points downward and it is perpendicular to the local geoid surface. Figure A.5 illustrates the NED frame and the local tangent plane.

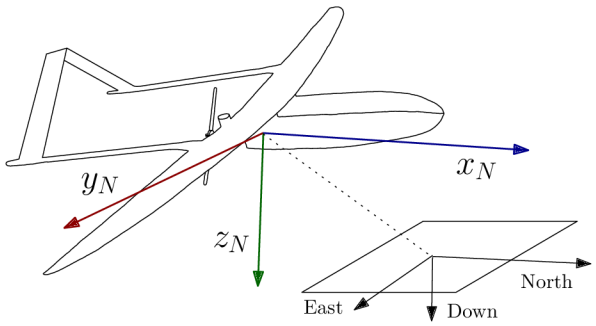

Figure A.5: North-East-Down frame visualization.

FRAME TRANSFORMATIONS

As was introduced in Figure A.1, all the mentioned coordinate frames can be transformed using a defined set of angles. The transformation matrices used for rotating the reference frames are a part of the Special Orthogonal Group (SO(3)). The orthonormality requirement has to be satisfied by the rotation matrix. Mathematically, the requirements form a set of valid rotations $\mathbf{R}_{X/Y} = \left\{ \mathbf{R}_{X/Y} \in \mathbb{R}^{3\times3} \mid \mathbf{R}_{X/Y}\mathbf{R}_{Y/X}^{T} = \mathbf{I}_3,\ \det \mathbf{R}_{X/Y} = 1 \right\}$. The first index refers to the output reference frame while the second index refers to the output frame. The transformation from the BFF to the local vertical frame is typically performed using Euler angles ϕ, θ, ψ. The transformation matrix can be expressed using equation (A.1).

$$\mathbf{R}_{E/B} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{A.1}$$

The same approach can be taken for the rest of the frame transformations, though some frames are transformed only by two rotations. For example, the transformation from BFF to the aerodynamic frame is computed by applying rotations $0, \alpha_A, -\beta_A$. The resulting transformation, as in equation (A.2) becomes:

$$\mathbf{R}_{A/B} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{A.2}$$

Unfortunately, the Euler angle representation suffers from numerical problems such as the presence of singularities. To represent a numerically robust rotation representation, the term DCM is used. It is typically realized by the unit quaternion representation. A unit quaternion is represented by the set $\mathbf{Q}_1 = \{\mathbf{q} \in \mathbb{R}^4 \mid \|\mathbf{q}\|_2 = 1\}$, where $\mathbf{q} = [q_0,\ iq_1,\ jq_2,\ kq_2] = [q_0,\ \mathbf{q}_v]$. Unit quaternions are a part of SO(3) and are used to represent valid orientations. The operator $\odot$ is used to define the quaternion multiplication as defined by equation (A.3).

$$\mathbf{q} \odot \mathbf{p} = \begin{bmatrix} q_0 p_0 - \mathbf{q}_v^T \mathbf{p}_v \\ (q_0 \mathbf{p}_v + \mathbf{q}_v \times \mathbf{p}) \end{bmatrix} \tag{A.3}$$

A vector $\mathbf{t}^Y$ expressed in Y reference frame can be transformed to the X frame using the defined multiplication and a quaternion conjugate $\mathbf{q}^c_{Y/X}$ as written in (A.4).

$$\begin{bmatrix} 0 \\ \mathbf{t}^Y \end{bmatrix} = \mathbf{q}_{Y/X} \odot \begin{bmatrix} 0 \\ \mathbf{t}^X \end{bmatrix} \odot \mathbf{q}^c_{Y/X} \tag{A.4}$$

This expression can be manipulated to form a convenient rotation matrix representation as written in equation (A.5).

$$\mathbf{R}_{A/B} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_1q_2) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_1q_2) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{A.5}$$

Such representation of rotation matrix benefits from a simple computation on a computer since there are no trigonometric functions to be evaluated. [11]

B NUMERICAL AND ANALYTICAL DIFFERENTIATION

Differentiation plays a pivotal role in many science and engineering applications. CPS have been strongly tied to dynamical systems theory, since real world applications are often based on time-evolving phenomena. The complex physics based or multi-scale physics-based dynamical systems have rarely been solved analytically. But, fortunately, the rise of digital computing has enabled numerical solutions to many engineering problems. The development of ANN was substantially dependent on optimization techniques and differentiation. A widespread adoption of Automatic Differentiation (AD) techniques enabled training ANN robustly with a reduced computational complexity. The importance of AD was even amplified with the emerging deep learning applications.

Starting from a basic definition of a derivative, let us assume a real-valued function $f : \mathbb{R} \to \mathbb{R}$. Its derivative at point a is defined by the limit (B.1).

$$\frac{df}{dt}(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \tag{B.1}$$

Let us now recall a set of mathematical objects using derivatives in context of vector-valued functions. Consider a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}$. The gradient denoted as $\nabla \mathbf{f}$ is defined by a vector (B.2).

$$\nabla \mathbf{f} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \frac{\partial \mathbf{f}}{\partial x_2} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix}^T \tag{B.2}$$

Considering the same vector-valued function, a matrix of all second partial derivatives with respect to all variables is called a Hessian $\nabla^2 \mathbf{f}$ and is defined by a matrix (B.3).

$$\nabla^2 \mathbf{f} = \begin{bmatrix} \frac{\partial^2 \mathbf{f}}{\partial x_1^2} & \frac{\partial^2 \mathbf{f}}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 \mathbf{f}}{\partial x_1 \partial x_n} \\ \frac{\partial^2 \mathbf{f}}{\partial x_2 \partial x_1} & \frac{\partial^2 \mathbf{f}}{\partial x_2^2} & & \frac{\partial^2 \mathbf{f}}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 \mathbf{f}}{\partial x_n \partial x_1} & \frac{\partial^2 \mathbf{f}}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 \mathbf{f}}{\partial x_n^2} \end{bmatrix} \tag{B.3}$$

Finally, assuming a vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, the Jacobian $\nabla \mathbf{f}$ of this function is defined by a matrix (B.4).

$$\nabla \mathbf{f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & & \frac{\partial f_2}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{B.4}$$

Although there are many numerical schemes to approximate derivatives numerically, simple techniques of forward differencing, backward differencing and central differencing are the most used techniques for derivative approximation due to their simple implementation

utilizing the definition of derivative from (B.1). However, some applications including optimization algorithms may find numerical derivatives inefficient, especially for Hessian matrix computation, and prone to numerical errors. In contrast, AD is based on the decomposition of differentials provided by the chain rule. It is, therefore, distinct from symbolic differentiation founded on the principles of computer algebra. Figure B.1 shows an abstraction diagram putting AD and symbolic differentiation into context.

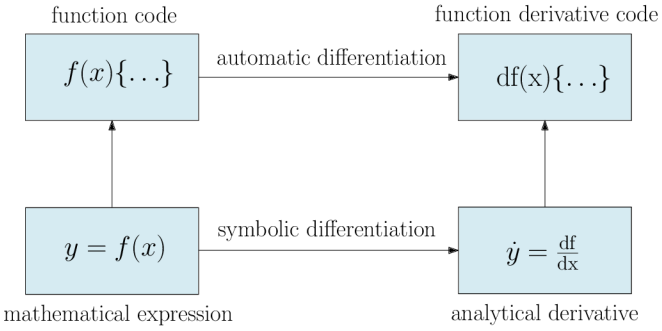

Figure B.1: Automatic and symbolic differentiation.

The chain rule is a foundation of all AD systems. Having two functions $f(x)$ and $g(x)$, the derivative of their composition is given by equation (B.5).

$$\frac{d\,[f(g)]}{dx}(x) = \frac{df}{dx}(g(x))\frac{dg}{dx}(x) \tag{B.5}$$

Due to a successive application of the chain rule, AD is able to compute derivatives in a fast and memory-efficient manner. There are two techniques to mention. Forward-mode AD fixes the variable which is being differentiated to and computes the derivative recursively using equation (B.6).

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial w_{n-1}}\frac{\partial w_{n-1}}{\partial x} = \frac{\partial f}{\partial w_{n-1}}\left(\frac{\partial w_{n-1}}{\partial w_{n-2}}\frac{\partial w_{n-2}}{\partial x}\right) = \ldots \tag{B.6}$$

The rule is often generalized to functions of multiple variables, enabling to compute Jacobian and Hessian functions efficiently. The functions $w_0, w_1, \ldots w_n$ are the composite functions, such that $w_0 = x$ and $w_n = f$. In contrast, in reverse-mode AD, the dependent variable is differentiated such that the derivatives with respect to all subexpressions are evaluated as denoted by equation (B.7).

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial w_1}\frac{\partial w_1}{\partial x} = \left(\frac{\partial f}{\partial w_2}\frac{\partial w_2}{\partial w_1}\right)\frac{\partial w_1}{\partial x} = \ldots \tag{B.7}$$

Forward- mode differentiation is more efficient since it requires n sweeps of the chain rule while the reverse-mode differentiation requires m sweeps.

Most libraries focused on ANN and ML in general such as Tensorflow [1], Pytorch [82], or JAX [37] are implemented with AD at their cores. The topic of AD techniques was introduced in a rather simple manner. An interested reader is advised to follow the literature on this topic such as [7] or [40].

C ORTHOGONAL POLYNOMIALS

In the context of parameter uncertainty quantification and pseudospectral transcription methods discussed in this thesis, a short review on orthogonal polynomials and their properties is presented. The strength of orthogonal polynomials is often associated with the useful mathematical properties of their roots. The class of orthogonal polynomials forms a basis for the Generalized Polynomial Chaos (gPC) expansion method for uncertainty quantification as well as the Pseudospectral collocation methods for optimal control, or solving some other types of ODEs in general. The hierarchical organization of orthogonal polynomials is provided by the Askey scheme [5]. In practice, orthogonal polynomials are often easy to generate using the Gram-Schmidt orthonormalization. More comprehensive reviews and mathematical proofs are given in [57], [39], and [126].

The definition of orthogonal polynomial is given by the Favard's Theorem (Shohat–Favard theorem) published in 1935: Let $A^{(m)}$, $B^{(m)}$ and $C^{(m)}$ be arbitrary sequences of real numbers and let the set $\{\phi^{(m)}(\theta)\}$ be defined by the recurrence relation (C.1), such that $\phi^{(0)}(\theta) = 1$ and $\phi^{(-1)}(\theta) = 0$.

$$\phi^{(m+1)}(\theta) = \left[A^{(m)}(\theta) + B^{(m)}\right]\phi^{(m)}(\theta) - C^{(m)}\phi^{(m-1)}(\theta), \quad m \geq 1. \tag{C.1}$$

Then the polynomials $\{\phi^{(m)}(\theta)\}$ with m being the polynomial degree, the relation (C.1) are the 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. In the context of this thesis, polynomials satisfying this definition and being orthogonal to specific probability distributions are introduced. Let us now introduce Hermite and Legendre polynomials used for uncertainty quantification of Gaussian distributed and Uniform distributed parameters. The classic orthogonal polynomials also include Laguerre and Jacobi, Gegenbauer, and Chebyshev polynomials.

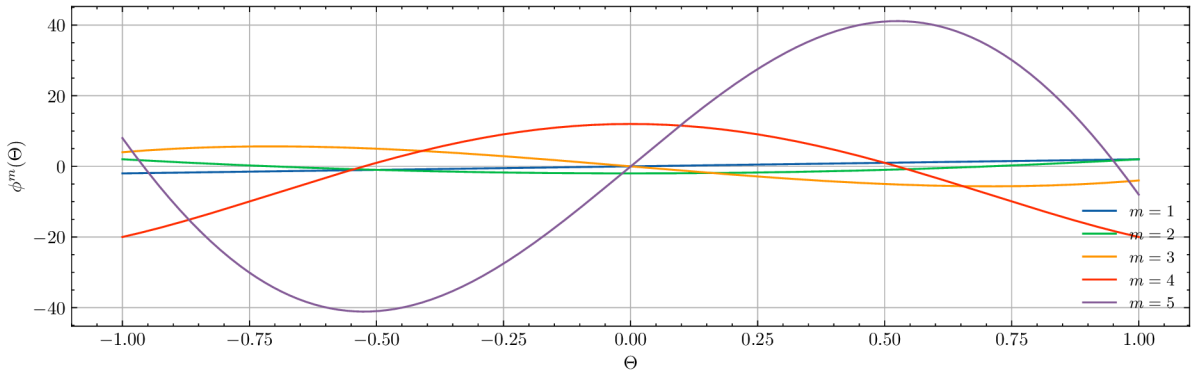

Figure C.1: Hermite polynomials for varying degrees on a unit domain.

Hermite polynomial basis satisfies a definition such that a recurrence sequence given by equation (C.2) is satisfied. Examples of Hermite polynomials using varying degree m are shown in Figure C.1.

$$\phi^{(m+1)}(\theta) = \theta\phi^{(m)}(\theta) - m\phi^{(m-1)}(\theta), \quad m \geq 0. \tag{C.2}$$

The orthogonality relation associated with Hermite polynomials is given by (C.3)

$$\int_{-\infty}^{\infty} \phi^{(n)}(\theta)\phi^{(m)}(\theta)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\theta^2}{2}\right)d\theta = m!\delta_{mn} \tag{C.3}$$

From the orthogonality relation, it can be seen that the Hermite polynomials are orthogonal with respect to the Probability Density Function (PDF) $\rho_\theta(\theta) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\theta^2}{2}\right)$ and their normalizing constant can be derived as $\left[h^{(m)}\right]^2 = m!$.

In a similar manner, the Legendre polynomials are defined by a recurrence sequence given by equation (C.4).

$$\phi^{(m+1)}(\theta) = \frac{2m+1}{m+1}\theta\phi^{(m)}(\theta) - \frac{m}{n+1}\phi^{(m-1)}(\theta), \quad m \geq 0. \tag{C.4}$$

The orthogonality relation associated with Legendre polynomials is given by (C.5).

$$\int_{-1}^{1} \phi^{(n)}(\theta)\phi^{(m)}(\theta)\frac{1}{\sqrt{2\pi}}d\theta = \frac{2}{2m+1}\delta_{mn} \tag{C.5}$$

Legendre polynomials are orthogonal with respect to PDF of $\rho_\theta = 1$ with the normalizing constant $\left[h^{(m)}\right]^2 = \frac{2}{2m+1}$.

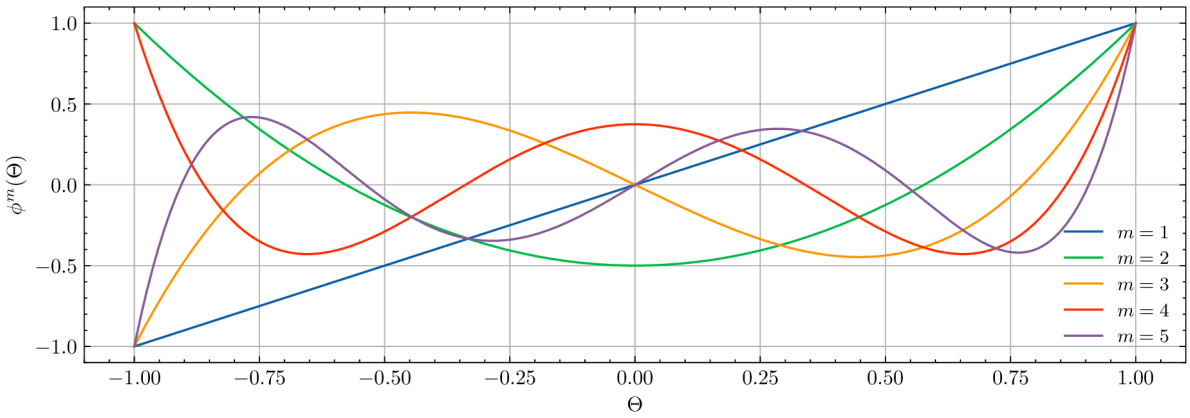

Figure C.2: Legendre polynomials for varying degrees on a unit domain.

LIST OF HARDWARE COMPONENTS

D

Pixhawk 6x flight controller

The Pixhawk® is a high-performance flight controller based on the Pixhawk® FMUv6X Open Standard and the Pixhawk® Autopilot Bus Standard. The controller is composed of a baseboard and a flight computer as shown in Figure D.1. Triple redundancy of on-board IMU and double redundancy of barometer sensors are one of the key advantages of this integrated solution. Apart from the preinstalled PX4 firmware, a hardware support of MATLAB® allows for the integration of custom flight control and path follower algorithms.

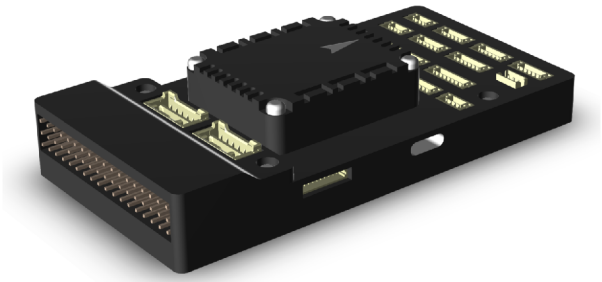

Figure D.1: Flight controller 3D representation.

Following the technical specification of the controller, Table D.1 lists an integrated set of processors and sensors onboard the flight controller. The electrical and mechanical specifications are given in Table D.2.

Table D.1: Processors and sensors of the flight controller.

FMU Processor	STM32H753, 32 Bit Arm® Cortex®-M7, 480MHz, 2MB FM, 1MB RAM
IO Processor	STM32F103, 32 Bit Arm® Cortex®-M3, 72MHz, 64KB SRAM
IMU 1	ICM-20649 or BMI088
IMU 2	ICM-42688-P
IMU 3	ICM-42670-P
Magnetometer	BMM150
Barometer	2x BMP388

The baseboard includes interfaces to 16 PWM servo outputs, 4 general purpose serial ports, R/C input, 2 GPS ports, a single I2C port, Ethernet port and many others. The operating temperature of the flight controller ranges from $-40°$ C to $85°$ C.

Table D.2: Electrical and mechanical specifications of the flight controller.

Max input voltage	6V	FCM dim.	38.8 x 31.8 x 14.6mm
USB Power Input	4.75 5.25V	Base dim.	52.4 x 103.4 x 16.7mm
Servo Rail Input	0 36V	FCM weight	23 g
Output current limiter	1.5A	Base weight	51 g

Raspberry Pi 5 companion computer

The Raspberry pi 5 computer is used as a companion computer to the Pixhawk controller. Computationally heavy tasks utilizing computer vision algorithms or solving the OCP for MPC of the drone are performed on this computer. It features a Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with Cryptographic Extension, 512KB per-core L2 caches, and a 2MB shared L3 cache processor. Table D.3 contains basic technical specifications of the companion computer.

Table D.3: Raspberry Pi camera 3 technical specifications.

Power	5V/5A DC	Wi-Fi	2.4 GHz and 5.0 GHz 802.11ac
Processor	Broadcom BCM2712	Memory	LPDDR4X-4267 SDRAM 8 GB
Graphics	VideoCore VII	Frequency	2.4 GHz
Storage	micro SD card	Dimensions	86 mm × 56 mm × 16 mm

Raspberry Pi camera 3

The Raspberry pi camera 3 module is a compact 12MP camera sensor with autofocus. It features a high dynamic range mode and can record video at 50 fps. This project uses the camera to detect a reference position without the use of GPS and relative pose with respect to the commonly used marker such as AprilTag. The marker detection algorithm is running on the Raspberry pi 5 companion computer. Table D.4 contains basic technical specifications of the camera module. The camera is based on a Back-illuminated, stacked CMOS 12-megapixel Sony IMX708 image sensor.

Table D.4: Raspberry Pi camera 3 technical specifications.

Resolution	11.9 MP	Sensor size	7.4mm sensor diagonal
Horizontal/vertical	4608 × 2592 px	Dimensions	25 × 24 × 11.5mm
Diagonal field of view	75°	Pixel size	1.4µm × 1.4 µm

Waveshare SC15 programmable servo

SC15 Programmable Serial Bus Servo is a high torque servo suitable for robotic applications. It has a low operating noise, high mechanical performance, and stability. In this project, a servo is used to move a tube connected to a water pump such that the drone can transfer liquid from its container to a target location. Table D.5 contains basic technical specifications of the servo.

Table D.5: Waveshare SC15 programmable servo technical specifications.

Torque 8.4 V	17 kg · cm	Resolution	180° / 1024
Torque 6.0 V	15 kg · cm	Voltage range	4.8-8.4V
Torque 4.8 V	14 kg · cm	Baudrate	38400bps ~ 1Mbps
Idling current	200 mA	Stalling current	1700 mA

SiK Telemetry Radio V3

Holybro SiK Telemetry Radio is a small open-source radio platform with compatibility to Pixhawk® flight controllers. It is used to establish a connection between the autopilot and the ground station. Table D.6 contains basic technical specifications of the telemetry radio.

Table D.6: SiK Telemetry Radio V3 technical specifications.

Output power	100 mW	Dimension	28 x 53 x 10.7mm
Frequency	433 MHz	Supply voltage	5V DC
Transmit current	100 mA at 20dBm	Receive current	25 mA

Zorro Radio Controller (M2)

The radio controller Zorro M2 is used for the testing purposes of the flight controller and safety measure allowing the user to override the autonomous flight regime and get manual control of the drone. Table D.7 contains basic technical specifications of the radio controller.

Table D.7: Zorro Radio Controller technical specifications.

Frequency	2.400GHz-2.480GHz	Dimension	170 x 159 x 108mm
Operational current	160mA at 8.4V	Antenna gain	Folding 2db
Operational voltage	6.6-8.4v DC	RF power	100mW
Protocol	ELRS	Weight	350 g

RP1/RP2 ExpressLRS 2.4ghz Nano Receiver

The RP1/RP2 ExpressLRS receiver is connected to the Pixhawk® 6x and used to communicate with the Zorro radio controller. The RadioMaster RP series receivers have a built-in temperature-compensated crystal oscillator for a higher reliability and stability of connection. Table D.8 contains basic technical specifications of the used nano receiver.

Table D.8: RP1/RP2 ExpressLRS 2.4ghz Nano Receiver technical specifications.

Antenna	65mm 2.4ghz T	Frequency Range	2.404 - 2.479 Ghz
Dimension	13 x 11 x 3mm	Weight	2.2 g
Max. refresh rate	500Hz	RF power	10mW

PM02D Power Module

A digital power module such as Holybro PM02D serves as a power regulation module, which also provides information on the voltage and current supplied to the flight management unit and the actuators/motors. Table D.9 contains basic technical specifications of the used power module. The module is designed for Pixhawk® FMUv5X and FMUv6X open standard.

Table D.9: PM02D power module technical specifications.

Max input voltage	36V	Max current	120A
Rated current	60A	Communication protocol	I2C
Regulator outputs	5.2V and 3A max	Weight	20g

BLHeli S ESC 20A

Electronic Speed Controller (ESC) is used to drive the motors and regulate their angular velocities. Each motor of the drone has its own ESC. Table D.10 contains basic technical specifications of the ESC. It is equipped with a high-frequency driver enabling smooth transitions of the controller. The ESCs are hidden in the drone's airframe.

Table D.10: BLHeli S ESC 20A technical specifications.

Current	20A	Maximum frequency	50 MHz
Burst Current(10S)	30A	Maximum speed	500k eRPM

Holybro 2216 KV920 Motor

Holybro 2216 KV920 is a BLDC motor with a maximum thrust of 1332 g with the supplied 10x4.5 propeller. Table D.11 contains basic technical specifications of the motor. The term KV

refers to RPM that is generated when 1 V is applied at no load. The KV rating is the ratio of the motor's unloaded RPM to the peak voltage.

Table D.11: Holybro 2216 KV920 motor technical specifications.

KV	920	Maximum power	272 W
Idle current(10V)	0.8A	Rated voltage	16 V
Peak current	17A	Internal resistance	115 ± 10 mΩ
Weight	64.2 g	Max operating temperature	80°

Power Distribution Board (PDB) for PM02D Power Module

The power distribution board for PM02D enables the power supply to all the required motors/servos or the companion computer. This power distribution board comes with XT30 connectors pre-soldered, which are connected directly to each motor's ESC. Table D.12 contains basic technical specifications of the power distribution board.

Table D.12: Power Distribution Board technical specifications.

Rated current	60A	Max current	120A
Battery connector	XT60	Dimensions	50x50mm

Gens Ace Battery LiPo Bashing 4S1P 60C EC5 14.8 V 5000 mAh

The Gens Ace Battery LiPo 5000 mAh is a high-capacity battery designed for RC models. Table D.13 contains basic technical specifications of the battery. Lithium-ion polymer (LiPo) batteries are known for their high specific energy, thus being suitable for applications where weight is an important factor.

Table D.13: Gens Ace Battery LiPo technical specifications.

Capacity	5000mAh	Dimensions	136x42x34mm
Voltage	14.8 V	Weight	437 g
C rate	60C	Discharge plug	EC5

Peristaltic mini water pump

A low pressure peristaltic water pump is used for liquid transfer with the possibility of the change of flow direction. Peristaltic pumps also known as roller pumps work by compression of a flexible tube stretched along the circumference of the rotary roller. It allows for in-flight liquid transfer from or to the vehicle as it has a satisfactory suction lift. Table D.14 contains basic technical specifications of the used water pump.

Table D.14: Peristaltic mini water pump technical specifications.

Rated voltage	5V DC	Suction lift	$\leq$ 1.5 m
No load current	0.3A	Flow rate	$\leq$ 48 ml/min
Suitable voltage	2V-6V DC	Weight	38 g

H-bridge module L9110S

The L9110S 2-Channel motor driver module is a compact board that can be used to drive small motors. A PWM is used to control the speed of a motor, and a digital output is used to change its direction. In this project, it is used to drive a peristaltic mini water pump and enable the change of flow direction of the liquid to or from the onboard container. Table D.15 contains basic technical specifications of the used H-bridge.

Table D.15: H-bridge module technical specifications.

Input voltage range	2.5-12V DC	PCB Size	29.2 x 23mm
Output current	800 mA	Motor control chip	L9110

BIBLIOGRAPHY

[1] Martín Abadi et al. *TensorFlow: A system for large-scale machine learning*. 2016. DOI: https://doi.org/10.48550/arXiv.1605.08695.

[2] Badgwell T. A. Qin J. S. Rawlings J. B. Wright S. J. Allgöwer F. "Nonlinear Predictive Control and Moving Horizon Estimation - An Introductory Overview." In: *Advances in Control*. Ed. by Paul M. Frank. London: Springer London, 1999, pp. 391–449. ISBN: 978-1-4471-0853-5. DOI: https://doi.org/10.1007/978-1-4471-0853-5_19.

[3] Erling Andersen and Knud Andersen. "The Mosek Interior Point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm." In: 33 (Jan. 1999). DOI: 10.1007/978-1-4757-3216-0_8.

[4] Joel Andersson et al. "CasADi: a software framework for nonlinear optimization and optimal control." In: *Mathematical Programming Computation* 11 (July 2018). DOI: 10.1007/s12532-018-0139-4.

[5] George E. Andrews and Richard Askey. "Classical orthogonal polynomials." In: *Polynômes Orthogonaux et Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 36–62. ISBN: 978-3-540-39743-4.

[6] Christophe Andrieu, Arnaud Doucet, and R. Holenstein. "Particle Markov chain Monte Carlo methods (with discussion)." In: *Journal of the Royal Statistical Society, Series B* 72 (Jan. 2010), pp. 1–33. DOI: https://doi.org/10.1111/j.1467-9868.2009.00736.x.

[7] Atilim Baydin et al. "Automatic differentiation in machine learning: A survey." In: *Journal of Machine Learning Research* 18 (Apr. 2018), pp. 1–43. DOI: https://doi.org/10.48550/arXiv.1502.05767.

[8] Joseph Z Ben-Asher. *Optimal control theory with aerospace applications*. eng. AIAA education series. Reston, VA: American Institute od Aronautics and Astronautic, 2010. ISBN: 978-1-60086-732-3. DOI: 10.2514/4.867347.

[9] Mohamed Ben-Daya. "Failure Mode and Effect Analysis." In: *Handbook of Maintenance Management and Engineering*. Ed. by Mohamed Ben-Daya et al. London: Springer London, 2009, pp. 75–90. ISBN: 978-1-84882-472-0. DOI: 10.1007/978-1-84882-472-0_4.

[10] Dimitri P. Bertsekas. *Dynamic programming and optimal control*. eng. Massachusetts: Athena Scientific, 2000. ISBN: 1-886529-09-4. DOI: 10.2307/3010291.

[11] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming, 2nd ed*. eng. 2nd ed. Advances in Design and Control 19. Society for Industrial and Applied Mathematics, 2010. ISBN: 0898716888. DOI: 10.1137/1.9780898718577.

[12] Thomas Bierling et al. "Comparative Analysis of MRAC Architectures in a Unified Framework." In: Aug. 2010. ISBN: 978-1-60086-962-4. DOI: 10.2514/6.2010-7536.

[13] Christopher M. Bishop. *Pattern recognition and machine learning*. eng. Information science and statistics. New York: Springer Science + Business Media, 2006. ISBN: 978-1-4939-3843-8. DOI: https://doi.org/10.1007/978-0-387-45528-0.

[14] Pieter-Tjerk Boer et al. "A Tutorial on the Cross-Entropy Method." In: *Annals of Operations Research* 134 (Feb. 2005), pp. 19–67. DOI: 10.1007/s10479-005-5724-z.

[15] Paul Boggs and Jon Tolle. "Sequential Quadratic Programming." In: *Acta Numerica* 4 (Jan. 1995), pp. 1–51. DOI: 10.1017/S0962492900002518.

[16] Steven L. Brunton. *Data-driven science and engineering : machine learning, dynamical systems, and control*. eng. First published. Cambridge: Cambridge University Press, 2019. ISBN: 978-1-108-42209-3. DOI: https://doi.org/10.1017/9781108380690.

[17] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems." In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[18] Fabrizio Caccavale et al. "Adaptive control for UAVs equipped with a robotic arm." In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 19 (Jan. 2014), pp. 11049–11054. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.00790.

[19] Alberto Cavallo et al. "Experimental Comparison of Sensor Fusion Algorithms for Attitude Estimation." eng. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 7585–7591. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.01173.

[20] François Chaumette, Seth Hutchinson, and Peter Corke. "Visual Servoing." In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Cham: Springer International Publishing, 2016, pp. 841–866. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_34.

[21] Kenan Cole and Adam Wickenheiser. "Spatio-Temporal Wind Modeling for UAV Simulations." eng. In: (2019). DOI: https://doi.org/10.48550/arXiv.1905.09954.

[22] SAE On-Road Automated Vehicle Standards Committee et al. "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems." In: *SAE Standard J* 3016 (2014), p. 1.

[23] Michael V Cook. *Flight dynamics principles : A linear systems approach to aircraft stability and control*. eng. 2nd ed. Elsevier aerospace engineering series. Oxford: Butterworth-Heinemann, 2008. ISBN: 978-0-7506-6927-6. DOI: https://doi.org/10.1016/C2010-0-65889-5.

[24] Andre Costa, Owen Dafydd Jones, and Dirk Kroese. "Convergence properties of the cross-entropy method for discrete optimization." In: *Operations Research Letters* 35.5 (2007), pp. 573–580. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2006.11.005.

[25] Steven Diamond and Stephen Boyd. *CVXPY: A Python-Embedded Modeling Language for Convex Optimization*. 2016. arXiv: 1603.00943.

[26] Alexander Domahidi, Eric Chu, and Stephen Boyd. "ECOS: An SOCP solver for embedded systems." In: July 2013, pp. 3071–3076. DOI: 10.23919/ECC.2013.6669541.

[27] John Doyle. "Guaranteed margins for LQG regulators." eng. In: *IEEE transactions on automatic control* 23.4 (1978), pp. 756–757. ISSN: 0018-9286.

[28] Jean-Guillaume Durand, Arthur Dubois, and Robert J. Moss. "Formal and Practical Elements for the Certification of Machine Learning Systems." In: *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*. 2023, pp. 1–10. DOI: 10.1109/DASC58513.2023.10311201.

[29] EASA. "Artificial Intelligence Roadmap: A human-centric approach to AI in aviation." In: *European Union Aviation Safety Agency, Tech. Rep.1.0* (Feb. 2020).

[30] Jos Elfring, Elena Torta, and René van de Molengraft. "Particle Filters: A Hands-On Tutorial." In: *Sensors* 21.2 (2021). ISSN: 1424-8220. DOI: 10.3390/s21020438.

[31] Stefan Emeis and Matthias Turk. "Comparison of Logarithmic Wind Profiles and Power Law Wind Profiles and their Applicability for Offshore Wind Profiles." In: Feb. 2007, pp. 61–64. ISBN: 978-3-540-33865-9. DOI: 10.1007/978-3-540-33866-6_11.

[32] FAA. "Verification of Adaptive Systems." In: *Federal Aviation Administration, Tech. Rep. Final report* (Apr. 2016).

[33] Davide Falanga et al. "PAMPC: Perception-Aware Model Predictive Control for Quadrotors." In: Apr. 2018. DOI: 10.1109/IROS.2018.8593739.

[34] Rolf Findeisen and Frank Allgöwer. "An introduction to nonlinear model predictive control." In: *21st Benelux Meeting on Systems and Control* (Jan. 2002).

[35] Rebecca Foust, Soon-Jo Chung, and Fred Y. Hadaegh. "Optimal Guidance and Control with Nonlinear Dynamics Using Sequential Convex Programming." In: *Journal of Guidance Control and Dynamics* 43 (2020), pp. 633–644. DOI: https://doi.org/10.2514/1.G004590.

[36] Gene F. Franklin. *Feedback control of dynamic systems*. eng. Eight edition; Global edition. NY: Pearson, 2020. ISBN: 978-1-292-27452-2.

[37] Roy Frostig, Matthew Johnson, and Chris Leary. "Compiling machine learning programs via high-level tracing." In: 2018.

[38] Joseph E. Gaudio et al. "Connections Between Adaptive Control and Optimization in Machine Learning." In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019. DOI: 10.1109/cdc40024.2019.9029197.

[39] Walter Gautschi. "On Generating Orthogonal Polynomials." In: *SIAM Journal on Scientific and Statistical Computing* 3.3 (1982), pp. 289–317. DOI: 10.1137/0903018.

[40] Assefaw Gebremedhin et al. "Efficient Computation of Sparse Hessians Using Coloring and Automatic Differentiation." In: *INFORMS Journal on Computing* 21 (May 2009), pp. 209–223. DOI: 10.1287/ijoc.1080.0286.

[41] Weibin Gu et al. "UAV Model-based Flight Control with Artificial Neural Networks: A Survey." In: *J. Intell. Robotic Syst.* 100 (2020), pp. 1469–1491. DOI: https://doi.org/10.1007/s10846-020-01227-8.

[42] Ahmad Hammoudeh. *A Concise Introduction to Reinforcement Learning*. Feb. 2018. DOI: 10.13140/RG.2.2.31027.53285.

[43] Drew Hanover et al. "Performance, Precision, and Payloads: Adaptive Nonlinear MPC for Quadrotors." In: *IEEE Robotics and Automation Letters* 7.2 (2022), 690–697. ISSN: 2377-3774. DOI: 10.1109/lra.2021.3131690.

[44] Anton J Haug and Anton J Haug. *Bayesian estimation and tracking: a practical guide.* eng. Hoboken: WILEY, 2012. ISBN: 9780470621707. DOI: 10.1002/9781118287798.

[45] Naira Hovakimyan et al. "L1 adaptive control for safety-critical systems." In: *IEEE Control Systems Magazine* 31.5 (2011), pp. 54–104. DOI: 10.1109/MCS.2011.941961.

[46] Geoffrey Huntington. "Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control." In: (Sept. 2008).

[47] Tom Z. Jiahao, M. Ani Hsieh, and Eric Forgoston. "Knowledge-based learning of nonlinear dynamics and chaos." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.11 (2021), p. 111101. DOI: 10.1063/5.0065617.

[48] Bailun Jiang et al. "Neural Network Based Model Predictive Control for a Quadrotor UAV." In: *Aerospace* 9.8 (2022), p. 460. DOI: 10.3390/aerospace9080460.

[49] Ping Jiang, Qi Zhou, and Xinyu Shao. *Surrogate Model-Based Engineering Design and Optimization*. Jan. 2020. ISBN: 978-981-15-0730-4. DOI: 10.1007/978-981-15-0731-1.

[50] Xue-Bo Jin and Yuan Gao. *Multi-Sensor Information Fusion*. eng. MDPI - Multidisciplinary Digital Publishing Institute, 2020. ISBN: 9783039283033. DOI: https://doi.org/10.3390/books978-3-03928-303-3.

[51] Xue-Bo Jin et al. "The new trend of state estimation: From model-driven to hybrid-driven methods." eng. In: *Sensors (Basel, Switzerland)* 21.6 (2021), pp. 1–25. ISSN: 1424-8220. DOI: https://doi.org/10.3390/s21062085.

[52] Matthias Katzfuss, Jonathan Stroud, and Christopher Wikle. "Understanding the Ensemble Kalman Filter." In: *The American Statistician* 70 (Feb. 2016). DOI: 10.1080/00031305.2016.1141709.

[53] Naghman Khan. *Drone Flight Simulation*. 2024. URL: https://www.simscale.com/blog/drone-flight-simulation/ (visited on 02/12/2024).

[54] Waqas Khan and Meyer Nahon. "A propeller model for general forward flight conditions." eng. In: *International journal of intelligent unmanned systems* 3.2/3 (2015), pp. 72–92. ISSN: 2049-6427. DOI: https://doi.org/10.1108/IJIUS-06-2015-0007.

[55] John C. Knight. "Safety critical systems: challenges and directions." In: *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*. 2002, pp. 547–550.

[56] Omar Knio and Olivier Le Maître. "Uncertainty propagation in CFD using polynomial chaos decomposition." In: *Fluid Dynamics Research* 38 (Sept. 2006), pp. 616–640. DOI: 10.1016/j.fluiddyn.2005.12.003.

[57] Tom H. Koornwinder. "Orthogonal Polynomials." In: *Texts & Monographs in Symbolic Computation*. Springer Vienna, 2013, pp. 145–170. DOI: 10.1007/978-3-7091-1616-6_6.

[58] Dimche Kostadinov and Davide Scaramuzza. "Online Weight-adaptive Nonlinear Model Predictive Control." In: Nov. 2020. DOI: 10.1109/IROS45743.2020.9341495.

[59] Denis Kotarski. "Generalized Control Allocation Scheme for Multirotor Type of UAVs." eng. In: IntechOpen, 2018. ISBN: 1789232848.

[60] Paul Kremer et al. "TRIGGER: A Lightweight Universal Jamming Gripper for Aerial Grasping." In: *IEEE Access* 11 (2023), 50098–50115. ISSN: 2169-3536. DOI: 10.1109/access.2023.3276486.

[61] Chris Lacor and Eric Savin. *General Introduction to Polynomial Chaos and Collocation Methods*. Jan. 2018.

[62] Ioan Doré Landau et al. "Introduction to Adaptive Control." In: *Adaptive Control: Algorithms, Analysis and Applications*. London: Springer London, 2011, pp. 1–33. ISBN: 978-0-85729-664-1. DOI: 10.1007/978-0-85729-664-1_1.

[63] Wenbo Li et al. "Trajectory Optimization with Complex Obstacle Avoidance Constraints via Homotopy Network Sequential Convex Programming." In: *Aerospace* 9.11 (2022). ISSN: 2226-4310. DOI: 10.3390/aerospace9110720.

[64] Xiang Li et al. "Aircraft Robust Trajectory Optimization Using Nonintrusive Polynomial Chaos." In: *Journal of Aircraft* 51 (Sept. 2014), pp. 1592–1603. DOI: 10.2514/1.C032474.

[65] Lennart Ljung. "Perspectives on system identification." In: *Annual Reviews in Control* 34.1 (2010), pp. 1–12. ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2009.12.001.

[66] Danylo Malyuta et al. "Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently." In: *IEEE Control Systems Magazine* 42.5 (2022), pp. 40–113. DOI: 10.1109/MCS.2022.3187542.

[67] Yuanqi Mao et al. "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems." In: *arXiv: Optimization and Control* (2018). DOI: https://doi.org/10.48550/arXiv.1804.06539.

[68] Fanyu Meng et al. "Analysis for global characteristics of Lyapunov exponents in vehicle plane motion system." In: *Scientific Reports* 12 (June 2022). DOI: 10.1038/s41598-022-13411-x.

[69] Rudolph Merwe, Eric Wan, and Simon Julier. "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion–Applications to Integrated Navigation." In: *Proceedings of the AIAA Guidance, Navigation and Control Conference* 3 (Aug. 2004). DOI: 10.2514/6.2004-5120.

[70] T Mesbahi et al. "Li-Ion Battery Emulator for Electric Vehicle Applications." eng. In: *2013 IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE, 2013, pp. 1–8. ISBN: 1479907200.

[71] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2019.

[72] Simon Muntwiler, Kim Peter Wabersich, and Melanie Nicole Zeilinger. "Learning-based Moving Horizon Estimation through Differentiable Convex Optimization Layers." In: *ArXiv* (2021). DOI: https://doi.org/10.48550/arXiv.2109.03962.

[73] Christopher Nemeth, Paul Fearnhead, and Lyudmila Mihaylova. "Sequential Monte Carlo Methods for State and Parameter Estimation in Abruptly Changing Environments." In: *Signal Processing, IEEE Transactions on* 62 (Mar. 2014), pp. 1245–1255. DOI: 10.1109/TSP.2013.2296278.

[74] Huan Nguyen et al. "Model Predictive Control for Micro Aerial Vehicles: A Survey." In: *2021 European Control Conference (ECC)* (2020), pp. 1556–1563. DOI: https://doi.org/10.48550/arXiv.2011.11104.

[75] Jiří Novák and Peter Chudý. "Surrogate Modeling of Optimal Control Based Collision Avoidance System for Multirotor Unmanned Aerial Vehicles." In: *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC) Proceedings*. Barcelona, ES: Institute of Electrical and Electronics Engineers, 2023, pp. 1–7. ISBN: 979-8-3503-3357-2. DOI: 10.1109/DASC58513.2023.10311265.

[76] Jiří Novák and Peter Chudý. "Unmanned Aerial Vehicle's Trajectory Optimization in Constrained Environments." In: *ICAS PROCEEDINGS*. 11. Stockholm, SE: International Council of the Aeronautical Sciences, 2022, pp. 1–15.

[77] Lukáš Novák and Drahomír Novák. "Polynomial chaos expansion for surrogate modelling: Theory and software." In: *Beton- und Stahlbetonbau* 113.S2 (2018), pp. 27–32. DOI: https://doi.org/10.1002/best.201800048.

[78] Lukáš Novák and Drahomír Novák. "Surrogate Modelling in the Stochastic Analysis of Concrete Girders Failing in Shear." In: May 2019.

[79] M.K. Ochi and V.S. Shin. "Wind Turbulent Spectra For Design Consideration of Offshore Structures." In: Apr. 2013. DOI: 10.4043/5736-MS.

[80] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3400–3407.

[81] Sanghyuk Park, John Deyst, and Jonathan How. "A New Nonlinear Guidance Logic for Trajectory Tracking." In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. DOI: 10.2514/6.2004-4900.

[82] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. DOI: https://doi.org/10.48550/arXiv.1912.01703.

[83] Jean Pereira, Valter Leite, and G.V. Raffo. "Nonlinear Model Predictive Control on SE(3) for Quadrotor Aggressive Maneuvers." In: *Journal of Intelligent & Robotic Systems* 101 (Mar. 2021). DOI: 10.1007/s10846-021-01310-8.

[84] Patrick Piprek. "Robust Trajectory Optimization Applying Chance Constraints and Generalized Polynomial Chaos." PhD thesis. May 2020. DOI: 10.13140/RG.2.2.17878.88649.

[85] Patrick Piprek, Haichao Hong, and Florian Holzapfel. "Optimal trajectory design accounting for the stabilization of linear time-varying error dynamics." In: *Chinese Journal of Aeronautics* 35.7 (2022), pp. 55–66. ISSN: 1000-9361. DOI: https://doi.org/10.1016/j.cja.2021.10.031.

[86] Johannes Pohlodek et al. "Flexible development and evaluation of machine-learning-supported optimal control and estimation methods via HILO-MPC." In: *ArXiv* (2022). DOI: https://doi.org/10.48550/arXiv.2203.13671.

[87] Tomáš Polóni, Boris Rohal-Ilkiv, and Tor Arne Johansen. "Moving Horizon Estimation for Integrated Navigation Filtering." In: *IFAC-PapersOnLine* 48.23 (2015). 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015, pp. 519–526. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.11.331.

[88] Ondřej Prosicky. *Barn owl, bird landing on wooden roof, action scene in the nature habitat, flying bird, France.* 2024. URL: https://stock.adobe.com/cz/images/barn-owl-tyto-alba-bird-landing-on-wooden-roof-action-scene-in-the-nature-habitat-flying-bird-france/109776303?prev_url=detail (visited on 02/15/2024).

[89] James B. Rawlings. "Moving Horizon Estimation." In: *Encyclopedia of Systems and Control*. Ed. by Samad Tariq Baillieul John. London: Springer London, 2013, pp. 1–7. ISBN: 978-1-4471-5102-9. DOI: 10.1007/978-1-4471-5102-9_4-1.

[90] Guy Revach et al. "KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics." eng. In: (2021). DOI: https://doi.org/10.1109/TSP.2022.3158588.

[91] W. R. T. Roderick, M. R. Cutkosky, and D. Lentink. "Bird-inspired dynamic grasping and perching in arboreal environments." In: *Science Robotics* 6.61 (2021), eabj7562. DOI: 10.1126/scirobotics.abj7562.

[92] Reuven Rubinstein. "The Cross-Entropy Method for Combinatorial and Continuous Optimization." In: *Methodology And Computing In Applied Probability* 1 (1999), pp. 127–190. ISSN: 1573-7713. DOI: 10.1023/A:1010091220143.

[93] Ihab S. Mohamed, Guillaume Allibert, and Philippe Martinet. "Model Predictive Path Integral Control Framework for Partially Observable Navigation: A Quadrotor Case Study." In: Dec. 2020, pp. 196–203. DOI: 10.1109/ICARCV50220.2020.9305363.

[94] Sajjad Safari, Faridoon Shabani, and Dan Simon. "Multirate multisensor data fusion for linear systems using Kalman filters and a neural network." eng. In: *Aerospace science and technology* 39 (2014), pp. 465–471. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2014.06.005.

[95] P.J. Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. "Characterization of the Aerodynamic Ground Effect and Its Influence in Multirotor Control." In: *International Journal of Aerospace Engineering* 2017 (Aug. 2017), pp. 1–17. DOI: 10.1155/2017/1823056.

[96] Davi A. Santos and José A. Bezerra. "On the control allocation of fully actuated multirotor aerial vehicles." In: *Aerospace Science and Technology* 122 (2022), p. 107424. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2022.107424.

[97] Robert G. Sargent. "Verification and validation of simulation models." In: *2008 Winter Simulation Conference*. 2008, pp. 157–169. DOI: 10.1109/WSC.2008.4736065.

[98] Julian D. Schiller et al. "A Lyapunov Function for Robust Stability of Moving Horizon Estimation." In: *IEEE Transactions on Automatic Control* 68.12 (2023), pp. 7466–7481. DOI: 10.1109/TAC.2023.3280344.

[99] Roland Schöbi, Bruno Sudret, and Stefano Marelli. "Rare Event Estimation Using Polynomial-Chaos Kriging." In: *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering* 500 (Mar. 2016), p. D4016002. DOI: 10.1061/AJRUA6.0000870.

[100] Roland Schöbi, Bruno Sudret, and Joe Wiart. "Polynomial-Chaos-Based Kriging." In: *International Journal of Uncertainty Quantification* 5 (Feb. 2015). DOI: 10.1615/Int.J.UncertaintyQuantification.2015012467.

[101] Max Schwenzer et al. "Review on model predictive control: an engineering perspective." In: *The International Journal of Advanced Manufacturing Technology* 117 (Nov. 2021), pp. 1–23. DOI: 10.1007/s00170-021-07682-3.

[102] Hoseong Seo, Suseong Kim, and H. Jin Kim. "Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 6362–6368. DOI: 10.1109/ICRA.2017.7989751.

[103] Jawad Shah, Mohamed Okasha, and Waleed Faris. "Gain scheduled integral linear quadratic control for quadcopter." In: *International Journal of Engineering and Technology(UAE)* 7 (Oct. 2018), pp. 81–85. DOI: 10.14419/ijet.v7i4.13.21334.

[104] Iliyana Simeonova and Tzvetan Semerdjiev. "Specific Features of IMM Tracking Filter Design." In: *An International Journal of Information and Security* 9 (Jan. 2002). DOI: 10.11610/isij.0909.

[105] Dan Simon. *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*. eng. 2006. ISBN: 9780470045343. DOI: https://doi.org/10.1002/0470045345.ch11.

[106] Ewoud J. J. Smeur, Qiping Chu, and Guido C. H. E. de Croon. "Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles." In: *Journal of Guidance, Control, and Dynamics* 39.3 (2016), pp. 450–461. DOI: 10.2514/1.G001490.

[107] Pantelis Sopasakis, Emil Fresk, and Panagiotis Patrinos. "OpEn: Code Generation for Embedded Nonconvex Optimization." In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 6548–6554. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.071.

[108] Lorenzo Stella et al. "A simple and efficient algorithm for nonlinear model predictive control." In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (2017), pp. 1939–1944. DOI: 10.1109/CDC.2017.8263933.

[109] Xin Sun, Senchun Chai, and Baihai Zhang. "Trajectory Planning of the Unmanned Aerial Vehicles with Adaptive Convex Optimization Method." In: *IFAC-PapersOnLine* 52.12 (2019). 21st IFAC Symposium on Automatic Control in Aerospace ACA 2019, pp. 67–72. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.11.071.

[110] István Szita and András Lörincz. "Learning Tetris Using the Noisy Cross-Entropy Method." In: *Neural Computation* 18.12 (2006), pp. 2936–2941. DOI: 10.1162/neco.2006.18.12.2936.

[111] Michael Szmuk and Behçet Açikmese. "Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time." In: *arXiv: Optimization and Control* (2018). DOI: https://doi.org/10.2514/6.2018-0617.

[112] Nguyen Khoi Tran, Eitan Bulka, and Meyer Nahon. "Quadrotor control in a wind field." In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2015, pp. 320–328. DOI: 10.1109/ICUAS.2015.7152306.

[113] Le Ngoc Truc and Tung Nguyen. "Quasi-physical modeling of robot IRB 120 using Simscape Multibody for dynamic and control simulation." In: *Turkish Journal of Electrical Engineering & Computer Sciences* 28 (July 2020), pp. 1949–1964. DOI: 10.3906/elk-1909-131.

[114] Sridhar Ungarala, Eric Dolence, and Keyu Li. "Constrained Extended Kalman Filter for Nonlinear State Estimation." In: *IFAC Proceedings Volumes* 40.5 (2007). 8th IFAC Symposium on Dynamics and Control of Process Systems, pp. 63–68. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20070606-3-MX-2915.00058.

[115] P. Vachhani, S. Narasimhan, and R. Rengaswamy. "Recursive state estimation in nonlinear processes." In: *Proceedings of the 2004 American Control Conference*. Vol. 1. 2004, 200–204 vol.1. DOI: 10.23919/ACC.2004.1383604.

[116] Andreas Wächter and Lorenz T Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." eng. In: *Mathematical programming* 106.1 (2006), pp. 25–57. ISSN: 0025-5610. DOI: 10.1007/S10107-004-0559-Y.

[117] Yiming Wan and Tamas Keviczky. "Real-time nonlinear moving horizon observer with pre-estimation for aircraft sensor fault detection and estimation." In: *International Journal of Robust and Nonlinear Control* 29 (Dec. 2017). DOI: 10.1002/rnc.4011.

[118] Bingheng Wang et al. "Neural Moving Horizon Estimation for Robust Flight Control." In: *IEEE Transactions on Robotics* 40 (2024), pp. 639–659. DOI: 10.1109/TRO.2023.3331064.

[119] John Wang and Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016.

[120] Xiaofeng Wang and Naira Hovakimyan. "L1 adaptive controller for nonlinear time-varying reference systems." In: *Systems & Control Letters* 61.4 (2012), pp. 455–463. ISSN: 0167-6911. DOI: https://doi.org/10.1016/j.sysconle.2012.01.010.

[121] Simon Watkins et al. "Atmospheric Winds and Their Implications for Microair Vehicles." In: *AIAA Journal* 44.11 (2006), pp. 2591–2600. DOI: 10.2514/1.22670.

[122] Zhuohuan Wu et al. "L1 Adaptive Augmentation for Geometric Tracking Control of Quadrotors." In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 1329–1336. DOI: 10.1109/ICRA46639.2022.9811946.

[123] Enric Xargay, Naira Hovakimyan, and Chengyu Cao. "L1 adaptive controller for multi-input multi-output systems in the presence of nonlinear unmatched uncertainties." In: Aug. 2010, pp. 874 –879. DOI: 10.1109/ACC.2010.5530686.

[124] Fouad Yacef et al. "Energy-Efficiency Path Planning for Quadrotor UAV Under Wind Conditions." In: *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*. Vol. 1. 2020, pp. 1133–1138. DOI: 10.1109/CoDIT49905.2020.9263968.

[125] Fouad Yacef et al. "Trajectory optimisation for a quadrotor helicopter considering energy consumption." In: *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*. 2017, pp. 1030–1035. DOI: 10.1109/CoDIT.2017.8102734.

[126] Larry C. Young. "Orthogonal collocation revisited." In: *Computer Methods in Applied Mechanics and Engineering* 345 (2019), pp. 1033–1076. ISSN: 0045-7825. DOI: 10.1016/j.cma.2018.10.019.

[127] Zhijie Zhu, Edward Schmerling, and Marco Pavone. "A convex optimization approach to smooth trajectories for motion planning with car-like robots." In: *2015 54th IEEE Conference on Decision and Control (CDC)* (2015), pp. 835–842.