

UNIFIED NONLINEAR MODEL PREDICTIVE PATH-FOLLOWING CONTROL WITH
COLLISION AVOIDANCE FOR ROBOTIC SYSTEMS

by

NAREN REDDY DHANSRI

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2026

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Manohar Das, Ph.D., Chair
Edward Y. Gu, Ph.D., Co-Chair
Guangzhi Qu, Ph.D.
Li Li, Ph.D.

© Copyright by Naren Reddy Dhansri, 2026
All rights reserved

To my mother, father, wife and son

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my mother, father, wife, and son for their unconditional love, sacrifices, patience, encouragement, and support throughout this journey. Their strength, belief in me, and unwavering support gave me the motivation to persevere and complete this dissertation.

I sincerely thank Prof. Das for his guidance, support, and encouragement, especially for serving as my advisor after Prof. Gu's retirement. I am particularly grateful for his steady direction, thoughtful technical feedback, and support during the final stages of this dissertation. I also extend my sincere gratitude to Prof. Gu for his mentorship, encouragement, and support throughout my doctoral studies, both before and after his retirement. Their guidance was instrumental in shaping this research and bringing this dissertation to completion.

I would like to sincerely thank my committee members, Prof. Guangzhi Qu and Prof. Li Li, for their valuable feedback, guidance, and support throughout the dissertation process. Their thoughtful questions and constructive suggestions helped strengthen the quality and clarity of this work.

I am thankful to my brother for his constant encouragement and motivation throughout this journey, and I also thank the Traxen Inc. team for their support, especially Ali Maleki, CEO of Traxen, for his continued motivation and support throughout this process.

Naren Reddy Dhansri

PREFACE

This dissertation was prepared as part of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering at Oakland University. It presents the research conducted during the doctoral program and documents the methods, developments, and results obtained throughout the course of this work.

The dissertation is organized into nine chapters. Chapter 1 introduces the research motivation, problem statement, literature background, and main contributions. Chapter 2 presents the mathematical preliminaries used throughout the dissertation. Chapter 3 formulates the general robotic modeling and unified Model Predictive Path Following Control framework. Chapter 4 develops analytical derivatives of distance functions for articulated robotic systems. Chapter 5 presents smooth path generation using composite Bézier curves. Chapter 6 develops the structure-exploiting primal–dual interior-point solver. Chapters 7 and 8 demonstrate the proposed framework on a tractor–semitrailer system and an underactuated log-arm robot, respectively. Chapter 9 summarizes the conclusions, discusses computational complexity, and outlines future research directions.

The purpose of this preface is to provide a brief guide to the organization and scope of the dissertation. Detailed technical contributions, methodology, and results are presented in the abstract and in the individual chapters that follow.

ABSTRACT

UNIFIED NONLINEAR MODEL PREDICTIVE PATH-FOLLOWING CONTROL WITH COLLISION AVOIDANCE FOR ROBOTIC SYSTEMS

By

NAREN REDDY DHANSRI

Robotic systems operating in constrained environments must often follow prescribed geometric paths while respecting nonlinear dynamics, actuator limits, state constraints, and obstacle-clearance requirements. Conventional trajectory tracking prescribes both path and timing, which can become restrictive when the motion is dynamically infeasible or when obstacle avoidance requires temporary deviation from the prescribed path. This dissertation develops a unified nonlinear Model Predictive Path-Following Control (MPPFC) framework with collision avoidance, in which path following, path progression, dynamics, constraints, and obstacle avoidance are jointly optimized within a single Model Predictive Control (MPC) problem.

The central contribution is a complete computational framework that makes the unified collision-aware MPPFC formulation practical by developing analytical collision-constraint derivatives and a tailored primal–dual interior-point solver for the resulting nonlinear control problem. Progression along the prescribed path is optimized online rather than imposed through a fixed time parameterization, allowing adaptation when actuator limits, dynamic constraints, or obstacle-

clearance requirements become active. Smooth composite Bézier paths provide analytical path derivatives, while differentiable minimum-distance collision constraints provide analytical gradients and Hessians through closest-point sensitivity analysis and articulated kinematic propagation.

The resulting nonlinear optimal-control problem is solved using a second-order primal–dual interior-point method tailored to the structured prediction-horizon formulation of MPPFC. The solver incorporates analytical derivatives, slack and relaxation variables, barrier terms, scaling, regularization, warm starting, and sparse Newton-system solution. By preserving the stage-wise temporal structure, the method efficiently handles path-following objectives, nonlinear dynamics, constraints, and collision-avoidance inequalities online.

The proposed methods are evaluated on two robotic systems: an articulated tractor–trailer navigating around static and dynamic obstacles, and an underactuated planar log-arm robot performing constrained path following. These studies show that adaptive path progression, obstacle avoidance, constraint satisfaction, and feasible control generation can be handled within a unified predictive control framework for nonlinear systems subject to actuator limits, obstacle interactions, changing clearance constraints, and limited actuation, while preserving a computational structure suitable for future real-time embedded implementation.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
PREFACE	v
ABSTRACT	vi
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF ALGORITHMS	xviii
LIST OF ABBREVIATIONS	xix
LIST OF SYMBOLS, NOTATIONS AND OPERATORS	xxi
CHAPTER ONE	
INTRODUCTION	1
1.1 Background and Motivation	1
1.2 Literature Review	6
1.3 Research Gap and Problem Statement	19
1.4 Research Challenges	20
1.5 Proposed Solution and Dissertation Contributions	23
1.6 Dissertation Organization	26
CHAPTER TWO	
MATHEMATICAL BACKGROUND	28
2.1 Linear Algebra Preliminaries and Matrix Identities	28
2.2 Derivatives of Matrix-Valued Functions	34
2.3 Derivative Identities for Matrix-Valued Functions	40
2.4 Kronecker-Product Differential Identities	47

TABLE OF CONTENTS—Continued

2.5 Auxiliary Linear Algebra Results	50
2.6 Chapter Summary	55
CHAPTER THREE	
GENERALIZED ROBOTIC MODELLING AND MODEL PREDICTIVE PATH FOLLOWING CONTROL FORMULATION	56
3.1 Introduction and Scope	56
3.2 Configuration Spaces for Robotic Systems	57
3.3 Rigid-Body Transformations and Forward Kinematics	60
3.4 State-Space Representation of Robotic Dynamics	66
3.5 Extended State Models for Robotic Systems	70
3.6 Task-Space Outputs and End-Effector Maps	71
3.7 Generalization to Nonlinear Dynamic Systems	73
3.8 Model Predictive Path Following Control Formulation	74
3.9 Unified Nonlinear MPPFC Challenges	87
3.10 Supplementary Analytical Derivatives for MPPFC	87
3.11 Chapter Summary	95
CHAPTER FOUR	
ANALYTICAL DISTANCE DERIVATIVES FOR ARTICULATED ROBOTIC SYSTEMS	97
4.1 Introduction and Motivation	97
4.2 Minimum Distance Formulation	100
4.3 Analytical Distance Gradients	106
4.4 Analytical Distance Hessians	114

TABLE OF CONTENTS—Continued

4.5 Aggregated Distance Constraints	125
4.6 Computational Evaluation of Distance Derivatives	129
4.7 Numerical Results	133
4.8 Chapter Summary	140
CHAPTER FIVE	
SMOOTH PATH GENERATION FROM WAYPOINTS	142
5.1 Introduction	142
5.2 Bézier Curves and Composite Bézier Paths	145
5.3 Polyline-Scaled C^2 Cubic Bézier Paths	153
5.4 Composite Bézier Path Examples	165
5.5 Chapter Summary	167
CHAPTER SIX	
STRUCTURE-EXPLOITING PRIMAL-DUAL INTERIOR-POINT METHOD FOR UNIFIED MPPFC	168
6.1 Softened Unified MPPFC Formulation	168
6.2 Scaled Interior-Point Reformulation	171
6.3 Perturbed Primal–Dual KKT Conditions	176
6.4 Second-Order Optimality Conditions	183
6.5 Structure-Exploiting Newton Step Computation	187
6.6 Globalization and Search-Direction Enhancement	204
6.7 Warm-Starting the Unified MPPFC Optimization Cycle	218
6.8 Scaling-Factor Selection	222

TABLE OF CONTENTS—Continued

6.9 Chapter Summary	223
CHAPTER SEVEN	
APPLICATION 1: TRACTOR–SEMITRAILER UNIFIED PATH FOLLOWING AND COLLISION AVOIDANCE	225
7.1 Introduction	225
7.2 States, Inputs, Frames, and Notation	226
7.3 Vehicle Geometry and Vehicle Kinematics	228
7.4 Energy-Based Vehicle Dynamics	231
7.5 Generalized Forces and Input mapping	238
7.6 Rigid-Body Transformations for Collision Modeling	245
7.7 Obstacle-Aware Cold-Start	247
7.8 Simulation Framework and Setup	253
7.9 Closed-Loop Simulation Results	258
7.10 Chapter Summary	274
CHAPTER EIGHT	
APPLICATION 2: PATH FOLLOWING FOR AN UNDERACTUATED LOG-ARM ROBOT	276
8.1 System Model and Control Setup	276
8.2 Dynamics and Output Model	278
8.3 Simulation Results	282
8.4 Chapter Summary	295
CHAPTER NINE	
CONCLUSIONS, COMPUTATIONAL COMPLEXITY, AND FUTURE WORK	297

TABLE OF CONTENTS—Continued

9.1 Computational Complexity Analysis	297
9.2 Summary of Contributions	299
9.3 Limitations of the Present Work	302
9.4 Future Work	306
REFERENCES	316

LIST OF TABLES

Table 4.1	Comparison of analytical vs. FD Pairwise Distance Gradient and Hessian computations	136
Table 4.2	Error summary for analytical vs FD derivatives in the 3D, 3-link robot case	140
Table 4.3	Comparison of analytical and FD derivative computation times for the 3D, 3-link robot case	140
Table 7.1	Geometric and Dynamic Parameters of Tractor–Trailer Model	228
Table 7.2	Numerical Parameters of the Tractor–Trailer Model	256
Table 7.3	Tractor–Trailer Simulation and Controller Parameters	257
Table 7.4	Tractor-Trailer MPPFC Constraints and Scaling Factors	257
Table 7.5	Tractor–Trailer Closed-Loop Simulation Scenarios	258
Table 8.1	Physical Parameters of the Log-Arm System	277
Table 8.2	Log-Arm MPPFC State and Control Constraints	277
Table 8.3	Log-Arm MPPFC Simulation and Solver Parameters	282

LIST OF FIGURES

Figure 3.1	Path-Aligned Corridor	83
Figure 3.2	Path-Speed Consistency and Orientation Error	85
Figure 4.1	Pairwise Distance Vectors	102
Figure 4.2	2D Tractor-Trailer Setup for Distance-Derivative Validation	133
Figure 4.3	2D Tractor-Trailer Distance-Derivative Verification	135
Figure 4.4	Runtime Scaling with Pairwise Distance Terms	136
Figure 4.5	Soft-Minimum Aggregation Error	137
Figure 4.6	Hard-Minimum and Global Soft-Minimum Distance Along the Trajectory	138
Figure 4.7	3D Distance-Derivative Validation Scene	139
Figure 5.1	Real-Time Smooth Reference Path Generation Pipeline	163
Figure 5.2	Smooth Composite Cubic Bézier Path Construction Example in $\mathbb{R}^2$	166
Figure 5.3	Smooth Composite Cubic Bézier Path Construction Example in $\mathbb{R}^3$	167
Figure 7.1	Tractor-Semitrailer Model and Geometric Parameters	229
Figure 7.2	Closed-Loop Simulation Architecture for Tractor-Trailer Unified Path Following and Collision Avoidance	255
Figure 7.3	Static-map recovery scenario S1 with large initial path offset and favorable initial heading	259
Figure 7.4	Vehicle states and control responses for static-map recovery scenario S1	260
Figure 7.5	Path-following performance for static-map recovery scenario S1	261
Figure 7.6	Static-map recovery scenario S2 with unfavorable initial heading and obstacle-constrained recovery	261

LIST OF FIGURES—Continued

Figure 7.7	Vehicle states and control responses for static-map recovery scenario S2	262
Figure 7.8	Path-following performance for static-map recovery scenario S2	262
Figure 7.9	Interior-point solver iterations for static-map scenario S2	263
Figure 7.10	Combined dynamic-obstacle scenario including slower-moving, stopped, and opposing obstacles	264
Figure 7.11	Vehicle states and controls during the combined dynamic-obstacle scenario	265
Figure 7.12	Path-following and path-progression variables during the combined dynamic-obstacle scenario	265
Figure 7.13	Interior-point iteration count during the combined dynamic-obstacle scenario	266
Figure 7.14	Local interaction with dynamic obstacle D01 traveling in the same direction as the host vehicle	267
Figure 7.15	Local interaction with dynamic obstacle D02 stopped along the reference corridor	268
Figure 7.16	Local interaction with dynamic obstacle D03 traveling opposite to the host vehicle	269
Figure 7.17	Multi-obstacle same-direction scenario with two slower vehicles traveling along the reference corridor	270
Figure 7.18	Vehicle states and control responses for the multi-obstacle same-direction scenario	271
Figure 7.19	Path-following and path-progression variables for the multi-obstacle same-direction scenario	271
Figure 7.20	Local interaction with same-direction obstacles in Scenario S4	272

LIST OF FIGURES—Continued

Figure 7.21	Interior-point iteration history for Scenario S2 using the Gauss–Newton collision-Hessian approximation	273
Figure 8.1	Underactuated Log-Arm Robot and Desired Circular Path	276
Figure 8.2	Path tracking performance of the log-arm system for the 0.125 m radius case	283
Figure 8.3	Evolution of system log-arm states for 0.125 m radius case	284
Figure 8.4	Log-arm system control inputs for the 0.125 m radius case	284
Figure 8.5	Log-arm system path position and path velocity for 0.125 m radius case	285
Figure 8.6	Number of interior-point iterations for the 0.125 m radius case	285
Figure 8.7	Path tracking performance of the log-arm system for the 0.25 m radius case	286
Figure 8.8	Evolution of system log-arm states for 0.25 m radius case	286
Figure 8.9	Log-arm system control inputs for the 0.25 m radius case	287
Figure 8.10	Log-arm system path position and path velocity for the 0.25 m radius case	287
Figure 8.11	Number of interior-point iterations for the 0.25 m radius case	287
Figure 8.12	Path tracking performance of the log-arm system for the 0.5 m radius case	288
Figure 8.13	Evolution of system log-arm states for 0.5 m radius case	289
Figure 8.14	Log-arm system control inputs for the 0.5 m radius case	290
Figure 8.15	Log-arm system path position and path velocity for the 0.5 m radius case	290
Figure 8.16	Number of interior-point iterations for the 0.5 m radius case	290

LIST OF FIGURES—Continued

Figure 8.17	Path tracking performance of the log-arm system for the 0.25 m radius case with relaxed prismatic displacement bounds	291
Figure 8.18	Evolution of system log-arm states for the 0.25 m radius case with relaxed prismatic displacement bounds	292
Figure 8.19	Log-arm system control inputs for the 0.25 m radius case with relaxed prismatic displacement bounds	293
Figure 8.20	Path tracking performance of the log-arm system for the 0.5 m radius case with relaxed prismatic displacement and torque bounds	293
Figure 8.21	Evolution of system log-arm states for the 0.5 m radius case with relaxed prismatic displacement and torque bounds	294
Figure 8.22	Log-arm system control inputs for the 0.5 m radius case with relaxed prismatic displacement and torque bounds	294

LIST OF ALGORITHMS

Algorithm 2.1	LDL^T Factorization with Inertia Detection	54
Algorithm 4.1	Distance-Derivative Evaluation for Convex Body Pairs	129
Algorithm 5.1	Thomas Method for Block Tridiagonal Linear Systems	163
Algorithm 5.2	Ramer–Douglas–Peucker Polyline Simplification	165
Algorithm 6.1	Fast Computation of Primal-Dual Search Direction	200
Algorithm 6.2	Interior-Point Optimization Cycle for the Unified MPPFC Problem	213
Algorithm 6.3	Warm-Start Initialization for the Unified MPPFC Optimization Cycle	221
Algorithm 7.1	Obstacle-Aware Cold-Start Initialization for the Tractor-trailer Unified MPPFC Problem	250

LIST OF ABBREVIATIONS

MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
MPPFC	Model Predictive Path-Following Control
GJK	Gilbert-Johnson-Keerthi
EPA	Expanding Polytope Algorithm
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
TrajOpt	Trajectory Optimization
KKT	Karush-Kuhn-Tucker
IPM	Interior-Point Method
HPIPM	High-Performance Interior-Point-Method
DH	Denavit-Hartenberg
PSD	Positive semidefinite
FD	Finite difference
RDP	Ramer–Douglas–Peucker
LU	Lower-upper matrix factorization
LDL^T	LDL^T Symmetric indefinite factorization
LICQ	Linear Independence Constraint Qualification
IPOPT	Interior Point Optimizer
SOC	Second Order Correction
CG	Center of Gravity
GPS	Global Positioning System

LIST OF ABBREVIATIONS—Continued

INS	Inertial Navigation System
IMU	Inertial Measurement Unit
GN	Gauss Newton
RL	Reinforcement Learning
MHE	Moving Horizon Estimation

LIST OF SYMBOLS, NOTATIONS AND OPERATORS

$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers
M^T	Transpose of matrix M
M^{-1}	Inverse of matrix M
M^+	Moore-Penrose pseudoinverse of M
$\det(M)$	Determinant of matrix M
$\text{rank}(M)$	Rank of matrix M
$\text{In}(M)$	Inertia of matrix M
$n_+(M)$	Number of positive eigenvalues of M
$n_-(M)$	Number of negative eigenvalues of M
$n_0(M)$	Number of zero eigenvalues of M
I_n	Identity matrix of size $n \times n$
$I_{\pm}$	Block difference matrix $I_{\pm} = [I_n \quad -I_n] \in \mathbb{R}^{n \times 2n}$
$\mathbf{1}_n$	Vector of ones of length n
$\otimes$	Kronecker product
$\odot$	Hadamard (elementwise) product
$\circledast$	Special block-wise product introduced for compact block-structured derivative expressions
$\text{vec}(\cdot)$	Vectorization operator
$\text{vec}_b(\cdot)$	Block-vectorization operator
$\text{diag}(\cdot)$	Diagonal matrix formed from a vector

LIST OF SYMBOLS, NOTATIONS AND OPERATORS —Continued

$\frac{\partial F}{\partial x_i}$	Partial derivative of F with respect to scalar x_i
$\frac{\partial^2 F}{\partial x_i \partial x_j}$	Second-order partial derivative of F with respect to scalars x_i and x_j
$\frac{\partial^3 F}{\partial x_i \partial x_j \partial x_k}$	Third-order partial derivatives of F with respect to scalar x_i, x_j and x_k
$\nabla_x F(x)$	Vertically stacked derivative of F with respect to vector x
$\nabla_x F^\top(x)$	Vertically stacked derivative of $F^\top$ with respect to vector x
$D_x F(x)$	Horizontally stacked derivative of F with respect to vector x
$D_x F^\top(x)$	Horizontally stacked derivative of $F^\top$ with respect to vector x
$\nabla_{xx}^2 F(x)$	Block-matrix arrangement of the second partial derivatives of F with respect to vector x
$\nabla_{xxx}^3 F(x)$	Block-structured arrangement of third partial derivatives of F with respect to vector x
$\mathcal{A}_{(k)}$	Mode- k unfolding of tensor $\mathcal{A}$
Σ	Summation operator
Π	Product operator
$\exists$	Existential quantifier, meaning “there exists”
$\forall$	Universal quantifier, meaning “for all”
$\in$	Set membership
$\subset$	Subset relation
$:=$	Defined as
$\triangleq$	Defined as

LIST OF SYMBOLS, NOTATIONS AND OPERATORS —Continued

$\circ$	Function composition
$\Rightarrow$	Logical implication
n_x	Number of variables or dimension associated with the subscripted vector or quantity x , unless otherwise defined
n_p	Number of prediction stages in the horizon
ρ	Generalized configuration coordinates
$\mathcal{Q}$	Configuration manifold
q	Generalized translational coordinates
φ	Generalized rotational coordinates
$SO(3)$	Special orthogonal group representing 3D rotations
$SE(3)$	Special Euclidean group representing rigid-body poses
$R(\cdot)$	Rotation matrix
$T(\cdot)$	Translation mapping
$\partial\mathcal{C}$	Boundary of convex body $\mathcal{C}$
$B(s)$	Boundary surface parameterization of a convex body
$T_\rho\mathcal{Q}$	Tangent space of $\mathcal{Q}$ at configuration ρ
ξ	Generalized velocity
x	Generalized state vector
u	Generalized input vector
y	System output vector
ζ	Internal or auxiliary state vector

LIST OF SYMBOLS, NOTATIONS AND OPERATORS —Continued

$f(x, u)$	Nonlinear system dynamics
$h(x)$	Output map
$c_i(x, u)$	Inequality constraint map
$c_e(x, u)$	User-defined equality constraint map
k	Prediction-stage index
$(\cdot)_k$	Quantity evaluated at prediction stage k
t	MPC optimization cycle index
τ	Sampling interval
J_k	Stage cost at prediction stage k
θ	Path-progression variable
v	Path speed
η	Path-timing state vector
A_η	State matrix for the path-timing integrator chain
B_η	Input matrix for the path-timing integrator chain
$Q_{(\cdot)}$	Quadratic weighting matrix for the corresponding MPC cost term
δ	Required safety margin or clearance distance
d_{min}	Required safety margin or clearance distance
$(\bar{\cdot})_k$	Upper bound on an optimization variable at stage index k
$(\underline{\cdot})_k$	Lower bound on an optimization variable at stage index k
$(\cdot)^*$	Superscript indicating an optimal value or optimizer

LIST OF SYMBOLS, NOTATIONS AND OPERATORS —Continued

$1_{\{\cdot\}}$	Indicator function that equals 1 when the stated condition is true and 0 otherwise
d_{ij}	Pairwise minimum distance between body i and body j
$\ \cdot\ $	Euclidean norm
$\ \cdot\ _p$	p-norm
C^r	Parametric continuity of order r
G^r	Geometric continuity of order r
$dist(\cdot, \cdot)$	Distance between a point and a set or segment
W	Ordered waypoint set
z_k	Physical system variable group at stage k , collecting states and controls
p_k	Path variable group at stage k , collecting path and timing variables
s_k	Slack variable vector at stage k
ε_k	Inequality constraint-softening variable vector at stage k
μ	Barrier parameter in the primal-dual interior-point method
L_μ	Barrier Lagrangian for fixed barrier parameter μ
L	Exact constrained-problem Lagrangian of the softened MPC problem
$\lambda_{(\cdot),k}$	Equality-constraint multiplier for the constraint group indicated by the subscript at stage k
$\gamma_{(\cdot),k}$	Lower-bound dual variable for the bound group indicated by the subscript at stage k

LIST OF SYMBOLS, NOTATIONS AND OPERATORS —Continued

$\beta_{(\cdot),k}$	Upper-bound dual variable for the bound group indicated by the subscript at stage k
$\widetilde{(\cdot)}$	Scaled version of the corresponding variable, residual, or multiplier
$\sigma_{(\cdot)}$	Scaling vector for the corresponding primal or slack variable group
$\rho_{(\cdot)}$	Scaling vector for the corresponding equality-constraint group
Δ_w	Primal regularization matrix in the regularized reduced primal-dual KKT system
Δ_c	Equality-constraint regularization matrix in the regularized reduced primal-dual KKT system
δ_w	Scalar primal regularization magnitude used to construct Δ_w
δ_c	Scalar constraint regularization magnitude used to construct Δ_c
E_μ	Optimality error for the barrier subproblem
E_0	Optimality error for the zero-barrier softened MPC problem
$S(\cdot)$	Horizon-shift operator used for warm starting

CHAPTER 1 - INTRODUCTION

1.1 Background and Motivation

Autonomous robotic systems increasingly operate in environments containing geometric constraints, static obstacles, and moving objects. Robotic manipulators work near equipment or humans, while mobile, aerial, and marine robots must follow desired routes while avoiding hazards. In such settings, controllers must satisfy dynamic and actuator constraints, maintain collision clearance, and compute actions in real time.

A traditional architecture separates motion planning from control. A planner generates a collision-free path, and a controller tracks a time-parameterized trajectory derived from that path [1]. While effective in structured environments, this approach can be problematic for nonlinear, underactuated, or constrained systems. A geometrically feasible path may not be dynamically executable because of inertia, actuator limits, nonholonomic constraints, or nonminimum-phase behavior [2]– [5]. Consequently, strict trajectory tracking may reduce safety margins, produce aggressive control actions, or degrade closed-loop performance.

Path following provides a more flexible formulation by specifying a geometric path independently of timing. Hauser and Hindman introduced maneuver regulation using a path-progress variable [6], and path-following control was later developed for robotic and marine systems, including autonomous underwater vehicles [7]. Aguiar et al. demonstrated that relaxing timing requirements can overcome limitations associated with reference tracking for nonminimum-phase systems [2],

[4], [5]. By regulating path progression, the controller can adapt speed according to dynamic feasibility, actuator limits, and environmental conditions.

Model Predictive Path-Following Control (MPPFC) embeds this concept within a receding-horizon optimization framework. Control inputs and path progression are optimized simultaneously while accounting for nonlinear dynamics and constraints. Foundational work established recursive feasibility and convergence guarantees for constrained path following [8]– [12], while experimental studies demonstrated practical implementation on industrial robots [13]. These results show that MPPFC provides a rigorous framework for constrained nonlinear motion control in which timing is optimized rather than prescribed.

The central advantage of MPPFC is that it converts path traversal into a decision process: the optimizer selects the path speed while satisfying system constraints. This flexibility is useful when the system starts far from the path, encounters actuator saturation, or must temporarily deviate for safety; in such cases, MPPFC can slow, pause, or resume path progression when feasible.

Most MPPFC formulations, however, assume that the reference path is collision free. This assumption is restrictive when obstacles are detected online, move during operation, or cannot be fully incorporated into an offline plan. Collision-aware MPPFC has therefore been studied using artificial variables, penalty terms, clearance conditions, and geometric constraints [14]– [17]. Application-oriented MPPFC studies further show that the same constrained path-following framework extends to diverse robotic and vehicle systems, including tower cranes,

underactuated vehicles, mobile robots, and articulated vehicle systems [18]– [21]. More recent collision-aware formulations have further considered distributed collision-avoidance schemes and dynamic obstacles modeled through soft constraints [22], [23]. However, these formulations are often tailored to specific robot classes, obstacle representations, or avoidance mechanisms, and the added clearance constraints, artificial variables, or penalty terms increase the size and complexity of the online optimization problem.

Collision avoidance has also been studied extensively through potential-field and predictive-control formulations. Artificial potential fields provide simple repulsive terms for obstacle avoidance and have been combined with MPC in autonomous driving and mobile robotics [24]– [26], but they are known to suffer from local minima and oscillatory behavior [27].

More generally, collision avoidance in Nonlinear Model Predictive Control (NMPC) is commonly formulated through distance or clearance constraints between robot and environment models. Efficient proximity-query algorithms such as GJK, EPA, and feature-tracking methods provide distance and contact information for convex bodies [28]– [32] and are widely used in robotics software [33]. However, distance evaluation alone is insufficient for efficient second-order optimization.

Gradient-based NMPC requires derivatives of collision constraints with respect to states and optimization variables. Existing approaches often rely on signed-distance fields, sequential convexification, finite differences, automatic differentiation, or quasi-Newton approximations [34]– [37]. Although these

methods are effective in many applications, they can become expensive or inaccurate as the number of collision pairs, prediction stages, and decision variables increases. This issue becomes more significant for articulated systems, where collision checking must be performed for multiple links, trailers, or body components rather than a single point or bounding circle.

Recent research has therefore focused on differentiable and optimization-compatible collision computation. Existing approaches include convex-duality reformulations [38], differentiable collision primitives [39], optimization-defined collision metrics [40], optimization-based acceleration of collision detection [41], and randomized smoothing for differentiable collision detection [42]. These studies show that collision information can be made more compatible with nonlinear optimization, but they also reveal different modeling choices. Some methods introduce auxiliary variables or separation certificates, while others use surrogate collision measures or smoothed approximations of nonsmooth distance functions. These approaches are valuable, but they do not directly provide analytical first- and second-order sensitivities of the Euclidean minimum-distance function for smooth convex parametric bodies.

Motivated by this gap, this dissertation derives analytical gradients and Hessians of the Euclidean minimum-distance function between smooth convex bodies. Closest points are treated as solutions of parameter-dependent optimization problems, and their sensitivities are obtained by differentiating the associated optimality conditions using classical results from sensitivity analysis and implicit

differentiation [43]– [45]. This preserves the geometric interpretation of clearance while providing derivative information suitable for second-order optimization. In contrast to finite-difference or quasi-Newton treatments, the resulting derivatives are tied directly to the geometry of the bodies and to the motion of the closest-point pair.

Accurate collision derivatives must also be paired with efficient numerical optimization. Collision-aware MPPFC generates large nonlinear programs containing dynamics, path-following objectives, bounds, path-progress variables, and clearance constraints. Although these problems exhibit strong stage-wise sparsity, general-purpose solvers may not exploit this structure efficiently. Fast MPC and embedded optimization research have shown the importance of optimal-control structure, warm starts, sparse linear algebra, and tailored solver design for real-time performance [46]– [58]. This dissertation combines these ideas with analytical collision derivatives so that clearance constraints can be embedded directly into a sparse second-order NMPC formulation.

This dissertation addresses these issues through a unified collision-aware MPPFC framework. Smooth convex geometric models are used for differentiable minimum-distance evaluation, analytical gradients and Hessians are obtained through closest-point sensitivity analysis and propagated through system kinematics, and the resulting optimization problem is solved using a structure-exploiting primal–dual interior-point method with warm-start capabilities. The contribution is methodological rather than application specific. The autonomous

tractor-trailer system and underactuated planar log-arm mechanism considered later serve as representative validation examples involving nonlinear dynamics, articulation, and collision interactions. Together, these examples demonstrate a general framework for real-time collision-aware path-following control based on differentiable geometric constraints and structure-exploiting optimization.

1.2 Literature Review

The proposed dissertation combines path-following control, collision-aware nonlinear predictive optimization, differentiable geometric computation, and structure-exploiting numerical methods. Although each area is well established, their integration remains limited. This section reviews the most relevant literature and identifies the gap addressed in this dissertation.

1.2.1 Model Predictive Path-Following Control

Path following differs from trajectory tracking because the reference is a geometric curve rather than a time-parameterized trajectory. This flexibility improves feasibility for constrained nonlinear systems by allowing progression along the path to adapt to dynamic and actuator limitations. In trajectory tracking, the time evolution of the reference is prescribed externally. If that timing is incompatible with the plant dynamics, the controller must either generate large control effort or accept tracking error. In path following, timing becomes part of the control objective, which provides an additional degree of freedom for constrained motion.

This distinction becomes important when the reference path is generated by

a higher-level planner without full knowledge of the closed-loop dynamic response. A planned geometric route may be acceptable from a kinematic or collision-checking perspective, but the corresponding time assignment may require accelerations, angular rates, or actuator effort that exceeds physical limits. In such cases, the tracking problem may become unnecessarily difficult or even infeasible, although the underlying geometric path remains useful. By allowing the path parameter to evolve as an optimization variable, MPPFC separates the question of where the system should move from the question of how fast it should move. This separation is central to the dissertation because obstacle avoidance and actuator limits are handled more naturally when path progression can be adjusted by the controller rather than imposed externally.

Early work by Hauser and Hindman introduced maneuver regulation as a path-based alternative to trajectory tracking [6]. Aguiar, Hespanha, and Kokotović showed that relaxing timing requirements can overcome limitations in tracking nonminimum-phase systems [2], [4], [5]. These studies established path following as a distinct objective for underactuated, nonholonomic, and constrained systems. The key insight is that convergence to a geometric set and progression along that set need not be tied to a fixed clock. This makes path following particularly relevant for systems whose feasible transient response depends strongly on state, speed, or actuator availability.

Path-following methods have since been applied to underwater vehicles, ground vehicles, marine vessels, cranes, manipulators, and mobile robots [7], [17]–

[21]. In such applications, the path is prescribed while the speed profile adapts to operating conditions and constraints. This separation between geometry and timing is useful in applications where the desired route is known but the appropriate speed is influenced by curvature, dynamic feasibility, obstacles, or actuator limits.

Model Predictive Path-Following Control incorporates the path parameter into a finite-horizon optimal-control problem, allowing both control inputs and path progression to be optimized. Foundational formulations were developed by Faulwasser, Kern, and Findeisen [10], with later work establishing convergence and constrained path-following results [9], [11]. Yu et al. extended these developments with recursive-feasibility and stability analyses [8], while Faulwasser’s monograph provides a comprehensive treatment of optimization-based path following and trajectory tracking [12]. These works provide the theoretical basis for using the path variable as a decision variable rather than a prescribed reference signal.

Real-time nonlinear MPPFC has been demonstrated on industrial robots [13], and later predictive path-following formulations have addressed obstacle avoidance as well as applications to autonomous vessels, cranes, trailer systems, underactuated vehicles, mobile robots, distributed robotic systems, and dynamic-obstacle scenarios [14]– [23]. These works establish MPPFC as a rigorous framework for adaptive path progression under constraints, while motivating the collision-aware predictive-control formulation developed in this dissertation.

1.2.2 Collision Avoidance in Predictive Control

Collision avoidance is essential for autonomous systems operating in

constrained environments. Classical planning-and-control architectures can fail when nonlinear dynamics, actuator limits, or changing obstacle information render a planned path infeasible. Predictive control addresses this limitation by incorporating obstacle-clearance requirements directly into the optimization problem together with system dynamics and constraints.

Early optimization-based methods relied on artificial potential fields, where obstacles generate repulsive costs and goals provide attractive terms. These approaches have been integrated into MPC for autonomous driving and mobile robotics [24]– [26] because of their simplicity. However, they suffer from local minima, oscillations, and limited safety guarantees in cluttered environments [27], since avoidance is enforced through soft costs. In addition, potential-field parameters must be tuned relative to tracking and control-effort terms, which can make the closed-loop behavior sensitive to weighting choices.

Trajectory-optimization methods introduced more systematic collision handling. CHOMP employed signed-distance obstacle costs within covariant gradient optimization [34], while TrajOpt used sequential convex optimization to generate collision-free trajectories [35]. Although effective for trajectory generation, these methods were developed primarily for planning rather than repeated real-time NMPC, where reliable derivatives, warm starts, and consistent closed-loop behavior are required. In receding-horizon control, the optimization must be solved repeatedly under changing initial states and obstacle conditions, so the computational cost of collision modeling becomes more critical.

Within predictive control, collision avoidance is commonly formulated as either a soft penalty or a hard inequality constraint. Soft penalties improve feasibility but cannot guarantee minimum clearance, whereas hard constraints provide explicit safety requirements but may introduce infeasibility. Practical NMPC formulations therefore often combine constraints with slack variables, penalties, or feasibility-restoration mechanisms. This balance is important because an overly strict collision constraint may make the optimization infeasible when the vehicle or robot starts close to an obstacle, while an overly soft penalty may not enforce the required clearance.

In the MPPFC literature, obstacle avoidance has been incorporated through several problem-specific formulations. Sánchez et al. proposed MPPFC formulations using artificial variables and penalties [14], [15]. Alessandretti et al. considered constrained underactuated vehicles [20], Abdelaal and Hahn addressed autonomous vessels in narrow channels [17], and Wen and Pagilla incorporated distance constraints for manipulator trajectory planning [16]. Extensions to distributed robotic systems and generalized n-trailer vehicles have also been reported [22], [23]. These works demonstrate the feasibility of combining path following with obstacle avoidance, but many are tailored to particular system classes, obstacle descriptions, or solver assumptions.

A related approach uses convex duality to formulate collision avoidance. Zhang, Liniger, and Borrelli transformed collision avoidance between unions of convex sets into smooth nonlinear constraints through convex separation conditions

[38]. While this yields differentiable constraints, it introduces additional collision-specific dual variables. In contrast, this dissertation treats the minimum Euclidean distance as the primary geometric quantity and derives its analytical sensitivities directly, providing explicit access to clearance values, gradients, and curvature information.

These studies demonstrate the feasibility of collision-aware predictive control but also reveal limitations. Many methods are tailored to specific robot classes or obstacle geometries, rely heavily on soft penalties, or use numerical derivatives and approximations. This dissertation adopts a minimum-distance formulation in which collision constraints are expressed directly through the separation between convex robot and obstacle bodies, enabling articulated systems to be represented as unions of convex components and allowing smooth aggregation of multiple pairwise distances. The resulting formulation is intended not only to detect or avoid collision, but to provide differentiable clearance constraints suitable for second-order NMPC.

1.2.3 Differentiable Collision Derivatives

Collision-aware NMPC requires geometric models that support repeated distance evaluation and provide information suitable for gradient-based optimization. The key quantity used in this dissertation is the minimum Euclidean distance between convex bodies, which measures clearance and can be incorporated as a nonlinear safety constraint.

For convex sets, distance computation is commonly performed using

specialized proximity algorithms. The Gilbert–Johnson–Keerthi algorithm computes distances using support mappings and the Minkowski difference [28], with extensions to more general convex bodies by Gilbert and Foo [29]. For intersecting objects, the Expanding Polytope Algorithm estimates penetration depth and contact information [30]. Other classical methods include Lin–Canny feature tracking [31] and accelerated proximity techniques based on Voronoi regions and hierarchical structures [32]. Robotics libraries such as the Flexible Collision Library provide efficient implementations of collision and distance queries for geometric primitives and meshes [33].

Signed-distance fields are also widely used in motion planning. CHOMP employed signed-distance obstacle costs [34], while TrajOpt incorporated collision information within sequential convex optimization [35]. Although these methods enable efficient distance and gradient queries, their accuracy depends on map resolution and interpolation, and articulated systems often require additional geometric approximations. For example, a signed-distance field naturally describes the distance from a point to an environment, but a robot link or vehicle body must still be represented through sampled points, bounding volumes, or other approximations if body-level clearance is required.

For articulated robots and vehicles, links and obstacles are commonly represented as unions of convex primitives such as boxes, capsules, ellipsoids, superquadrics, or convex polytopes. Pairwise distances between these bodies provide a unified framework for obstacle avoidance, robot–robot interactions, and

self-collision constraints. However, classical proximity algorithms compute distances but not the analytical sensitivities required by NMPC. Since distance is the solution of an inner optimization problem, its derivatives depend on both rigid-body motion and the variation of the optimal closest-point pair.

This issue is particularly important for second-order optimization. Smooth convex bodies generally yield differentiable distance functions away from contact degeneracies, whereas polyhedral models are only piecewise smooth because closest features can change. Spheres and capsules provide simple distance formulas but can be conservative, while ellipsoids and superquadrics offer smoother representations at the cost of increased computational complexity. Convex polytopes capture geometry accurately but are less suitable for second-order optimization because of nonsmooth feature transitions.

Recent work has increasingly linked collision detection with optimization theory. Montaut et al. interpreted GJK as a Frank–Wolfe optimization method and proposed accelerated collision-detection algorithms [41]. Zimmermann et al. developed differentiable collision avoidance using collision primitives [39], Tracy et al. introduced differentiable collision detection for convex primitives [40], and Montaut et al. later proposed randomized smoothing techniques for collision derivatives [42]. These studies demonstrate growing interest in differentiable proximity computation and provide important alternatives to purely geometric collision queries.

Despite this progress, several limitations remain for the class of problems

considered in this dissertation. Some methods rely on surrogate metrics, smoothing, or auxiliary optimization variables rather than exact Euclidean minimum distance. Others focus primarily on first-order sensitivities and do not provide analytical Hessians suitable for second-order NMPC. Primitive-based formulations may also not generalize directly to smooth parametric surfaces or large articulated systems. These distinctions are important because the collision constraints in a primal-dual interior-point method influence both the constraint Jacobian and the Lagrangian Hessian. If the supplied derivatives do not accurately represent the local geometry of the clearance function, the resulting Newton directions may be less reliable near active collision constraints.

The approach adopted in this dissertation treats minimum-distance computation as a parameter-dependent optimization problem. Body configurations act as parameters, while closest-point coordinates are optimization variables. Under suitable smoothness and regularity assumptions, the closest-point solution varies smoothly with the parameters, allowing distance sensitivities to be obtained by differentiating the optimality conditions. This perspective is supported by classical sensitivity analysis and perturbation theory for nonlinear optimization [43]– [45]. It also provides a direct way to account for the motion of the closest points, rather than approximating the distance derivative using only the current contact normal or local separation direction.

For articulated systems, pose-level derivatives are propagated through the kinematic chain to obtain sensitivities with respect to generalized coordinates. The

resulting Hessians capture body motion, surface curvature, and variations in closest-point locations. Analytical derivatives also support smooth aggregation of multiple pairwise distances and contribute directly to constraint Jacobians, Lagrangian Hessians, and interior-point search directions. Thus, the proposed framework extends convex-distance computation into a derivative model suitable for collision-aware second-order MPPFC. This connection between geometric distance computation and sparse nonlinear optimization is central to the dissertation: distance is not treated only as a collision-checking quantity, but as a differentiable constraint function designed for repeated real-time NMPC solution.

1.2.4 Fast NMPC and Structure-Exploiting Optimization

NMPC can simultaneously handle nonlinear dynamics, constraints, path-following objectives, and collision avoidance, but its practical use is limited by computational cost. At each sampling instant, a constrained nonlinear program must be solved using updated state and obstacle information. For long horizons and cluttered environments, real-time performance depends on exploiting the structure of the optimization problem.

Finite-horizon NMPC problems exhibit a sparse structure: costs are typically stage-separable, dynamics couple only neighboring stages, and most constraints are stage-local. As a result, Jacobian and Hessian matrices contain repeated block patterns that can be exploited to reduce computational effort. Fast NMPC therefore relies on sparsity-aware formulations, tailored linear algebra, and warm-start strategies rather than generic nonlinear programming methods.

Interior-point and structure-exploiting optimization methods have played a central role in fast MPC. Wang and Boyd demonstrated fast online optimization methods for MPC [46], while Diehl, Ferreau, and Haverbeke reviewed real-time iteration schemes and efficient numerical methods for NMPC [47]. Wächter and Biegler developed the interior-point filter line-search framework for large-scale nonlinear programming [49], [51], and Bartlett, Wächter, and Biegler compared active-set and interior-point approaches for MPC applications [50]. Vukov et al. demonstrated the benefits of code generation and tailored solver design for nonlinear MPC [52], while Rawlings, Mayne, and Diehl [53] and Grüne and Pannek [55] provide comprehensive treatments of NMPC theory, computation, and stability. Rao, Wright, and Rawlings also showed how MPC structure can be exploited within interior-point algorithms [54]. Applications such as multicopter attitude control further highlight the need for efficient nonlinear optimization in real-time systems [56].

Modern software frameworks incorporate many of these principles. Acados provides a modular environment for embedded optimal control using multiple shooting, efficient Hessian approximations, and code generation [57], while HPIPM offers high-performance interior-point solvers for structured MPC quadratic programs [58]. These frameworks demonstrate that real-time performance depends not only on the choice of optimization algorithm, but also on preserving the algebraic structure of the optimal-control problem through discretization, derivative assembly, and linear-system solution.

Warm starting is a key component of real-time NMPC because successive optimization problems differ only slightly. Previous solutions can be shifted forward to initialize the next horizon, reducing iteration counts. For interior-point methods, shifted iterates must remain compatible with barrier and slack-variable requirements, making classical work on warm-starting and reoptimization relevant [59]– [61]. In collision-aware problems, warm starting is especially important because the optimizer must respond to changing obstacle sets while maintaining primal feasibility, interiority, and acceptable constraint residuals.

For collision-aware NMPC, computational cost arises not only from solving the KKT system but also from evaluating collision constraints and their derivatives. Finite-difference gradients require many proximity queries, while approximate Hessians may neglect important curvature information. This dissertation therefore maintains stage-local costs and collision constraints, with coupling introduced primarily through system and path-progress dynamics. Analytical distance derivatives contribute only to local Jacobian and Hessian blocks, preserving the sparse banded structure of the Newton system. The resulting primal–dual interior-point method uses barrier terms, slack variables, line-search globalization, and horizon-shift warm starting to exploit this structure efficiently.

The sparse KKT structure of NMPC is especially important in the proposed formulation because collision constraints are stage-local. Each collision term depends on the predicted state, configuration, and obstacle information at a particular prediction stage. Therefore, if the distance derivatives are assembled

carefully, the collision constraints add local Jacobian and Hessian contributions without creating unnecessary coupling across the horizon. The dynamic constraints remain the primary source of inter-stage coupling. This separation is what allows the resulting Newton system to retain the block-banded structure required for efficient solution.

Warm starting also becomes more delicate when collision constraints are included. In standard NMPC, the previous optimal solution can often be shifted forward and used as the initial guess for the next horizon. In collision-aware NMPC, however, the active obstacle set, clearance margins, and nearest body pairs may change between sampling instants. A shifted solution that was feasible at the previous step may be close to a bound, outside an updated clearance constraint, or poorly centered with respect to the interior-point barrier terms. Thus, warm-starting must be combined with interiority restoration, scaling, and line-search safeguards. This motivates the solver design used later in the dissertation, where warm-started iterates are repaired and regularized before computing the primal-dual Newton step.

In summary, fast NMPC depends on preserving sparsity, exploiting warm starts, and using derivative information efficiently. Existing fast MPC and embedded optimal-control methods provide the computational foundation [46]– [61]. This dissertation extends these ideas by integrating analytical minimum-distance collision derivatives into a unified MPPFC framework for real-time collision-aware path-following control.

1.3 Research Gap and Problem Statement

The literature reviewed above demonstrates substantial progress in path-following control, collision-aware predictive control, geometric distance computation, differentiable collision modeling, and fast NMPC algorithms. However, these developments remain only partially integrated. Existing MPPFC approaches provide a rigorous framework for adaptive path progression under nonlinear dynamics and constraints, but they often assume collision-free reference paths or rely on simplified obstacle-avoidance mechanisms. Conversely, many collision-avoidance methods focus on trajectory planning or local motion correction rather than repeated second-order NMPC solution in real time.

A key gap lies in the treatment of collision constraints within the optimization problem. Collision avoidance is naturally expressed through minimum-distance constraints between robot bodies and obstacles, yet standard proximity-query methods typically provide only distances and closest-point information, not the analytical gradients and Hessians required by efficient second-order NMPC. Existing differentiable collision approaches highlight the importance of smooth collision modeling, but they do not fully address the integration of analytical distance derivatives within a unified MPPFC framework solved using sparse second-order optimization methods.

This dissertation addresses this gap by developing a computationally efficient unified nonlinear MPPFC framework that simultaneously performs path following and collision avoidance for nonlinear robotic systems. The proposed approach

combines nonlinear system dynamics, path-progress variables, state and input constraints, and minimum-distance collision constraints within a single finite-horizon optimization problem. Collision avoidance is enforced through differentiable clearance constraints between smooth convex bodies, while the resulting NMPC problem is solved using a structure-exploiting primal–dual interior-point method that incorporates analytical collision gradients and Hessians.

The research problem can therefore be summarized as follows: develop a unified collision-aware MPPFC formulation and solver that leverage analytical minimum-distance derivatives to enable efficient second-order NMPC for nonlinear robotic systems operating in constrained environments. The framework is validated using two representative case studies: an underactuated log-arm system and an articulated tractor–trailer system.

1.4 Research Challenges

The proposed framework introduces several interconnected challenges arising from the interaction of nonlinear dynamics, path-following objectives, collision geometry, derivative computation, and numerical optimization. These challenges are not independent; the modeling choices used for collision avoidance directly influence differentiability, derivative accuracy, solver conditioning, and real-time performance.

The first challenge is the nonconvexity introduced by collision-aware path following. Even when individual robot links and obstacles are represented by convex bodies, the set of system states that satisfies all clearance constraints is generally

nonconvex because the robot configuration, body poses, and obstacle distances depend nonlinearly on the decision variables. As a result, the NMPC problem may admit multiple local solutions, and the computed motion can depend on the warm start, obstacle placement, horizon length, and available actuator authority. This is particularly important in receding-horizon control, where a sequence of local nonlinear programs is solved online rather than a single global planning problem. The formulation must therefore provide useful local clearance information while remaining compatible with warm-starting, line-search globalization, and robust numerical solution.

The second challenge is differentiability of the collision model. Traditional representations based on meshes, polytopes, or occupancy grids often produce only piecewise-smooth distance functions. These representations are effective for collision checking, but they are less suitable for Newton-type optimization methods that require consistent first- and second-order derivative information. Smooth geometric representations are therefore needed to balance geometric fidelity with the differentiability requirements of second-order NMPC.

The third challenge is analytical derivative computation. The minimum distance between two convex bodies is itself the solution of an optimization problem, and its derivatives depend on how the closest points vary with system configuration. Numerical differentiation is expensive because each derivative direction requires additional distance evaluations, and the resulting approximations can be inaccurate near contact or near changes in the closest-point configuration.

Analytical sensitivity analysis is therefore essential for efficient and reliable second-order optimization.

The fourth challenge is scalability. In articulated systems, collision avoidance may require evaluating many body–obstacle distance constraints across multiple prediction stages. The number of pairwise interactions can grow quickly with the number of robot links, static obstacles, dynamic obstacles, and prediction steps. Without careful formulation, the computational burden can grow faster than the real-time controller can tolerate. Preserving sparsity and stage-wise structure is therefore critical for practical implementation.

The fifth challenge is numerical robustness of the primal–dual interior-point solve. Collision-aware MPPFC combines nonlinear dynamics, inequality constraints, barrier terms, softened constraints, and feasibility-restoring slack variables. These elements can lead to ill-conditioned KKT systems, especially when constraints become active, slack variables become small, or scaling varies significantly across physical units. Stable solution requires regularization, scaling, fraction-to-the-boundary step control, line-search globalization, and warm-start management.

The sixth challenge is feasibility under changing obstacle information. In closed-loop operation, the obstacle set may change from one sampling instant to the next. A previously feasible predicted trajectory can become infeasible when a new obstacle enters the horizon or when a moving obstacle changes the local clearance requirements. The controller must therefore handle temporary infeasibility without producing unstable commands. This motivates the use of softened constraints and

feasibility-restoring mechanisms in the proposed formulation.

Finally, path following and obstacle avoidance must be balanced within a single optimization problem. Excessive emphasis on path tracking can prevent necessary avoidance maneuvers, while overly conservative collision avoidance can hinder path progression. The controller must therefore optimize path progress, tracking performance, dynamic feasibility, and obstacle clearance simultaneously. This balance is one of the main reasons for adopting a unified MPPFC formulation rather than a separated planner-tracker architecture.

These challenges define the technical scope of the dissertation and motivate the collision-aware MPPFC framework developed in the following chapters.

1.5 Proposed Solution and Dissertation Contributions

This dissertation proposes a unified nonlinear Model Predictive Path-Following Control framework for robotic systems operating in constrained environments. The formulation treats path following, obstacle avoidance, nonlinear dynamics, actuator limits, and path progression as parts of a single finite-horizon optimization problem. It allows the controller to adjust both physical motion and progression along the reference path while enforcing collision-clearance constraints.

The framework has three core components. First, augmented path variables allow the path parameter and its rate of evolution to be optimized together with system states and controls. This allows the controller to reduce path speed, pause progression, or resume forward motion according to feasibility and obstacle-clearance requirements. Second, collision avoidance is expressed using minimum-

distance constraints between smooth convex representations of robot bodies and obstacles. Third, the resulting nonlinear program is solved using a structure-exploiting primal-dual interior-point method that preserves the sparse horizon structure of the MPPFC problem.

The first contribution is a differentiable geometric framework for minimum-distance constraints between smooth convex bodies. Robot links, obstacles, and vehicle components are represented as convex parametric surfaces or smooth convex approximations, and pairwise clearance is expressed through Euclidean minimum distance. This provides a direct safety measure while remaining compatible with articulated systems represented as unions of convex components.

The second contribution is the derivation of analytical gradients and Hessians of the minimum-distance function. By differentiating the closest-point optimality conditions, the dissertation obtains first- and second-order sensitivities that account for rigid-body motion, closest-point variation, and surface curvature. These derivatives are propagated through articulated kinematic chains to obtain sensitivities with respect to generalized coordinates and predicted states. This avoids finite-difference collision Jacobians and provides curvature information that can be used by second-order optimization methods.

The third contribution is a unified collision-aware MPPFC formulation. The formulation combines nonlinear system dynamics, path-progress dynamics, path-following objectives, state and input constraints, and obstacle-clearance requirements within a single finite-horizon optimization problem. The path variable

is optimized rather than prescribed, allowing the controller to adapt its progress when constraints become active. This enables path recovery, path following, collision avoidance, and feasible control generation to be handled within the same predictive optimization problem.

The fourth contribution is a structure-exploiting primal–dual interior-point solver for the resulting NMPC problem. The solver uses analytical derivatives, softened constraints, barrier terms, warm starting, and sparse KKT-system structure to improve computational efficiency. Collision-constraint derivatives contribute to stage-local Jacobian and Hessian blocks, preserving the sparse structure needed for repeated solution. This is essential for collision-aware MPPFC because both derivative assembly and KKT-system solution can become computational bottlenecks when many obstacle interactions are present.

The fifth contribution is validation on representative nonlinear systems. The underactuated log-arm system evaluates constrained nonlinear path-following performance, while the articulated tractor–trailer system evaluates collision-aware path following in a higher-dimensional setting involving articulation, actuator limits, static obstacles, dynamic obstacles, and multiple body–obstacle interactions. These examples are selected not as isolated applications, but as complementary validation cases that exercise different aspects of the proposed framework.

Together, these contributions connect geometric collision modeling, analytical sensitivity analysis, path-following control, and sparse interior-point optimization. The result is a framework for embedding differentiable minimum-

distance constraints directly into a unified MPPFC problem and solving the resulting nonlinear program using a structure-exploiting second-order optimization method.

1.6 Dissertation Organization

The remainder of this dissertation is organized as follows.

Chapter 2 presents the mathematical preliminaries used throughout the dissertation, including matrix calculus notation, differential identities, vectorization conventions, and derivative tools required for later analytical developments.

Chapter 3 develops the system-level modeling and optimization framework. It defines robotic systems using rigid-body transformations and kinematic chains, describes configuration-dependent body representations for collision modeling, and formulates the unified nonlinear model predictive control problem.

Chapter 4 derives analytical gradients and Hessians of minimum-distance functions between smooth convex bodies by differentiating the optimality conditions of the closest-point problem with respect to pose and configuration variables.

Chapter 5 presents the smooth path-generation framework used to construct reference paths from waypoint data. Composite Bézier curves and a fast waypoint-fitting algorithm are used to obtain path representations suitable for real-time path-following control.

Chapter 6 develops the structure-exploiting primal–dual interior-point algorithm for the unified collision-aware MPPFC problem, including the barrier formulation, softened constraints, KKT system, scaling, regularization, globalization,

warm starting, and sparse linear algebra.

Chapter 7 evaluates the proposed framework on an articulated tractor-trailer system in cluttered environments, including static and dynamic obstacle-avoidance scenarios.

Chapter 8 presents the second application study on the underactuated planar log-arm system, evaluating the proposed path-following and optimization framework on a nonlinear, underactuated mechanical system benchmark.

Chapter 9 concludes the dissertation by summarizing the main findings, discussing limitations, and identifying future work on nonsmooth geometry handling, embedded implementation, feasibility guarantees, and experimental validation.

CHAPTER 2 - MATHEMATICAL BACKGROUND

This chapter summarizes the mathematical notation and identities used throughout the dissertation. The focus is on linear algebra, structured matrix derivatives, Kronecker-product identities, and matrix calculus tools that are repeatedly used in the derivation of differentiable collision constraints, analytical gradients and Hessians, and structure-exploiting nonlinear model predictive control algorithms. The results presented here are standard unless otherwise noted; therefore, detailed textbook proofs are omitted except where a specialized notation is introduced for later use.

2.1 Linear Algebra Preliminaries and Matrix Identities

This section introduces the linear algebra notation used in later chapters. Particular emphasis is placed on inertia, Schur complements, Kronecker products, vectorization, and structured block operations, since these tools appear repeatedly in the analysis of constrained optimization systems and derivative expressions.

2.1.1 Inertia of a Matrix

Let $M \in \mathbb{C}^{n \times n}$ be a Hermitian matrix, where $\mathbb{C}$ denote the field of complex numbers. Denote by $n_+(M)$, $n_-(M)$, and $n_0(M)$ the number of positive, negative and zero eigenvalues of M . The inertia of M is defined as

$$In(M) \triangleq (n_+(M), n_-(M), n_0(M)) \quad (2.1)$$

The inertia of a Hermitian matrix is invariant under nonsingular congruence transformations, i.e., for any nonsingular matrix T ,

$$In(M) = In(T^*MT)$$

where $(\cdot)^*$ denotes the conjugate transpose. This invariance is central in definiteness analysis, saddle-point systems, and second-order optimality conditions for constrained optimization problems [62].

2.1.2 Schur Complements and Inertia Additivity

Consider the block Hermitian block matrix

$$H = \begin{bmatrix} A & B \\ B^* & C \end{bmatrix}$$

where $A \in \mathbb{C}^{p \times p}$, $B \in \mathbb{C}^{p \times q}$, and $C \in \mathbb{C}^{q \times q}$. If A is invertible, the Schur complement of A in H is

$$H/A \triangleq C - B^*A^{-1}B \quad (2.2)$$

If C is invertible, the Schur complement of C in H is defined as

$$H/C \triangleq A - BC^{-1}B^* \quad (2.3)$$

The Haynsworth inertia additivity formula states that

$$\text{In}(H) = \text{In}(A) + \text{In}(H/A), \quad \text{if } A \text{ is invertible} \quad (2.4)$$

and analogously,

$$\text{In}(H) = \text{In}(C) + \text{In}(H/C), \quad \text{if } C \text{ is invertible} \quad (2.5)$$

This result relates the inertia of a block matrix to the inertia of one diagonal block and its Schur complement. It is particularly useful for analyzing KKT systems and saddle-point matrices arising in equality-constrained and interior-point optimization [63], [64].

2.1.3 Block Matrix Inversion and Determinant Identities

Consider the block matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

with conformable block dimensions. If A is invertible, define

$$S_A \triangleq D - CA^{-1}B$$

If A and S_A are invertible, then

$$M^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BS_A^{-1}CA^{-1} & -A^{-1}BS_A^{-1} \\ -S_A^{-1}CA^{-1} & S_A^{-1} \end{bmatrix} \quad (2.6)$$

If D is invertible, define

$$S_D := A - BD^{-1}C$$

If D and S_D are invertible, then

$$M^{-1} = \begin{bmatrix} S_D^{-1} & -S_D^{-1}BD^{-1} \\ -D^{-1}CS_D^{-1} & D^{-1} + D^{-1}CS_D^{-1}BD^{-1} \end{bmatrix} \quad (2.7)$$

If both A and D are invertible, then the same inverse may also be written in a factorized Schur-complement form as

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} S_D^{-1} & 0 \\ 0 & S_A^{-1} \end{bmatrix} \begin{bmatrix} I & -BD^{-1} \\ -CA^{-1} & I \end{bmatrix} \quad (2.8)$$

These identities follow directly from block Gaussian elimination and Schur-complement factorization [64], [65].

For the same block matrix M , the determinant can be expressed using Schur complements. If A is invertible,

$$\det(M) = \det(A) \det(S_A) \quad (2.9)$$

If D is invertible,

$$\det(M) = \det(D) \det(S_D) \quad (2.10)$$

These determinant formulas follow directly from block Gaussian elimination and are

useful in the analysis of block factorizations and nonsingularity conditions [64],[65].

2.1.4 Kronecker and Hadamard Product

Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$. The Kronecker product $A \otimes B \in \mathbb{R}^{(mp) \times (nq)}$ is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \quad (2.11)$$

A fundamental property is the mixed-product identity

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \quad (2.12)$$

whenever the products AC and BD are defined [62], [65]. A useful consequence of the mixed product identity is:

$$(I_p \otimes M)(N \otimes I_n) = N \otimes M \quad (2.13)$$

Another frequently used identity is

$$I_m \otimes (I_n \otimes A) = I_{mn} \otimes A \quad (2.14)$$

These identities are used repeatedly when arranging block derivatives and horizon-stacked NMPC quantities.

Let $A, B \in \mathbb{R}^{m \times n}$. The Hadamard product, also known as the Schur product, is defined elementwise as

$$[A \odot B]_{ij} = A_{ij}B_{ij} \quad (2.15)$$

The Hadamard product is commutative and associative and distributes over addition. It appears in this dissertation mainly in elementwise scaling operations and structured matrix derivative expressions [62].

2.1.5 Special Block-wise Product

Certain second-order derivative expressions involve repeated block-wise multiplication between horizontally and vertically partitioned matrices. To represent these expressions compactly, define the following custom block-wise product. Let

$$M = [M_1 \quad \cdots \quad M_m] \in \mathbb{R}^{p \times (mq)}, \quad N = \begin{bmatrix} N_1 \\ \vdots \\ N_n \end{bmatrix} \in \mathbb{R}^{(nq) \times r}$$

where $M_i \in \mathbb{R}^{p \times q}$, $N_j \in \mathbb{R}^{q \times r}$. Define

$$M \circledast N \triangleq \begin{bmatrix} M_1 N_1 & \cdots & M_m N_1 \\ \vdots & \ddots & \vdots \\ M_1 N_n & \cdots & M_m N_n \end{bmatrix} \in \mathbb{R}^{(np) \times (mr)} \quad (2.16)$$

This operator is distinct from the Kronecker, Hadamard, and Khatri–Rao products; it is introduced only as compact notation for the block-structured mixed terms that arise in second-order matrix derivative formulas. By construction,

$$N^\top \circledast M^\top = (M \circledast N)^\top \quad (2.17)$$

2.1.6 Vectorization and Block Vectorization

For $M \in \mathbb{R}^{p \times q}$, the vectorization operator stacks the columns of M into a single vector

$$\text{vec}(M) = [m_{11} \quad \cdots \quad m_{p1} \quad m_{1q} \quad \cdots \quad m_{pq}]^\top \quad (2.18)$$

A standard identity involving vectorization is

$$\text{vec}(ABC) = (C^\top \otimes A) \text{vec}(B) \quad (2.19)$$

whenever the product ABC is defined [62], [66]. For a block matrix M with blocks

$M_{ij} \in \mathbb{R}^{m \times n}$, define the block vectorization operator

$$vec_b(M) \triangleq \begin{bmatrix} M_{11} \\ \vdots \\ M_{p1} \\ \vdots \\ M_{1q} \\ \vdots \\ M_{pq} \end{bmatrix} \quad (2.20)$$

This representation preserves block structure and is useful when constructing horizon-stacked vectors and block-sparse derivative arrays in nonlinear model predictive control.

2.1.7 Diagonal Scaling Identities

Let $v \in \mathbb{R}^m$, $A \in \mathbb{R}^{mn \times mn}$ and let $e_n \in \mathbb{R}^n$ denote the vector of ones. Then

$$diag(v)A = (e_n^\top \otimes v) \odot A \quad (2.21)$$

If $B \in \mathbb{R}^{n \times n}$, then

$$(e_n^\top \otimes v) \odot (AB) = ((e_n^\top \otimes v) \odot A)B \quad (2.22)$$

These identities are useful for representing scaling operations in elementwise and block-structured form.

2.1.8 Permutation Matrices

Let $I_{(i,j)}^{(m,n)} \in \mathbb{R}^{m \times n}$ denote the elementary matrix whose entries are all zero except for a single unity entry in the i -th row and j -th column. Define the commutation matrix $P_{(m,n)} \in \mathbb{R}^{mn \times mn}$:

$$P_{(m,n)} \triangleq \begin{bmatrix} I_{(1,1)}^{(m,n)} & I_{(2,1)}^{(m,n)} & \cdots & I_{(m,1)}^{(m,n)} \\ I_{(1,2)}^{(m,n)} & I_{(2,2)}^{(m,n)} & \cdots & I_{(m,2)}^{(m,n)} \\ \vdots & \vdots & \ddots & \vdots \\ I_{(1,n)}^{(m,n)} & I_{(2,n)}^{(m,n)} & \cdots & I_{(m,n)}^{(m,n)} \end{bmatrix} \quad (2.23)$$

The matrix $P_{(m,n)}$ is a permutation matrix. An equivalent representation of $P_{(m,n)}$ is given by Kronecker-sum form

$$P_{(m,n)} = \sum_{i=1}^m \sum_{j=1}^n \left(I_{(j,i)}^{(n,m)} \otimes I_{(i,j)}^{(m,n)} \right)$$

It satisfies the standard commutation identity

$$P_{(m,n)} \text{vec}(X) = \text{vec}(X^\top), \quad \text{for any } X \in \mathbb{R}^{m \times n} \quad (2.24)$$

and its inverse is

$$P_{(m,n)}^{-1} = P_{(m,n)}^\top = P_{(n,m)} \quad (2.25)$$

In particular, for $m = n$, the matrix is symmetric. For vectors $v \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$,

$$P_{(m,n)}(v \otimes w) = w \otimes v \quad (2.26)$$

For a block-partitioned matrix

$$Y = [Y_1 \quad \cdots \quad Y_m], \quad Y_i \in \mathbb{R}^{p \times q}$$

and a matrix $X \in \mathbb{R}^{n \times n}$, the Kronecker product distributes over the horizontal block partition:

$$X \otimes Y = [X \otimes Y_1 \quad \cdots \quad X \otimes Y_m] \quad (2.27)$$

Finally, for

$$A = [A_1 \quad \cdots \quad A_m] \in \mathbb{R}^{p \times m}, \quad v = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \in \mathbb{R}^m$$

where $A_i \in \mathbb{R}^p$ is a column vector and each $v_i \in \mathbb{R}$ is a scalar. Then the following identity holds:

$$I_n \otimes (Av) = (I_n \otimes A)(I_n \otimes v) \quad (2.28)$$

2.2 Derivatives of Matrix-Valued Functions

Let

$$F(x): \mathbb{R}^n \rightarrow \mathbb{R}^{m \times p}, \quad x = [x_1 \quad \cdots \quad x_n]^T$$

be a smooth matrix-valued function. The objective of this section is to define first- and second-order derivatives in a form that is suitable for matrix calculus while preserving the natural matrix structure of F . This is useful because several intermediate quantities and derivative objects in later chapters, including rigid-body transformations, closest-point mappings, dynamics-related matrices, collision-distance derivative arrays, and KKT derivative blocks, naturally have matrix-valued or block-matrix structure.

2.2.1 First-Order Derivative

For each $k \in \{1, \dots, n\}$ and a scalar x_k , define the matrix partial derivative

$$G_k := \frac{\partial F(x)}{\partial x_k} \in \mathbb{R}^{m \times p}, \quad [G_k]_{ij} = \frac{\partial F_{ij}(x)}{\partial x_k}$$

The gradient of F with respect to x is represented by vertically stacking these partial derivative matrices:

$$\nabla_x F(x) \triangleq \begin{bmatrix} G_1 \\ \vdots \\ G_n \end{bmatrix} \in \mathbb{R}^{mn \times p} \quad (2.29)$$

An alternative arrangement stacks the same blocks horizontally:

$$D_x F(x) \triangleq [G_1 \quad \cdots \quad G_n] \in \mathbb{R}^{m \times pn} \quad (2.30)$$

The operators ∇_x and D_x contain the same first-order derivative information but use different block arrangements. The ∇_x convention is useful for vertically stacked derivative propagation, while D_x is useful for Jacobian-like product rules and right-multiplied expressions.

2.2.2 Second-Order Derivative

For each pair $(k, l) \in \{1, \dots, n\}^2$, define:

$$H_{kl} \triangleq \frac{\partial G_k(x)}{\partial x_l} = \frac{\partial^2 F(x)}{\partial x_l \partial x_k} \in \mathbb{R}^{m \times p}$$

The Hessian of F is represented as the block matrix:

$$\nabla_{xx}^2 F = \begin{bmatrix} H_{11} & \cdots & H_{1n} \\ \vdots & \ddots & \vdots \\ H_{n1} & \cdots & H_{nn} \end{bmatrix} \in \mathbb{R}^{mn \times pn} \quad (2.31)$$

where each $H_{kl} \in \mathbb{R}^{m \times p}$. Equivalently,

$$\nabla_{xx}^2 F(x) = \nabla_x (D_x F(x)) \quad (2.32)$$

This representation is a matrix-structured analogue of the classical scalar Hessian and is used later to express second-order derivatives without prematurely vectorizing the underlying matrix-valued functions.

2.2.3 Third-Order Derivative

The same stacking convention is used for third-order derivatives. Let $F: \mathbb{R}^n \rightarrow \mathbb{R}^{m \times p}$

be thrice continuously differentiable. The third derivative of F is defined by differentiating the transpose of the second-derivative array:

$$\nabla_{xxx}^3 F \triangleq \nabla_x (\nabla_{xx}^2 F)^\top \quad (2.33)$$

The corresponding identity for the transpose of F is

$$\nabla_x (\nabla_{xx}^2 F^\top)^\top = \nabla_{xxx}^3 F^\top.$$

This convention is used later when differentiating second-order matrix-structured derivative terms. It is a block-matrix representation of the corresponding third-

order derivative tensor and preserves the same stacking convention used in (2.29)–(2.32).

Also, the derivative of a transposed first-derivative array can be written in terms of the block-vectorized second derivative as

$$\nabla_x(\nabla_x F^\top) = \text{vec}_b(\nabla_{xx}^2 F)^\top \quad (2.34)$$

This identity follows from the definitions of ∇_x , D_x , and $\text{vec}_b(\cdot)$: differentiating the transposed first-derivative blocks rearranges the second-derivative blocks into the block-vectorized transpose form.

2.2.4 Tensor Unfolding

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ be an N -way tensor with entries $\mathcal{A}_{i_1, i_2, \dots, i_N}$. Tensor unfolding (or matricization) rearranges tensor entries into a matrix by grouping selected tensor modes into row indices and the remaining modes into column indices.

For *mode- n* unfolding, the n – *th* index defines the rows, while all remaining indices define the columns. The unfolded matrix has size:

$$A_{(n)} \in \mathbb{R}^{I_n \times (\prod_{k \neq n} I_k)}$$

More generally, if R and C are disjoint subsets of $\{1, 2, \dots, N\}$ with $R \cup C = \{1, 2, \dots, N\}$.

The (R, C) -unfolding has size

$$A_{(R, C)} \in \mathbb{R}^{(\prod_{r \in R} I_r) \times (\prod_{c \in C} I_c)}$$

Tensor unfolding and matricization operations are standard tools in multilinear algebra [67]–[69].

2.2.5 Relationship to Tensor Notation

The first derivative can be represented as the third-order tensor

$$T_{ijk}^{(1)} = \frac{\partial F_{ij}(x)}{\partial x_k}, \quad i \in \{1, \dots, m\}, \quad j \in \{1, \dots, p\}, \quad k \in \{1, \dots, n\}$$

with $T^{(1)} \in \mathbb{R}^{m \times p \times n}$. The derivative operation in (2.29) is obtained by mode-3 unfolding

$$\nabla_x F(x) = \text{unfold}_{(i,k),j}(T^{(1)}) \quad (2.35)$$

where the composite index (i, k) forms the row index and j remains the column index. This transformation is a permutation of tensor entries and therefore preserves all derivative information. Similarly, the second derivative can be represented by the fourth-order tensor

$$T_{ijkl}^{(2)} := \frac{\partial^2 F_{ij}(x)}{\partial x_k \partial x_l}, \quad i \in \{1, \dots, m\}, \quad j \in \{1, \dots, p\}, \quad k, l \in \{1, \dots, n\}$$

with dimensions $T^{(2)} \in \mathbb{R}^{m \times p \times n \times n}$. The block-matrix Hessian representation $\nabla_{xx}^2 F$ is obtained from $T^{(2)}$ by a mode-(3,4) unfolding:

$$\nabla_{xx}^2 F(x) = \text{unfold}_{(i,k),(j,l)}(T^{(2)}) \quad (2.36)$$

2.2.6 Literature Usage

The derivative representation used here is consistent with several standard viewpoints in applied mathematics. In matrix differential calculus, the derivative of a matrix-valued function is often expressed using vectorization and Kronecker products [66]. The conventional Jacobian of F can be recovered from blocks G_k as

$$J_F = [\text{vec}(G_1), \dots, \text{vec}(G_n)]$$

In continuum mechanics, derivatives of tensor-valued quantities are

commonly written in index form, such as $\partial F_{ij} / \partial x_k$ and the present notation corresponds to a matricized version of that derivative tensor [70]. In automatic differentiation, forward-mode propagation computes derivatives with respect to each input direction separately, which is also consistent with the block-stacking convention in (2.29) [71].

2.2.7 Advantages of the Derivative Convention for MPPFC

The main advantage of the proposed notation is that it preserves the natural matrix and block structure of the original function. Instead of flattening $F(x)$ into a vector at the beginning of every derivation, the derivative blocks $G_k = \partial F / \partial x_k$ remain visible. This allows product rules, inverse rules, and chain rules to be written in a form that is compatible with the dimensions and sparsity patterns of the original problem.

For a perturbation $h \in \mathbb{R}^n$, directional derivatives can be evaluated directly as weighted combinations of the matrix blocks G_k . When conversion to a standard vectorized Jacobian is needed, each block may be vectorized and stacked as in the conventional matrix calculus representation. Therefore, the notation used here is not a different derivative concept; it is a structured arrangement of the same derivative information

2.2.8 Relevance to Nonlinear MPPFC

In the NMPC framework developed in this dissertation, several intermediate quantities and derivative objects naturally have matrix-valued or block-matrix structure. Examples include the mass matrix and other dynamics-related matrices,

rigid-body transformation derivatives, first derivatives of collision-distance quantities with respect to configuration variables, and the block derivatives that appear in the first-order KKT conditions. Although the final constraint functions used by the optimizer are scalar- or vector-valued, their analytical evaluation often depends on matrix-valued intermediate mappings and structured derivative arrays.

This is particularly important in real-time NMPC implementations, where these quantities must be evaluated repeatedly during linearization, quadratic approximation, and primal-dual interior-point iterations. Preserving the matrix and block structure of the intermediate derivatives reduces unnecessary reshaping, clarifies sparsity patterns, and supports efficient assembly of Jacobians, Hessians, and KKT-system components.

2.3 Derivative Identities for Matrix-Valued Functions

This section states derivative identities for the structured operators ∇_x and D_x . These identities are used repeatedly in later derivations involving matrix-valued intermediate mappings, structured derivative arrays, inverse matrices, and second-order expressions that arise in analytical differentiation and NMPC formulations.

Let $M: \mathbb{R}^n \rightarrow \mathbb{R}^{m \times p}$ be a differentiable matrix-valued function. Then:

$$\nabla_x M^\top = (D_x M)^\top \quad (2.37)$$

$$D_x M^\top = (\nabla_x M)^\top \quad (2.38)$$

These identities follow directly from elementwise differentiation and the definitions of the two derivative arrangements. If M is twice continuously differentiable, then

$$\nabla_{xx}^2 M^\top = (\nabla_{xx}^2 M)^\top \quad (2.39)$$

The transpose in (2.39) is understood blockwise. The identity follows from equality of mixed partial derivatives. More generally, if M is twice continuously differentiable with respect to two variable vectors x and u , then

$$\nabla_{xu}^2 M^\top = (\nabla_{ux}^2 M)^\top \quad (2.40)$$

This identity also follows from the equality of mixed partial derivatives.

2.3.1 Product Rule

Let

$$M(x): \mathbb{R}^n \rightarrow \mathbb{R}^{p \times q}, \quad N(x): \mathbb{R}^n \rightarrow \mathbb{R}^{q \times r}$$

be two matrix-valued continuously differentiable functions of the vector $x = (x_1, \dots, x_n)$. Then the following first order derivative identities hold:

$$\nabla_x(MN) = (\nabla_x M)N + (I_n \otimes M)(\nabla_x N) \quad (2.41)$$

$$D_x(MN) = (D_x M)(I_n \otimes N) + M(D_x N) \quad (2.42)$$

If M and N are twice continuously differentiable, then

$$\nabla_x^2(MN) = (\nabla_x^2 M)(I_n \otimes N) + (D_x M) \odot (\nabla_x N) + (\nabla_x M)(D_x N) + (I_n \otimes M)(\nabla_x^2 N) \quad (2.43)$$

where $\otimes$ denotes the Kronecker product, and $\odot$ denotes the block-wise product defined in section 2.1.

Proof

For each component x_i , the ordinary product rule gives,

$$\frac{\partial(MN)}{\partial x_i} = \frac{\partial M}{\partial x_i} N + M \frac{\partial N}{\partial x_i}$$

Stacking these equations vertically according to the definition of ∇_x yields

$$\nabla_x(MN) = \begin{bmatrix} \frac{\partial M}{\partial x_1} N \\ \vdots \\ \frac{\partial M}{\partial x_n} N \end{bmatrix} + \begin{bmatrix} M \frac{\partial N}{\partial x_1} \\ \vdots \\ M \frac{\partial N}{\partial x_n} \end{bmatrix}$$

The second term may be written compactly using the Kronecker product as $(I_n \otimes M)(\nabla_x N)$. This proves (2.41). Horizontal stacking of the same componentwise product rule gives (2.42).

To obtain the second-order identity, differentiate (2.42) using the structured gradient operator:

$$\nabla_x^2(MN) = \nabla_x((D_x M)(I_n \otimes N)) + \nabla_x(M(D_x N))$$

Applying the first-order product rule to each term gives four contributions: the curvature of M , the mixed derivative term involving $D_x M$ and $\nabla_x N$, the mixed derivative term involving $\nabla_x M$ and $D_x N$, and the curvature of N . Arranging these terms using the definitions of ∇_x , D_x , and $\odot$ gives (2.43).

2.3.2 Weighted Contraction of Second Derivatives

Let $g: \mathbb{R}^n \rightarrow \mathbb{R}^q$ be twice continuously differentiable and let $w \in \mathbb{R}^q$. The contraction of the structured second derivative of g with the weight vector w can be written as

$$(I_n \otimes w^\top) \nabla_{xx}^2 g.$$

For implementation, this expression is equivalently evaluated as the weighted sum of component Hessians:

$$(I_n \otimes w^\top) \nabla_{xx}^2 g = \sum_{i=1}^q w_i \nabla_{xx}^2 g_i \quad (2.44)$$

This summation form is preferred in numerical implementations because it avoids explicitly forming the Kronecker product and the full stacked second-derivative array. Instead, each component Hessian $\nabla_{xx}^2 g_i$ can be accumulated directly with scalar weight w_i .

2.3.3 Derivative of Matrix Inverse

Let $M(x) \in \mathbb{R}^{p \times p}$ be a twice continuously differentiable matrix-valued function of $x \in \mathbb{R}^n$, and assume that $M(x)$ is invertible on domain of interest. Then the first and second order derivatives of $M^{-1}(x)$ are given by:

$$\nabla_x M^{-1} = -(I_n \otimes M^{-1})(\nabla_x M)M^{-1} \quad (2.45)$$

$$D_x M^{-1} = -M^{-1}(D_x M)(I_n \otimes M^{-1}) \quad (2.46)$$

$$\nabla_{xx}^2 M^{-1} = -(I_n \otimes M^{-1})(\nabla_{xx}^2 M - (D_x M) \odot M^{-1} \odot (\nabla_x M) - (\nabla_x M)M^{-1}(D_x M))(I_n \otimes M^{-1}) \quad (2.47)$$

Proof

The proof follows by differentiating the identity

$$M(x)M^{-1}(x) = I$$

Applying the gradient operator ∇_x and using the matrix product rule above gives

$$\nabla_x(MM^{-1}) = (\nabla_x M)M^{-1} + (I_n \otimes M)(\nabla_x M^{-1}) = 0$$

Solving for $\nabla_x M^{-1}$ yields (2.45). Similarly, applying the operator D_x gives

$$D_x(MM^{-1}) = (D_x M)(I_n \otimes M^{-1}) + M(D_x M^{-1}) = 0$$

which gives (2.46) after left multiplication M^{-1} .

For the second-order derivative, differentiate $D_x(MM^{-1}) = 0$ using the matrix product rule. This gives

$$\nabla_{xx}^2(MM^{-1}) = 0 = (\nabla_{xx}^2 M)(I_n \otimes M^{-1}) + (D_x M) \odot (\nabla_x M^{-1}) + (\nabla_x M)(D_x M^{-1}) + (I_n \otimes M)(\nabla_x^2 M^{-1})$$

Rearranging

$$(I_n \otimes M)(\nabla_{xx}^2 M^{-1}) = -(\nabla_{xx}^2 M)(I_n \otimes M^{-1}) - (D_x M) \odot (\nabla_x M^{-1}) - (\nabla_x M)(D_x M^{-1}) \quad (2.48)$$

Substituting the first-order expressions (2.45) - (2.46),

$$(D_x M) \odot (\nabla_x M^{-1}) = -(D_x M) \odot ((I_n \otimes M^{-1})(\nabla_x M)M^{-1})$$

and

$$(\nabla_x M)(D_x M^{-1}) = -(\nabla_x M)M^{-1}(D_x M)(I_n \otimes M^{-1})$$

The first mixed term requires one additional simplification. From the definitions of $D_x M$ and $\nabla_x M$,

$$D_x M = \begin{bmatrix} \frac{\partial M}{\partial x_1} & \cdots & \frac{\partial M}{\partial x_n} \end{bmatrix}, \quad \nabla_x M = \begin{bmatrix} \frac{\partial M}{\partial x_1} \\ \vdots \\ \frac{\partial M}{\partial x_n} \end{bmatrix}$$

Therefore,

$$(I_n \otimes M^{-1})(\nabla_x M)M^{-1} = \begin{bmatrix} M^{-1} \frac{\partial M}{\partial x_1} M^{-1} \\ \vdots \\ M^{-1} \frac{\partial M}{\partial x_n} M^{-1} \end{bmatrix}$$

Using the definition of the block-wise product $\odot$, the product

$$(D_x M) \odot ((I_n \otimes M^{-1})(\nabla_x M)M^{-1})$$

collects all block products from

$$\frac{\partial M}{\partial x_i} M^{-1} \frac{\partial M}{\partial x_j} M^{-1}$$

Right-multiplication by $I_n \otimes M$ acts blockwise and cancels the trailing factor M^{-1} , yielding

$$\left((D_x M) \circledast ((I_n \otimes M^{-1})(\nabla_x M)M^{-1}) \right) (I_n \otimes M) = (D_x M) \circledast M^{-1} \circledast (\nabla_x M)$$

Substituting these expressions into (2.48), left multiplying by $I_n \otimes M^{-1}$, and collecting the right factor $I_n \otimes M^{-1}$, yields

$$\begin{aligned} \nabla_{xx}^2 M^{-1} &= -(I_n \otimes M^{-1})(\nabla_{xx}^2 M - (D_x M) \circledast M^{-1} \circledast (\nabla_x M) \\ &\quad - (\nabla_x M)M^{-1}(D_x M))(I_n \otimes M^{-1}) \end{aligned}$$

which proves (2.47).

The result generalizes the classical scalar identity

$$\nabla_{xx}^2 f^{-1} = -f^{-2} \nabla_{xx}^2 f + 2f^{-3} (\nabla_x f)(\nabla_x f)^\top$$

to matrix-valued functions while preserving matrix structure.

2.3.4 Chain Rule for Composite Mappings

Consider the composite mapping

$$u(x) = f^{(l)} \circ f^{(l-1)} \circ \dots \circ f^{(1)}(x)$$

where

$$z^{(0)} \triangleq x \in \mathbb{R}^{n_0} \tag{2.49}$$

$$z^{(k)} := f^{(k)} \circ z^{(k-1)} \in \mathbb{R}^{n_k}, \quad k = 1, \dots, l, \tag{2.50}$$

with

$$f^{(k)}: \mathbb{R}^{n_{k-1}} \rightarrow \mathbb{R}^{n_k}, \quad f^{(k)} \in \mathbb{C}^2 \tag{2.51}$$

Thus, the composite mapping is

$$u(x) \in \mathbb{R}^{n_l} \tag{2.52}$$

First Derivatives of the Composite Mapping

Define the Jacobians of each layer

$$J_k := D_{z^{(k-1)}} f^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}, \quad k = 1, \dots, l \quad (2.53)$$

The Jacobian of the composition is

$$D_x u(x) = J_l J_{l-1} \dots J_1 \in \mathbb{R}^{n_l \times n_0} \quad (2.54)$$

Equivalently,

$$D_x u(x) = \left(D_{f^{(l-1)}} f^{(l)} \right) \left(D_{f^{(l-2)}} f^{(l-1)} \right) \dots \left(D_x f^{(1)} \right) \in \mathbb{R}^{n_l \times n_0} \quad (2.55)$$

Using the gradient-stacking convention introduced earlier,

$$\nabla_x u(x) = \text{vec}(D_x u(x)) \in \mathbb{R}^{(n_l n_0) \times 1} \quad (2.56)$$

Using (2.19) and setting $C = J_1$, $B = J_l \dots J_2$ and $A = I_{n_l}$, we obtain

$$\nabla_x u(x) = (J_1^\top \otimes I_{n_l}) \text{vec}(J_l \dots J_2)$$

Applying the formula recursively yields

$$\nabla_x u(x) = (J_1^\top \otimes I_{n_l}) (J_2^\top \otimes I_{n_l}) \dots (J_{l-1}^\top \otimes I_{n_l}) \text{vec}(J_l)$$

Using the mixed-product property of Kronecker product (2.12) and setting $J_i^\top = A$,

$B = D = I_{n_l}$ and $J_{i+1}^\top = C$ yields

$$(J_i^\top \otimes I_{n_l}) (J_{i+1}^\top \otimes I_{n_l}) = (J_i^\top J_{i+1}^\top) \otimes I_{n_l}$$

Using this repeatedly and simplifying yields,

$$\nabla_x u(x) = \left((J_1^\top J_2^\top \dots J_{l-1}^\top) \otimes I_{n_l} \right) \text{vec}(J_l)$$

Using $J_i^\top = \left(\nabla_{f^{(i-1)}} f^{(i)} \right)^\top$ and $\text{vec}(J_l) = \nabla_{f^{(l-1)}} f^{(l)}$, we arrive at

$$\nabla_x u(x) = \left(\left[\left(\nabla_{f^{(1)}} f^{(2)} \right)^\top \dots \left(\nabla_{f^{(l-2)}} f^{(l-1)} \right)^\top \right] \otimes I_{n_l} \right) \left(\nabla_{f^{(l-1)}} f^{(l)} \right) \in \mathbb{R}^{n_l n_0 \times 1} \quad (2.57)$$

which represents the stacked gradient form of the chain rule

Second Derivatives of the Composite Mapping

For each layer k , denote the second derivative of the mapping $f^{(k)}$ in the stacked operator form

$$H^{(k)} := \nabla_{f^{(k-1)} f^{(k-1)}}^2 f^{(k)} \in \mathbb{R}^{n_k n_{k-1} \times n_{k-1}} \quad (2.58)$$

Define the prefix and suffix products

$$J_{1:k-1} := J_{k-1} \dots J_1, \quad J_{1:0} := I_{n_0} \quad (2.59)$$

$$\Pi_{k+1:l} := J_l \dots J_{k+1}, \quad \Pi_{l+1:l} := I_{n_l} \quad (2.60)$$

By applying gradient operator to $D_x u(x)$, using the formula (2.41) recursively and simplifying, we obtain the stacked second derivative of the composite function as

$$\nabla_{xx}^2 u(x) = \sum_{k=1}^l \left[(I_{n_0} \otimes \Pi_{k+1:l}) \nabla_{f^{(k-1)} f^{(k-1)}}^2 f^{(k)} J_{1:k-1} \right] \quad (2.61)$$

Each term in (2.61) represents the curvature contribution of one layer of the composite mapping. The suffix product propagates that curvature to the final output, while the prefix product pulls it back to the original variable x . This identity is used later when differentiating nested kinematic and geometric mappings.

2.4 Kronecker-Product Differential Identities

This section collects differential identities involving Kronecker products that arise frequently in the structured derivative calculations used later in the dissertation. Let

$$x \in \mathbb{R}^n, \quad f: \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad A: \mathbb{R}^n \rightarrow \mathbb{R}^{m \times pm}, \quad B: \mathbb{R}^n \rightarrow \mathbb{R}^{m \times r}$$

Derivative of $f(x) \otimes I_p$

We first compute $\nabla_x(f(x) \otimes I_p)$. Since I_p is constant, differentiation acts only on $f(x)$. For each component x_i ,

$$\frac{\partial}{\partial x_i}(f(x) \otimes I_p) = \frac{\partial f}{\partial x_i} \otimes I_p$$

Stacking these derivatives according to the definition of ∇_x gives

$$\nabla_x(f(x) \otimes I_p) = (\nabla_x f) \otimes I_p \quad (2.62)$$

Derivative of $I_p \otimes f(x)$

This derivative is slightly more involved because the Kronecker product places copies of $f(x)$ in a block-diagonal pattern. Since $I_p \otimes f(x)$ has a block diagonal structure,

$$I_p \otimes f(x) = \text{diag}(f(x), \dots, f(x))$$

Therefore,

$$\frac{\partial}{\partial x_i}(I_p \otimes f(x)) = I_p \otimes \frac{\partial f}{\partial x_i}$$

Stacking over $i = 1, \dots, n$ yields

$$\nabla_x(I_p \otimes f(x)) = \begin{bmatrix} I_p \otimes \frac{\partial f}{\partial x_1} \\ \vdots \\ I_p \otimes \frac{\partial f}{\partial x_n} \end{bmatrix}$$

This stacking order differs from the direct vectorization order induced by $D_x f$. The permutation matrix $P_{(p,n)}$ introduced in Section 2.1 aligns the p Kronecker blocks with the n derivative directions. Thus,

$$\nabla_x(I_p \otimes f) = (P_{(p,n)} \otimes I_m) (I_p \otimes \text{vec}(D_x f)) \quad (2.63)$$

Equation (2.63) expresses the same componentwise derivatives as the stacked form above, but in a compact form compatible with the adopted vectorization and block-stacking conventions.

Derivative of $H(x) = A(x) (I_p \otimes B(x))$

Let

$$H(x) = A(x) (I_p \otimes B(x))$$

Using the matrix product rule (2.41),

$$\nabla_x H = (\nabla_x A)(I_p \otimes B) + (I_n \otimes A) \nabla_x (I_p \otimes B)$$

Since I_p is a constant

$$\nabla_x (I_p \otimes B) = \begin{bmatrix} I_p \otimes \frac{\partial B}{\partial x_1} \\ \vdots \\ I_p \otimes \frac{\partial B}{\partial x_n} \end{bmatrix}$$

Therefore

$$(I_n \otimes A) \nabla_x (I_p \otimes B) = \begin{bmatrix} A \left(I_p \otimes \frac{\partial B}{\partial x_1} \right) \\ \vdots \\ A \left(I_p \otimes \frac{\partial B}{\partial x_n} \right) \end{bmatrix}$$

Partition A into p blocks of size $m \times m$

$$A = [A_1 \quad \cdots \quad A_p], \quad A_i \in \mathbb{R}^{m \times m}$$

Then for each k ,

$$A \left(I_p \otimes \frac{\partial B}{\partial x_k} \right) = \begin{bmatrix} A_1 \frac{\partial B}{\partial x_k} & \cdots & A_p \frac{\partial B}{\partial x_k} \end{bmatrix}$$

Substituting this expression into (2.65) yields,

$$(I_n \otimes A) \nabla_x (I_p \otimes B) = \begin{bmatrix} A_1 \frac{\partial B}{\partial x_1} & \cdots & A_p \frac{\partial B}{\partial x_1} \\ \vdots & \ddots & \vdots \\ A_1 \frac{\partial B}{\partial x_n} & \cdots & A_p \frac{\partial B}{\partial x_n} \end{bmatrix}$$

By definition of the block-wise product $\odot$, this matrix equals $A \odot \nabla_x B(x)$. Hence,

$$\nabla_x H(x) = \nabla_x A(x) (I_p \otimes B(x)) + A \odot \nabla_x B(x) \quad (2.64)$$

which proves the identity.

2.5 Auxiliary Linear Algebra Results

This section records auxiliary linear algebra results used later in the analysis of structured derivative mappings and sparse optimization systems. The results are standard, but they are included here to fix notation and to make later derivations self-contained.

Lemma 2.1: Local Solution of a Full-Row-Rank Parametric Linear System

Consider the parametric linear system

$$A(x)y = B(x), \quad A \in \mathbb{R}^{k \times n}, \quad B \in \mathbb{R}^{k \times 1}$$

and suppose that $A(x)$ and $B(x)$ are continuously differentiable in a neighborhood of x_0 . Suppose that

$$\text{rank}(A(x_0)) = k$$

Then there exists an open neighborhood X_0 of x_0 such that

$$\text{rank}(A(x)) = k, \quad \forall x \in X_0$$

In particular. For all $x \in X_0$ the Moore-Penrose pseudoinverse $A^+(x)$ is well-defined and admits the explicit representation

$$A^+(x) = A^\top(x)(A(x)A^\top(x))^{-1}$$

Moreover, the mapping

$$G(x) \triangleq A^+(x)B(x)$$

is continuously differentiable on X_0 and satisfies

$$A(x)G(x) = B(x), \quad \forall x \in X_0$$

Proof

Since $\text{rank}(A(x_0)) = k$, the matrix $A(x_0)A^\top(x_0) \in \mathbb{R}^{k \times k}$ is nonsingular. By continuity of $A(x)$, there exists an open neighborhood X_0 of x_0 such that $A(x)A^\top(x)$ remains nonsingular for all $x \in X_0$. Hence, $A(x)$ has full row rank on X_0 , and the full row rank pseudoinverse is

$$A^+(x) = A^\top(x)(A(x)A^\top(x))^{-1}$$

Therefore,

$$A(x)A^+(x) = A(x)A^\top(x)(A(x)A^\top(x))^{-1} = I_k$$

With $G(x) = A^+(x)B(x)$, it follows that

$$A(x)G(x) = A(x)A^+(x)B(x) = B(x)$$

Since $A(x)$ and $B(x)$ are continuously differentiable, and matrix inversion is continuously differentiable on nonsingular matrices, $A^+(x)$ and $G(x)$ are continuously differentiable on X_0 .

Lemma 2.2: Inverse of a Block Lower Triangular Matrix

Let $L = \begin{bmatrix} L_{11} & 0 & \cdots & 0 \\ L_{21} & L_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{p1} & L_{p2} & \cdots & L_{pp} \end{bmatrix}$ be a block lower triangular matrix. If each

diagonal block $L_{ii} \in \mathbb{R}^{n \times n}$ is invertible, then L is invertible and L^{-1} is also block lower triangular:

$$L^{-1} = \begin{bmatrix} \bar{L}_{11} & 0 & \cdots & 0 \\ \bar{L}_{21} & \bar{L}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{L}_{p1} & \bar{L}_{p2} & \cdots & \bar{L}_{pp} \end{bmatrix}$$

where

$$\bar{L}_{ij} = \begin{cases} L_{ii}^{-1} & , \quad i = j \\ -L_{ii}^{-1} \sum_{l=j}^{i-1} L_{il} \bar{L}_{lj} & , \quad i > j \\ 0 & , \quad i < j \end{cases} \quad (2.65)$$

Proof

Because L is block lower triangular,

$$\det(L) = \prod_{i=1}^p \det(L_{ii})$$

Since each $\det(L_{ii}) \neq 0$, it follows that $\det(L) \neq 0$ and therefore L is invertible. Let

$$L^{-1} = [\bar{L}_{ij}]_{i,j=1}^p$$

Imposing the identity $LL^{-1} = I$ and comparing block entries yields

- For $i = j$, $L_{ii} \bar{L}_{ii} = I \Rightarrow \bar{L}_{ii} = L_{ii}^{-1}$
- For $i > j$, $\sum_{l=j}^i L_{il} \bar{L}_{lj} = 0$ Solving recursively for $\bar{L}_{ij}$ yields

$$\bar{L}_{ij} = -L_{ii}^{-1} \sum_{l=j}^{i-1} L_{il} \bar{L}_{lj}$$

- For $i < j$, $\bar{L}_{ij} = 0$

Thus L^{-1} is block lower triangular with the stated block recursion.

Lemma 2.3: $LDL^\top$ Factorization, Inverse, and Inertia

Let $A \in \mathbb{R}^{n \times n}$ be symmetric, and suppose that it admits the factorization

$$A = LDL^\top$$

where

$$D = \text{diag}(d_1, \dots, d_n), \quad L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}$$

If $d_i \neq 0$ for all i , then A is nonsingular and its inverse is given by

$$A^{-1} = L^{-\top} D^{-1} L^{-1}$$

Moreover, since A is congruent to D , Sylvester's law of inertia gives

$$\text{In}(A) = (n_+, n_-, n_0)$$

Therefore,

$$n_+ = \#\{d_i > 0\}, \quad n_- = \#\{d_i < 0\}, \quad n_0 = \#\{d_i = 0\}$$

Proof

If $d_i \neq 0$ for all i , then D is invertible. Since L is a lower triangular unit, it is also invertible. Hence

$$A^{-1} = L^{-\top} D^{-1} L^{-1}$$

The inertia statement follows because $A = LDL^\top$ is a congruence transformation of D . By Sylvester's law of inertia, congruent symmetric matrices have the same inertia. Since the eigenvalues of D are its diagonal entries, the inertia of A is determined by the signs of $d_1, \dots, d_n$.

In numerical implementations, symmetric pivoting is generally used to

improve stability and avoid zero or small pivots. Therefore, the unpivoted form above should be interpreted as the algebraic form used to explain the relationship between LDL^T factors and inertia; practical sparse solvers typically use pivoted variants.

Algorithm 2.1: LDL^T Factorization with Inertia Detection

Inputs: Symmetric matrix $A \in \mathbb{R}^{n \times n}$.

Outputs:

- $L \in \mathbb{R}^{n \times n}$: unit lower triangular factor.
- $D \in \mathbb{R}^{n \times n}$: diagonal factor.
- flag: validity indicator, where flag = 1 if the factorization completes without zero pivots and flag = 0 otherwise.

Procedure:

1. Initialize $L = I_n, D = 0$.

2. For $k = 1, \dots, n$:

a. Compute $d_k = A_{kk} - \sum_{j=1}^{k-1} l_{kj}^2 d_j$.

b. Set $D_{kk} = d_k$.

c. If $d_k = 0$, set flag = 0 and terminate.

d. For $i = k + 1, \dots, n$, compute $l_{ik} = \frac{1}{d_k} \left(A_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj} d_j \right)$.

3. If all pivots are nonzero, set flag = 1.

4. Return L, D , and flag.

The signs of the diagonal entries of D determine the inertia of A . If all $d_i > 0$,

then A is positive definite. If both positive and negative pivots occur, then A is indefinite. The LDL^T factorization is therefore useful for symmetric indefinite systems, including the KKT systems that arise in interior-point and sequential quadratic programming methods. In such systems, the factorization provides both a mechanism for solving the linear equations and a way to monitor inertia-related curvature information.

2.6 Chapter Summary

This chapter introduced the notation and identities used throughout the dissertation, including Schur complements, matrix inertia, Kronecker products, vectorization, tensor unfolding, and structured derivatives. These tools support the analytical derivative developments and structure-exploiting optimization methods presented in later chapters. The structured derivative notation preserves matrix and block organization instead of immediately vectorizing intermediate quantities, making later derivative expressions more compact and exposing sparsity in dynamics, collision-distance derivatives, and KKT derivative blocks. Together, these definitions and auxiliary results provide the mathematical framework used later to derive differentiable collision constraints, analytical derivative formulas, and sparse primal-dual optimization algorithms.

CHAPTER 3 - GENERALIZED ROBOTIC MODELLING AND MODEL PREDICTIVE PATH FOLLOWING CONTROL FORMULATION

3.1 Introduction and Scope

Robotic systems encountered in modern motion planning, control, and optimization problems exhibit significant diversity in mechanical structure, actuation architecture, and task specification. Nevertheless, a broad class of robotic platforms, including fixed-base manipulators, floating-base robots, mobile robots, articulated vehicles, and other rigid-body systems, share a common structural property: their evolution can be described as a finite-dimensional dynamical system governed by ordinary differential equations, whose states encode the configuration and motion of rigid bodies [72]- [76].

This chapter defines the notation and system representation used throughout the dissertation. It states the finite-dimensional rigid-body assumptions used for the collision modeling, analytical derivatives, and MPPFC formulation developed in later chapters. The notation includes generalized configuration coordinates, first-order state-space dynamics, actuator and internal states, task-output maps, and path-following variables.

Robotic systems whose configuration cannot be represented by a finite-dimensional rigid-body configuration manifold, such as soft robots, continuum manipulators, or systems governed by distributed-parameter models, are explicitly excluded from the present scope and deferred to future work. These systems require different modeling assumptions and alter the geometric structure

underlying collision modeling and distance-function differentiation. The assumptions and notation defined in this chapter provide the basis for the differentiable collision constraints, analytical derivative expressions, and structure-exploiting NMPC solver developed in the following chapters.

Throughout Chapters 3–8, calligraphic letters denote sets and manifolds, uppercase Roman letters denote matrices, and lowercase letters denote vectors or scalars. Symbols used across multiple chapters retain a single principal meaning. Fixed model and tuning parameters are defined once and are omitted from function argument lists unless they are treated as independent variables.

3.2 Configuration Spaces for Robotic Systems

3.2.1 Configuration Space Dimension

Let the configuration of a robotic system be denoted by

$$\rho \in \mathcal{Q} \tag{3.1}$$

where $\mathcal{Q}$ is a smooth configuration manifold. Its dimension is defined as

$$n_\rho := \dim(\mathcal{Q}) \tag{3.2}$$

and represents the number of independent configuration degrees of freedom of the system. The dimension n_ρ is distinct from the dimension of the ambient (workspace) space, which is typically $\mathbb{R}^3$ for spatial robotic systems. While rigid bodies are embedded in the ambient space, their configuration evolves on the manifold $\mathcal{Q}$. The configuration-space formulation adopted here follows the standard geometric modeling framework used in robotics and multibody dynamics literature [72]–[76]. The configuration manifold $\mathcal{Q}$ is termed nonlinear when it cannot be

globally identified with a Euclidean vector space $\mathbb{R}^{n_\rho}$ through a single coordinate chart. Such nonlinearity arises naturally in robotic systems involving rotations, rigid-body motion, or kinematic constraints.

3.2.2 Lie Groups for Rigid-Body Configuration

To describe rigid-body configurations in a form compatible with the state-space and collision models used later, this section recalls the basic configuration manifolds $SO(3)$, $SE(3)$ and their Cartesian products with Euclidean joint-coordinate spaces [72], [73], [76].

The Special Orthogonal Group is defined as

$$SO(3) = \{R \in \mathbb{R}^{3 \times 3} \mid R^T R = I, \det(R) = 1\} \quad (3.3)$$

The group $SO(3)$ represents proper rotations in three-dimensional space. Elements of $SO(3)$ act on vectors in $\mathbb{R}^3$ through the mapping

$$y = Rx$$

In the state representation, orientations may be parameterized using Euler angles, angle-axis variables, unit quaternions, or other suitable coordinates.

The Special Euclidean Group is defined as

$$SE(3) = \{(R, p) \mid R \in SO(3), p \in \mathbb{R}^3\} \quad (3.4)$$

The group $SE(3)$ represents rigid-body motions combining rotation and translation. Each element maps local-frame points to global-frame points by

$$y = p + Rx \quad (3.5)$$

The manifold $SE(3)$ has dimension six and is commonly used to describe the pose of a free rigid body or floating base [72], [73], [75], [76].

For systems whose configuration contains both a rigid-body pose and additional internal generalized coordinates, the configuration manifold can be represented as a Cartesian product of the pose space and a Euclidean internal-coordinate space:

$$\mathcal{Q} = SE(3) \times \mathbb{R}^{n_\rho} \quad (3.6)$$

where n_j denotes the number of internal generalized coordinates, such as revolute or prismatic joint variables. The total configuration dimension is therefore

$$n_\rho = \dim(\mathcal{Q}) = 6 + n_j \quad (3.7)$$

Here, the $SE(3)$ component describes the rigid-body pose, while the $\mathbb{R}^{n_j}$ component describes the internal generalized coordinates. This Cartesian product structure is common in floating-base articulated systems, humanoid robots, legged robots, mobile manipulators, and free-flying space robots [72], [73], [75], [76].

3.2.3 Generalized Configuration Coordinates

Throughout this dissertation, the system configuration is described using generalized configuration coordinates or simply generalized coordinates.

Definition 3.1 (Generalized Configuration Coordinates):

The vector $\rho \in \mathcal{Q}$ denotes a minimal set of independent generalized coordinates, comprising position and orientation variables sufficient to uniquely determine the full geometric configuration (pose) of every rigid body or link in the system. The generalized coordinates are decomposed as

$$\rho \triangleq (q, \varphi) \quad (3.8)$$

where $q \in \mathbb{R}^{n_q}$ represents generalized translational coordinates, including global

base position and prismatic joint displacements, and $\varphi \in \mathbb{R}^{n_\varphi}$ represents generalized rotational coordinates, including revolute joint angles or floating-base orientation parameters. The total number of configuration degrees of freedom satisfies

$$n_\rho = n_q + n_\varphi \quad (3.9)$$

The coordinates ρ are minimal, meaning that redundant pose representations such as homogeneous transformation matrices are not included as state variables. Instead, such quantities are treated as derived variables computed through smooth kinematic mappings from the generalized coordinates [73]- [76].

3.3 Rigid-Body Transformations and Forward Kinematics

This section introduces a geometric model of rigid bodies and robot links that is compatible with smooth differentiation. The key objective is to define a mapping from the generalized configuration coordinates to the world-frame occupied set of each rigid body, and to do so in a form that supports analytic gradients and Hessians of collision-related quantities developed in later chapters.

3.3.1 Parametric Convex-Set Model of Rigid Bodies

Let $\mathcal{C} \subset \mathbb{R}^3$ denote a rigid body modeled as a closed, bounded, non-empty convex set embedded in the ambient space $\mathbb{R}^3$. The body geometry is represented in local reference frame by a smooth parametric mapping. Specifically, define

$$\mathcal{C} = \{y \in \mathbb{R}^3 \mid y = S(u, v, w), \quad (u, v, w) \in [0,1]^3\} \quad (3.10)$$

where $S: [0,1]^3 \rightarrow \mathbb{R}^3$ is assumed to be at least twice continuously differentiable.

This regularity ensures that surface normals, gradients, and second-order

derivatives are well defined, which is required for the analytic distance derivatives used in collision-aware optimization. A boundary parameterization is obtained by restricting the parameter domain to a two-dimensional chart. Without loss of generality, fix $w = w_b$ and define $B(s) = S(u, v, w_b)$ with $s = (u, v) \in [0,1]^2$. The boundary surface is represented by

$$C_b = \{y \in \mathbb{R}^3 \mid y = B(s), \quad s \in [0,1]^2\} \quad (3.11)$$

where $B: [0,1]^2 \rightarrow \mathbb{R}^3$ parameterizes the boundary of the rigid body ∂C .

This parametric representation is intentionally adopted to avoid discretization-based collision modeling. Surface discretization and sampling-based approximations may introduce resolution-dependent artifacts and non-smooth gradients, which are undesirable in optimization-based planners requiring differentiable collision constraints [42].

3.3.2 World-Frame Rigid-Body Transformation

To place the local rigid-body model into the world frame, a rigid transformation, consisting of translation and rotation is applied. Using the generalized configuration variables (q, φ) introduced earlier, define the world frame occupied set

$$C(q, \varphi) = \{y \in \mathbb{R}^3 \mid y = T(q, \varphi) + R(\varphi)S(u, v, w), 0 \leq u, v, w \leq 1\} \quad (3.12)$$

and the world frame boundary set

$$C_b(q, \varphi) = \{y \in \mathbb{R}^3 \mid y = T(q, \varphi) + R(\varphi)B(s), 0 \leq s \leq 1\} \quad (3.13)$$

where $R(\varphi) \in SO(3)$ is the rotation matrix mapping the body frame to the world frame, and $T(q, \varphi) \in \mathbb{R}^3$ is the translation of the body-frame origin expressed in

world coordinates. The mappings $R(\varphi)$ and $T(q, \varphi)$ are assumed smooth, enabling analytical gradients and Hessians of geometric functions in the later chapters. For a single rigid body object, the $T(q, \varphi)$ reduces to the translational coordinates q . For articulated objects, such as a robot manipulator, the transformation is computed via forward kinematics, with $T(q, \varphi)$ and $R(\varphi)$ giving the position and orientation of a specific link's reference frame as a function of the full configuration.

The rotation matrix $R(\varphi)$ depends on the choice of rotation parameterization used to represent the orientation variables φ . Common parameterizations include Euler angles, unit quaternions, and axis-angle (exponential coordinate) representations. Each parameterization provides a smooth mapping from a finite-dimensional coordinate vector φ to a rotation matrix $R(\varphi) \in SO(3)$, but differs in terms of singularities, redundancy, and numerical properties.

In this dissertation, no specific rotation parameterization is assumed. The mapping $R(\varphi)$ is treated abstractly as a smooth function of the rotational coordinates. Consequently, analytical derivatives of $R(\varphi)$ with respect to φ are not derived here. If required for implementation in a particular coordinate representation, the corresponding gradient and Hessian expressions can be obtained using standard identities for the chosen parameterization.

3.3.3 Forward Kinematics for Multi-Link Systems

To describe the pose of any link in a robotic manipulator with multiple degrees of freedom, we define a hierarchy of reference frames:

- World (global) frame F_w : A fixed inertial frame of reference.

- Body (local) frame F_b : Rigidly attached to a mobile robot's chassis. It moves with the robot's chassis and is used to describe the robot's pose relative to the world frame.
- Manipulator link frames F_i : Rigidly attached to each individual joint or links. These frames describe the relative motion between consecutive links in the kinematic chain.

Consider a manipulator with n links indexed by $\{0, 1, \dots, n - 1\}$, where link 0 is the base. Let joint k connect links $k - 1$ and k . Let $H_i^w \in SE(3)$ denote the homogenous transformation from F_i to F_w . The base-to-world transformation is

$$H_0^w = \begin{bmatrix} R(\varphi_0) & p_0(q_0) \\ 0_{1 \times 3} & 1 \end{bmatrix} \quad (3.14)$$

and the relative transform from frame k to frame $k - 1$ is

$$H_k^{k-1} = \begin{bmatrix} R(\varphi_k) & p_k(q_k) \\ 0_{1 \times 3} & 1 \end{bmatrix} \quad (3.15)$$

where

- $R(\varphi_k) \in SO(3)$ is the rotation matrix describing the orientation of frame F_k relative to frame F_{k-1} ,
 - $p_k(q_k) \in \mathbb{R}^3$ denotes the local translation vector from the origin of frame F_{k-1} to the origin of frame F_k , expressed in frame F_{k-1} ,
 - $\varphi_k \in \mathbb{R}^{n_{\varphi_k}}$ is the rotational parameter vector associated with joint k ;
 - $q_k \in \mathbb{R}^{n_{q_k}}$ is the translational parameter vector associated with joint k ;
- maybe scalar or vector-valued depending on the joint type.

For a serial chain, the world pose of link i is given recursively by

$$H_i^w = H_0^w \prod_{k=1}^i H_k^{k-1} \quad (3.16)$$

Expanding (3.16) yields

$$H_i^w = \begin{bmatrix} \prod_{k=0}^i R(\varphi_k) & \sum_{l=0}^i \left(\prod_{k=0}^{l-1} R(\varphi_k) \right) p_l(q_l) \\ 0_{1 \times 3} & 1 \end{bmatrix} \quad (3.17)$$

Global Coordinates of a Point on a Link

Let $y_{l_i} \in \mathbb{R}^3$ denote a point expressed in the local link frame F_{l_i} . Its world coordinates, $y_i \in \mathbb{R}^3$ are obtained via:

$$\begin{bmatrix} y_i \\ 1 \end{bmatrix} = H_i^w \begin{bmatrix} y_{l_i} \\ 1 \end{bmatrix} \quad (3.18)$$

or equivalently:

$$y_i = \sum_{l=0}^i \left(\prod_{k=0}^{l-1} R(\varphi_k) \right) p_l(q_l) + \left(\prod_{k=0}^i R(\varphi_k) \right) y_{l_i} \quad (3.19)$$

These expressions are used later when mapping link-surface parameterizations into world coordinates for collision distance evaluation.

3.3.4 Denavit–Hartenberg (DH) Parameterization

A standard convention for assigning coordinate frames along a serial robot chain is the Denavit–Hartenberg parameterization [74], [76]. Each link is described by four parameters: joint angle φ , link offset q , link length l , and link twist angle α .

The DH transformation matrix from link $k - 1$ to link k is defined as:

$$H_k^{k-1}(q_k, \varphi_k) = \begin{bmatrix} \cos \varphi_k & -\sin \varphi_k \cos \alpha_k & \sin \varphi_k \sin \alpha_k & l_k \cos \varphi_k \\ \sin \varphi_k & \cos \varphi_k \cos \alpha_k & -\cos \varphi_k \sin \alpha_k & l_k \sin \varphi_k \\ 0 & \sin \alpha_k & \cos \alpha_k & q_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.20)$$

Here, φ_k is the angle between x_{k-1} and x_k , measured as a rotation about the z_{k-1} axis, and is a controlled variable for revolute and hybrid joints. The link offset q_k is the distance between x_{k-1} and x_k , measured along the z_{k-1} axis, and is a controlled variable for prismatic and hybrid joints. The link twist α_k is the angle between z_{k-1} and z_k , measured as a rotation about the x_k axis. The link length l_k is the distance between z_{k-1} and z_k , measured along the x_k axis, and corresponds to the common-normal distance between consecutive joint axes. In this dissertation, the DH convention is used to construct the forward kinematics of articulated robotic systems. The generalized configuration variables are constructed by stacking only the actuated DH variables, together with any mobile-base coordinates.

Let n_{prism} denote the number of prismatic joints and n_{rev} the number of revolute joints. The translational and rotational configuration vectors are defined as

$$\begin{aligned} q &= \text{vec}(q_0, q_1, \dots, q_{n_{prism}}) \in \mathbb{R}^{n_q} \\ \varphi &= \text{vec}(\varphi_0, \varphi_1, \dots, \varphi_{n_{rev}}) \in \mathbb{R}^{n_\varphi} \end{aligned} \quad (3.21)$$

where q_0 denotes the translational coordinates of the robot base when the system is mobile, and q_k denotes the displacement of the k -th prismatic joint. Similarly, φ_0 denotes the orientation parameters of the robot base when the robot has a floating or mobile base, and φ_k denotes the joint angle associated with the k -th revolute joint.

3.3.5 World-Frame Mapping of Convex Link Geometry

Assume each link i is modeled as a convex rigid body with a smooth boundary parameterization $B_i(s_i)$, $s_i \in [0,1]^2$, defined in an intrinsic geometry frame. Allowing an embedding offset between the intrinsic geometry frame and the link frame F_i , define the link-frame boundary point

$$y_{l_i}(s_i) = y_{ls_i} + R(\varphi_{s_i})B_i(s_i) \quad (3.22)$$

where $y_{l_i} \in \mathbb{R}^3$, $R(\varphi_{s_i}) \in SO(3)$ are constant offsets describing how the geometry model is placed within F_i .

Composing (3.19) with (3.22) yields the world-frame coordinates of a boundary point on link i . We write the result in the compact form

$$y_i(q, \varphi, s_i) = T_i(q, \varphi) + R_i(\varphi)B_i(s_i) \quad (3.23)$$

where

$$T_i(q, \varphi) = \sum_{j=0}^i \left(\prod_{k=0}^{j-1} R(\varphi_k) \right) p_j(q_j) + \left(\prod_{k=0}^i R(\varphi_k) \right) y_{ls_i} \quad (3.24)$$

$$R_i(\varphi) = \left(\prod_{k=0}^i R(\varphi_k) \right) R(\varphi_{s_i}) \quad (3.25)$$

$R(.)$: elementary rotaion, $R_i(\varphi)$: cumulative rotaion

3.4 State-Space Representation of Robotic Dynamics

The instantaneous motion of the system at configuration $\rho \in \mathcal{Q}$ is described by a generalized velocity $\xi \in T_\rho \mathcal{Q}$, where $T_\rho \mathcal{Q}$ denotes the tangent space of the configuration manifold at ρ . The tangent space $T_\rho \mathcal{Q}$ represents the set of all admissible instantaneous motions consistent with the geometry of the configuration

manifold. Throughout this dissertation, generalized velocities are interpreted as elements of the tangent space $T_\rho Q$. Quantities such as generalized forces, momenta, and gradients of scalar functions belong to the cotangent space $T_\rho^* Q$ and are treated separately in optimization and optimality conditions [72], [76]. With the above definitions, the system dynamics are written in first-order form as

$$\begin{aligned}\dot{\rho} &= f_\rho(\rho, \xi) \\ \dot{\xi} &= f_\xi(\rho, \xi, u)\end{aligned}\tag{3.26}$$

where $u \in \mathbb{R}^{n_u}$ denotes the control input, f_ρ encodes the kinematic relationship between configuration and generalized velocity, and f_ξ encodes the system dynamics.

The mapping f_ρ accounts for the geometry of the configuration manifold. In particular, for Euclidean configuration spaces $f_\rho(\rho, \xi) = \xi$, whereas for nonlinear manifolds (e.g., $SO(3)$, $SE(3)$), f_ρ represents the appropriate reconstruction equation relating tangent-space velocities to time evolution of the configuration coordinates [72], [73], [76].

This representation provides a geometrically consistent state-space model valid for both Euclidean and nonlinear configuration manifolds [72], [73], [76] and forms the basis for the optimization-based control formulations developed in subsequent chapters [53].

Example: Fixed-Base Manipulators (Euler–Lagrange Form)

A classical rigid fixed-base manipulator is commonly modeled by the Euler–Lagrange equations

$$M(\rho)\ddot{\rho} + H(\rho, \dot{\rho})\dot{\rho} + K(\rho) = G(\rho, \xi)u$$

where $\rho \in \mathbb{R}^{n_\rho}$, $M(\rho)$ is the configuration dependent positive definite mass matrix, $H(\rho, \dot{\rho})\dot{\rho}$ captures Coriolis and centrifugal effects, $K(\rho)$ represents conservative generalized forces such as gravity, and u denotes the generalized forces applied to the system and $G(\rho, \xi)$ maps the control inputs to the generalized forces.

Such formulations arise naturally from the Euler–Lagrange equations of rigid-body systems and are standard in robotic dynamics [74]- [76]. By defining the generalized velocity as $\xi := \dot{\rho}$, the first-order form becomes

$$\begin{aligned} \dot{\rho} &= \xi \\ \dot{\xi} &= M^{-1}(\rho)(-H(\rho, \xi)\xi - K(\rho) + G(\rho, \xi)u) \end{aligned} \quad (3.26)$$

which is a direct specialization of (3.25) with $f_\rho(\rho, \xi) = \xi$.

Floating-Base Robots

Floating-base robotic systems arise in applications where the robot's base is not fixed to the environment, such as humanoid robots, legged robots during flight phases, mobile manipulators with unactuated bases, and free-flying space robots. In such systems, the configuration includes both a free rigid-body base and internal joint coordinates [75], [76]. The generalized configuration coordinates take the form

$$\rho = (p, R, q_j) \in SE(3) \times \mathbb{R}^n \quad (3.27)$$

where:

- $p \in \mathbb{R}^3$ denotes the position of the base reference frame in the ambient space,
- $R \in SO(3)$ denotes the orientation of the base reference frame,
- $q_j \in \mathbb{R}^n$ denotes the internal joint coordinates.

The corresponding generalized velocity

$$\xi \in T_{\rho}Q$$

contains both:

- the rigid-body twist of the floating base (linear and angular velocity), and
- the joint-rate vector associated with q_j .

When the base orientation is represented by a rotation matrix $R \in SO(3)$, its time derivative follows a reconstruction equation defined on Lie group $SO(3)$. Let

$$\omega \in \mathbb{R}^n$$

denote the body-frame angular velocity. Define the hat operator

$$\widehat{(\cdot)}: \mathbb{R}^3 \rightarrow SO(3)$$

which maps $\omega = [\omega_1, \omega_2, \omega_3]^\top$ to the skew-symmetric matrix

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \quad \widehat{\omega}^\top = -\widehat{\omega} \quad (3.28)$$

This mapping satisfies

$$\widehat{\omega}x = \omega \times x, \quad \forall x \in \mathbb{R}^3$$

and provides an isomorphism between $\mathbb{R}^3$ and the Lie algebra $SO(3)$ [72], [73], [76].

With this notation, the rotational kinematics of the floating base are given by

$$\dot{R} = R\widehat{\omega} \quad (3.29)$$

which ensures that $R(t) \in SO(3)$ for all time. If the angular velocity is instead expressed in the spatial (world) frame, the equivalent form is $\dot{R} = R\widehat{\omega}$.

Unlike fixed-base manipulators, floating-base systems require reconstruction equations because their orientation evolves on $SO(3)$. These equations keep the

configuration trajectory on the manifold and allow floating-base systems to be represented within the same first-order optimization and NMPC structure used for fixed-base systems [72], [73], [76].

Example: Nonholonomic Systems

Many mobile robots and articulated vehicles are subject to nonholonomic constraints that restrict admissible velocities. Such systems are often expressed as

$$\dot{\rho} = \Gamma(\rho)\xi$$

where ξ is a reduced velocity coordinate and $\Gamma(\rho)$ maps into $T_\rho Q$. Dynamic extensions can be written as $\dot{\xi} = f_\xi(\rho, \xi, u)$, which again conforms to (3.26) with $f_\rho(\rho, \xi) = \Gamma(\rho)\xi$. This embedding allows nonholonomic kinematics to be handled within the same NMPC interface.

3.5 Extended State Models for Robotic Systems

Many robotic and mechanical systems exhibit dynamics that extend beyond the second-order rigid-body equations implied by configuration and generalized velocity alone. Such effects include actuator dynamics, drivetrain dynamics, compliance states, contact-related internal variables, payload dynamics, and estimator or observer states [76]. Provided that these effects remain finite-dimensional and do not alter the underlying rigid-body kinematic structure, they can be incorporated through state augmentation. To this end, let $\zeta \in \mathbb{R}^{n_\zeta}$ denote a vector of internal or auxiliary states, capturing dynamics that influence the system evolution but are not part of the generalized configuration or generalized velocity. The augmented system dynamics are written as

$$\begin{aligned}
\dot{\rho} &= f_{\rho}(\rho, \xi) \\
\dot{\xi} &= f_{\xi}(\rho, \xi, \zeta, u) \\
\dot{\zeta} &= f_{\zeta}(\rho, \xi, \zeta, u)
\end{aligned} \tag{3.30}$$

where f_{ζ} governs the evolution of internal states.

The internal state vector ζ may represent actuator states, drivetrain states, compliance or contact-related variables, payload dynamics, or estimator and observer states. Crucially, the introduction of ζ does not alter the configuration manifold $\mathcal{Q}$ or the rigid-body transformations defined in section 3.2 and 3.3. All geometric quantities, including link poses and collision geometry, remain functions of the generalized configuration coordinates ρ .

3.6 Task-Space Outputs and End-Effector Maps

The outputs of the system are defined as smooth functions of the system state and input. In this dissertation, attention is restricted to rigid-body task definitions, for which task-space quantities such as end-effector position or orientation are uniquely determined by the generalized configuration coordinates.

$$y = \begin{bmatrix} h_e(\rho) \\ h_d(\rho, \xi, \zeta) \end{bmatrix} \in \mathbb{R}^{n_y} \tag{3.31}$$

where:

- $h_e(\rho)$ represents task-space quantities such as end-effector positions, orientations, or other rigidly attached task features,
- $h_d(\rho, \xi, \zeta)$ represents additional dynamic or performance-related outputs, such as velocities, accelerations, energy-related metrics, actuator-related variables, or constraint-related quantities.

Throughout the remainder of this dissertation, it is assumed that the end-effector map h_e depends only on the generalized configuration coordinates ρ . This assumption is consistent with rigid-body kinematics and is sufficient for the collision modeling, distance-function derivatives, and optimization-based control formulations developed in subsequent chapters.

3.6.1 Multiple End-Effectors and Stacked Task Maps

Many robotic systems naturally involve multiple end-effectors or multiple task points. Examples include dual-arm manipulators, humanoids with two feet and two hands, mobile manipulators with a base pose and an arm tool point, and articulated vehicles where multiple points on the body are treated as task-relevant outputs. To model this within the configuration-only task assumption, define m end-effector maps

$$h_e^{(i)}(\rho): \mathcal{Q} \rightarrow \mathbb{R}^{n_{e,i}}, \quad i = 1, \dots, m$$

where $h_e^{(i)}(\rho)$ represents the i -th end-effector/task feature (e.g., a position vector in $\mathbb{R}^3$, an orientation representation, or a stacked pose vector). The combined end-effector output is formed by stacking

$$h_e(\rho) = \begin{bmatrix} h_e^{(1)}(\rho) \\ \vdots \\ h_e^{(m)}(\rho) \end{bmatrix} \in \mathbb{R}^{n_e}, \quad n_e := \sum_{i=1}^m n_{e,i} \quad (3.32)$$

This stacked representation allows multiple task objectives and constraints to be handled through a single output vector. It also provides a uniform interface for derivative computations while supporting heterogeneous task components such as positions, orientations, poses, or selected feature points.

3.6.2 Limitations of Configuration-Only End-Effector Maps

Assumption

In classical rigid-body kinematics, the end-effector pose is assumed to be uniquely determined by the configuration $y_e = h_e(\rho)$. This assumption holds when the end-effector or task point is rigidly attached to the kinematic chain and when no additional internal or dynamical states influence the pose of that task point [72], [74], [76]. However, this assumption does not hold universally. In systems with compliance, cable stretch, suspended payloads, contact-dependent task points, deformable tools, or perception-based task variables, the effective task output may also depend on the internal or auxiliary state vector ζ .

In such cases, the end-effector map can be generalized as $y_e = h_e(\rho, \zeta)$. If the measured or effective task output also depends on generalized velocity, a still more general representation is $y_e = h_e(\rho, \xi, \zeta)$. The present dissertation restricts attention to rigid-body task definitions for which each end-effector or task map is a smooth function of configuration alone. This scope is sufficient for the collision modeling, distance-function derivatives, and MPPFC formulation developed in the subsequent chapters, while systems requiring $h_e(\rho, \zeta)$ or $h_e(\rho, \xi, \zeta)$ are left as natural extensions of the framework.

3.7 Generalization to Nonlinear Dynamic Systems

Although the modeling framework has been introduced in the context of robotic systems, it is not inherently restricted to robotics. Any nonlinear dynamic system of the form

$$\dot{x} = f(x, u) \quad (3.33)$$

can be embedded within the same structure through an appropriate partition of the state vector. In particular, by defining

$$x := [\rho, \xi, \zeta]^\top$$

the system dynamics in (3.33) can be written in the above form. In this interpretation, the variables ρ , ξ , and ζ need not correspond strictly to mechanical coordinates when the plant is non-robotic; rather, they represent a structured partitioning of the state that preserves compatibility with the NMPC solver architecture. Outputs are defined generically as

$$y = h(x) = \begin{bmatrix} h_e(x) \\ h_d(x) \end{bmatrix} \quad (3.34)$$

where the output components represent performance and constraint-related quantities.

This formulation enables a single NMPC structure to be used for trajectory tracking, stabilization, and regulation across heterogeneous nonlinear systems, including both robotic and non-robotic plants. The key requirement is that the dynamics, outputs, and constraints remain sufficiently smooth for the derivative-based optimization methods used later in the dissertation. By separating rigid-body geometry from the general dynamic-state representation, this notation also provides a consistent basis for the differentiable collision constraints and distance-function derivatives developed in the following chapters.

3.8 Model Predictive Path Following Control Formulation

Building on the system representation introduced in the previous sections,

we now formulate a unified NMPC problem that couples path-following with collision avoidance through differentiable clearance constraints. The formulation proceeds in several stages. First, the system model used within the predictive controller is introduced. Next, an augmented MPPFC structure is defined to decouple path geometry from timing. A geometric framework for collision avoidance is then incorporated using differentiable distance functions. Finally, the complete NMPC problem is stated together with the assumptions required for efficient numerical solution.

3.8.1 System Model and Setup

The NMPC formulation developed in this section builds directly on the general nonlinear system representation introduced in Section 3.7. Recall that the plant dynamics can be written in continuous-time form as

$$\dot{x} = f(x, u)$$

$$y = h(x)$$

For predictive control, the continuous-time dynamics are discretized with sampling interval τ , yielding the discrete-time model

$$x_{k+1} = f(x_k, u_k), \quad x_k \in \mathbb{R}^{n_x}, \quad u_k \in \mathbb{R}^{n_u} \quad (3.35)$$

$$y_k = h(x_k), \quad y_k \in \mathbb{R}^{n_y} \quad (3.36)$$

The system may also be subject to smooth inequality constraints of the form

$$g(x_k, u_k) \leq 0$$

representing actuator limits, safety constraints, and other operational restrictions.

At each sampling instant t , we solve a finite-horizon NMPC problem to compute an

optimal control sequence over a prediction horizon of length p [53], [55]:

$$\begin{aligned}
\{x_k^*, u_k^*\}_{k=t}^{k=t+n_p} = & \underset{\{x_k, u_k\}_{k=t}^{k=t+n_p}}{\operatorname{argmin}} \sum_{k=t}^{t+n_p} J(x_k, u_k) + J_f(x_{t+n_p}) \\
\forall k \in [t, t+n_p] \quad & x_k = x_{k-1} + \frac{\tau}{2} \left[f(x_k, u_k) + f(x_{k-1}, u_{k-1}) \right] \\
& y_k = h(x_k) \\
& g(x_k, u_k) \leq 0 \\
& x_{t-1} = x_0, \quad u_{t-1} = u_0
\end{aligned} \tag{3.37}$$

Here, $J(x_k, u_k)$ denotes the stage cost, which penalizes state, output, input, or other stage-wise performance errors, while $J_f(x_{t+n_p})$ denotes the terminal cost applied at the end of the prediction horizon. The scalar τ represents the step size. Only the first control input of the optimized sequence $\{u_k^*\}_{k=t}^{k=t+n_p}$ is applied to the system, and at the next control update, the optimization is re-solved using updated state measurements. Trapezoidal integration is used for second-order accuracy and favorable numerical stability, while preserving sparsity for real-time solvers [53]. We assume that all maps are C^2 . This ensures well-defined gradients and Hessians required by second order optimization and standard NMPC regularity results, e.g., [53], [55].

3.8.2 Reference Path and Augmented Path Dynamics

We define the geometric reference as a C^2 , i.e., twice continuously differentiable, curve:

$$r_p(\theta), \quad \theta \in [\theta_l, \theta_h] \subset \mathbb{R} \tag{3.38}$$

where θ denotes the path progression variable and r_p represents a smooth mapping that describes the desired path in output space (e.g., position, orientation, etc.). If r_p

is not sufficiently smooth, a continuously differentiable approximation may be employed. To decouple the path geometry from timing, $\theta(t)$ follows an integrator chain

$$\frac{d^{(r)}\theta}{dt^{(r)}} = \alpha \quad (3.39)$$

where $\alpha \in \mathbb{R}$ is a virtual control input and a decision variable within the optimization problem. To allow the optimizer to attach to the most appropriate segment of the path, θ_t is treated as a free variable at time t and forward progress is enforced along the horizon:

$$\dot{\theta}_k \geq 0 \quad \forall k \geq t + 1 \quad (3.40)$$

ensuring that the path parameter evolves monotonically forward over the prediction horizon. A small regularization term such as

$$w_\theta \|\theta_t - \theta_{t-1}\|^2$$

discourages abrupt jumps between MPC iterations while preserving this “join-anywhere” freedom.

This construction accommodates off-path initialization while enforcing forward progression over the horizon [9]–[11], [17]. Temporary slowing or pausing remains possible through the optimized path speed. When the path degenerates to a single point, i.e., $\frac{dr_p}{d\theta} = 0$, the formulation reduces to a setpoint regulation NMPC problem.

To unify path-following, trajectory tracking, and stabilization, we define the augmented state, input, and output variables as:

$$x_a = (\rho, \xi, \zeta, \eta), \quad u_a = (u, \alpha), \quad y_a = (e_p, e_v, e_d) \quad (3.41)$$

with system dynamics and output errors

$$\begin{bmatrix} \dot{\rho} \\ \dot{\xi} \\ \dot{\zeta} \\ \dot{\eta} \end{bmatrix} = \begin{bmatrix} f_\rho(\rho, \xi) \\ f_\xi(\rho, \xi, \zeta, u) \\ f_\zeta(\rho, \xi, \zeta, u) \\ A_\eta \eta + B_\eta \alpha \end{bmatrix}, \quad \begin{bmatrix} e_p \\ e_d \end{bmatrix} = \begin{bmatrix} h_e(\rho) - r_p(\eta_1) \\ h_d(\rho, \xi, \zeta) - r_d \end{bmatrix} \quad (3.42)$$

Here

- $(\rho, \xi, \zeta, \eta, u)$: system states and controls, as defined in the previous sections,
- $\eta_1 = \theta, \eta_2 = v$: variables parameterizing position and speed along the path,
- $\eta = (\eta_1, \dots, \eta_r)$: path-timing states vector, where $\eta_{i+1} = \frac{d^{(i)}\theta}{dt^{(i)}}$
- α : virtual input controlling path progress,
- h_e : mapping between robot's configuration and the pose of its end effector (3.32),
- e_p : geometric path error, with respect to the desired path r_p ,
- e_d : dynamic specification error, with respect to dynamic output reference r_d

This augmented MPPFC structure is aligned with [9]- [11], [16], and accommodates:

- *Path following*: Free evolution along $r_p(\eta_1)$ with no dynamic references
- *Trajectory Tracking*: Fixed timing, i.e., $\eta_1(t)$ is prescribed
- *Stabilization*: No progression, i.e., $\eta_2(t) = 0$

3.8.3 Collision-Aware Geometric Framework

To facilitate an efficient and differentiable collision-avoidance framework, we model robot and obstacles as unions of convex bodies:

$$\mathcal{R}(q, \varphi) = \bigcup_{i \in \mathcal{I}_r} C_i(q, \varphi), \quad \mathcal{O}(q, \varphi) = \bigcup_{j \in \mathcal{I}_o} C_j(q, \varphi) \quad (3.43)$$

where $\mathcal{I}_r$ and $\mathcal{I}_o$ index the constituent convex components of the robot and obstacle sets, respectively. Each convex component is described by a smooth parametric boundary representation. Using the rigid-body transformations introduced in Section 3.3.5, the boundary of a convex body can be written as

$$C_b(q, \varphi) = \{y \in \mathbb{R}^3 \mid y = T(q, \varphi) + R(\varphi)B(s), 0 \leq s \leq 1\}$$

where $B(s)$, $T(q, \varphi)$, and $R(\varphi)$ denote the intrinsic surface parametrization and the configuration-dependent rigid-body transformations defined previously in (3.23)–(3.25). For any pair of bodies i and j (either robot-robot or robot-obstacle), the pairwise minimum distance is defined as

$d_{ij}(q, \varphi) = \min_{y_i, y_j} \ y_i - y_j\ $ $\text{s. t. } y_i \in C_{b_i}(q_i, \varphi_i)$ $y_j \in C_{b_j}(q_j, \varphi_j)$	(3.44)
--	----------

For each relevant body pair $(i, j) \in \mathcal{P}$, collision avoidance requires

$$d_{ij}(q, \varphi) \geq \delta, \quad \forall (i, j) \in \mathcal{P}$$

where $\delta > 0$ denotes the prescribed safety margin and $\mathcal{P}$ denotes the relevant body pairs. Although d_{ij} is defined as a solution to the optimization problem, in practice it may be evaluated in real time using fast algorithms such as the Gilbert-Johnson-Keerthi (GJK) [28] and its Expanding Polytope Algorithm (EPA) extension [30].

All sets are assumed smooth and convex so that the associated distance functions admit the differentiability structure required for the analytical derivative framework developed in Chapter 4 [43]–[45], [77]. Environmental obstacles are

generally described as convex sets as well, and when provided in non-smooth v-polytope form, they are treated under the same framework by assuming that a smooth convex surrogate can be constructed. For predictive collision avoidance, the pairwise constraints in (3.45) may either be enforced individually or combined into a single aggregate constraint. Two formulations may be employed. Several aggregation strategies may be employed:

1. *Individual constraints*: each inequality $d_{ij} \geq \delta$ is enforced separately
2. *Smooth minimum aggregation*: a differentiable surrogate to approximate the minimum pairwise distance as a single smooth constraint.

These aggregation strategies and their implications for optimization and solver performance are examined in detail in Chapter 4. In general, the collision-avoidance requirement can be expressed through an aggregated clearance constraint of the form

$$g_c \left(\{d_{ij}(\rho)\}_{(i,j) \in \mathcal{P}} \right) \leq 0 \quad (3.45)$$

where g_c denotes an aggregation function constructed from the pairwise distances d_{ij} such that the constraint is satisfied only when the required safety margin δ is maintained for all relevant body pairs.

3.8.4 Path Error, Corridor Constraints, and Speed Consistency

In path-following problems, the reference is a geometric curve rather than a time-parameterized trajectory. Therefore, the tracking error is evaluated relative to the local path geometry so that deviations normal to the path are penalized strongly, while motion along the path tangent is penalized only weakly. This avoids the

artificial coupling between path progression and lateral deviation that may arise from component-wise tracking penalties. Such geometric formulations are commonly used in nonlinear model predictive path-following control [9]– [11], [17].

Since the path tracking error $e_p(\rho, \eta_1)$ introduced in (3.42) may include positional as well as orientation components, we introduce a linear operator Π_p that extracts only the position components:

$$e_{pos}(\rho, \eta_1) = \Pi_p e_p(\rho, \eta_1) \in \mathbb{R}^3 \quad (3.46)$$

The unit tangent vector to the path is given by

$$U(\eta_1) = \frac{dr_p(\eta_1)/d\eta_1}{\|dr_p(\eta_1)/d\eta_1\|}$$

At each point along the path, we define an orthonormal moving frame

$$\{U(\eta_1), n_1(\eta_1), n_2(\eta_1)\}$$

where $n_1(\eta_1)$ and $n_2(\eta_1)$ span the normal plane of the path. Define

$$N(\eta_1) = [n_1(\eta_1) \quad n_2(\eta_1)] \in \mathbb{R}^{3 \times 2}$$

which satisfies

$$N^\top(\eta_1)N(\eta_1) = I_2, \quad N^\top(\eta_1)U(\eta_1) = 0 \quad (3.47)$$

Thus, the orthogonal projection onto the normal plane can be written as

$$N(\eta_1)N^\top(\eta_1) = I - U(\eta_1)U^\top(\eta_1)$$

The projection of the positional error onto the normal plane is

$$z(\rho, \eta_1) = N^\top(\eta_1)e_{pos}(\rho, \eta_1) \in \mathbb{R}^2 \quad (3.48)$$

The vector $z(\rho, \eta_1)$ contains the normal component of the path-position error. The primary normal-error penalty is therefore

$$J_{pn}(\rho, \eta_1) = z^\top(\rho, \eta_1) Q_p z(\rho, \eta_1), \quad Q_p > 0$$

Using (3.48), and adding a small full-position regularization, the implemented normal-dominant path-position cost is

$$J_p(\rho, \eta_1) = e_{pos}^\top(\rho, \eta_1) [N(\eta_1) Q_p N^\top(\eta_1) + Q_n I] e_{pos}(\rho, \eta_1), \quad Q_p, Q_n > 0 \quad (3.51)$$

The first term penalizes the normal error, while the small $Q_n I$ term weakly regularizes the full position error, including the tangential direction. In the implementation, Q_n is selected much smaller than the normal tracking weight, typically about 5%, so the controller remains path-following rather than trajectory-tracking. This also improves KKT conditioning and avoids introducing an additional closest-point equality constraint such as

$$U(\eta_1)^T e_{pos}(\rho, \eta_1) = 0$$

The same projected-error structure can also define a path-aligned corridor. Let $W(\eta_1) > 0$ describe the cross-sectional geometry of the path-aligned tube. The corresponding NMPC constraint is

$$g_{cor}(\rho, \eta_1) = e_{pos}^\top(\rho, \eta_1) N(\eta_1) W(\eta_1) N^\top(\eta_1) e_{pos}(\rho, \eta_1) - 1 \leq 0, \quad W(\eta_1) > 0 \quad (3.52)$$

Figure 3.1 shows the cross-section of the path-aligned corridor in the normal plane. For a planar path, the corridor reduces to a lateral interval, whereas for a spatial path, it has an elliptical cross-section.

In addition to path-position tracking, the path-progress rate should remain consistent with the physical motion of the robot along the path tangent. The end-

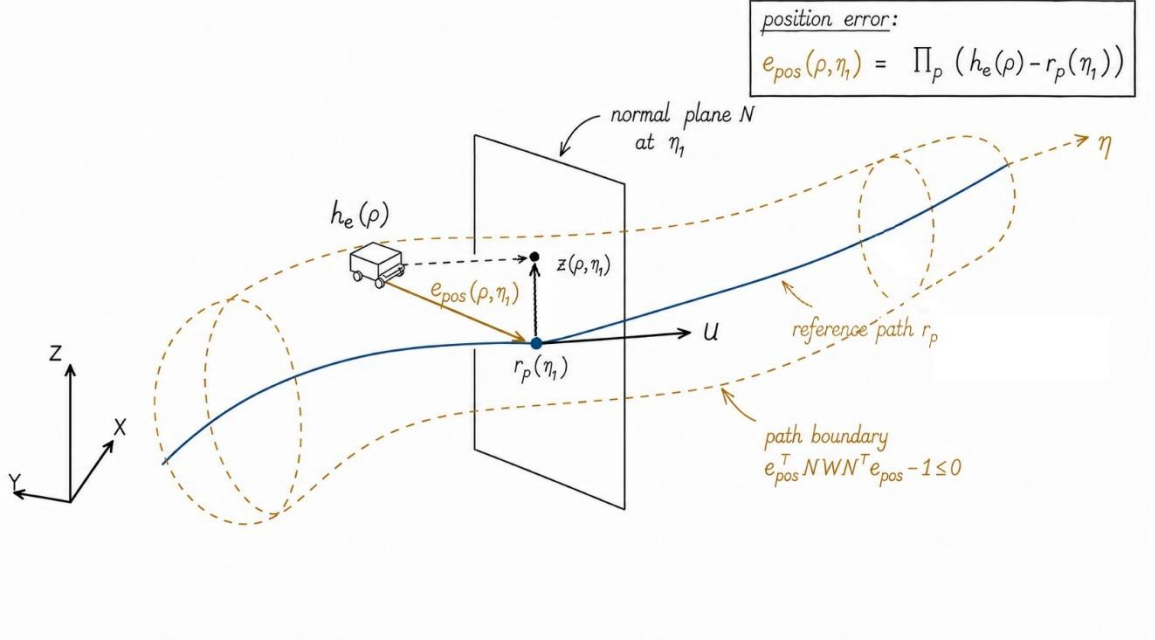


Figure 3.1: Path-Aligned Corridor

effector positional velocity is obtained by differentiating the positional output. Since

$$\dot{h}_{e,pos} = \Pi_p \dot{h}_e$$

and using the kinematic state dynamics, this gives

$$\dot{h}_{e,pos}(\rho) = \Pi_p \frac{\partial h_e(\rho)}{\partial \rho} f_\rho(\rho, \xi) \quad (3.53)$$

The component of the physical velocity along the path tangent is therefore

$$v_{\parallel}(\rho, \xi, \eta_1) = U^T(\eta_1) \dot{h}_{e,pos}(\rho)$$

To enforce consistency between robot motion and path progression, define the reference-speed residual

$$e_v(\rho, \xi, \eta_1) = v_{\parallel}(\rho, \xi, \eta_1) - r_v(\eta_1)$$

where $r_v(\eta_1)$ is a smooth reference speed profile, and the path-progression consistency residual

$$e_\theta(\rho, \xi, \eta_1, \eta_2) = v_\parallel(\rho, \xi, \eta_1) - \eta_2$$

The reference-speed residual is assigned a small weight, while the path-progression consistency residual is assigned dominant weight. Thus, the combined stage cost is

$$J_v(\rho, \xi, \eta_1, \eta_2) = Q_{v,r} \left(v_\parallel(\rho, \xi, \eta_1) - r_v(\eta_1) \right)^2, \quad 0 \leq Q_{v,r} \ll Q_{v,c} \quad (3.54)$$

$$+ Q_{v,c} (v_\parallel(\rho, \xi, \eta_1) - \eta_2)^2$$

This term discourages non-physical path progression and improves numerical conditioning without forcing strict time-parameterized trajectory tracking. Thus, the controller preserves the path-following character of the formulation while maintaining consistency between spatial progression and system dynamics [9]– [11], [17].

Orientation tracking is included by extracting the orientation component of the path-tracking error:

$$e_{ori}(\rho, \eta_1) = \Pi_o e_p(\rho, \eta_1) \quad (3.55)$$

The corresponding quadratic orientation penalty is:

$$J_o(\rho, \eta_1) = e_{ori}^\top(\rho, \eta_1) Q_o e_{ori}(\rho, \eta_1), \quad Q_o > 0 \quad (3.56)$$

As illustrated in Fig. 3.2, the physical velocity is projected onto the local path tangent $U(\eta_1)$ to obtain $v_\parallel$, while the orientation error is measured between the system orientation φ and the desired path orientation φ_d . Both quantities are evaluated at the same path location $r_p(\eta_1)$. This term penalizes deviation from the desired path orientation while remaining compatible with the normal-dominant position-tracking cost and the tangential velocity-consistency term.

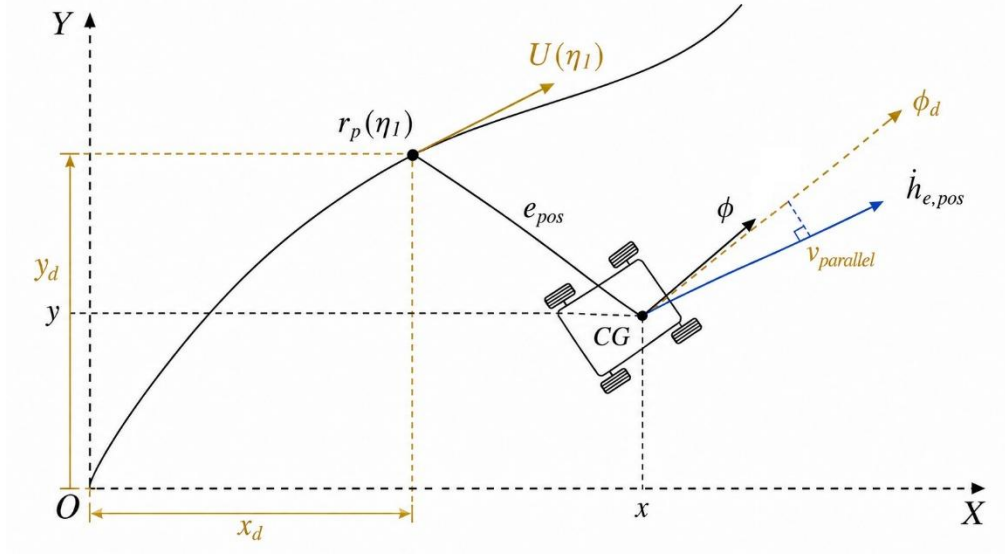


Figure 3.2: Path-Speed Consistency and Orientation Error

3.8.5 Unified Nonlinear MPPFC Formulation

Using the objective function terms defined in previous sections, the stage cost is defined as

$$J_k \triangleq J_p(x_k, \eta_k) + J_o(x_k, \eta_k) + J_v(x_k, \eta_k, \alpha_k) + \|e_{d,k}\|_{Q_{d,k}}^2 + \|u_k\|_{Q_{u,k}}^2 + J_c(x_k, u_k) \quad (3.57)$$

where k denotes the prediction step. All weighting matrices $Q_{(\cdot,k)} \geq 0$ and terms

such as $\|e_{d,k}\|_{Q_{d,k}}^2 = e_{d,k}^T Q_{d,k} e_{d,k}$ follow standard quadratic forms. The custom term

$J_c(x_k, u_k)$ captures optional user-defined stage objectives, such as energy efficiency, actuator smoothness, or other domain-specific penalties. For notational simplicity, the terminal cost is absorbed into the stage cost only at the terminal prediction step:

$$J_k \leftarrow J_k + J_f(x_{t+n_p}, \eta_{t+n_p}) \quad (3.58)$$

The MPPFC problem with collision avoidance is formulated in Eq. (3.59) as a trapezoidally discretized nonlinear optimal control problem solved at each instant t :

$$\begin{aligned}
\{x_k^*, \eta_k^*, u_k^*, \alpha_k^*\}_{k=t}^{t+n_p} = & \underset{\substack{\text{argmin} \\ \{x_k, \eta_k, u_k, \alpha_k\}_{k=t}^{t+n_p} \\ \text{s. t.} \\ \forall k \in [t, t+n_p]}}{\sum_{k=t}^{t+n_p} J_k - P_\theta \eta_{1,t+n_p} + \|\eta_{1,t} - \eta_{1,t-1}\|_{Q_\theta}^2} \\
& x_k = x_{k-1} + \frac{\tau}{2} \begin{bmatrix} f(x_k, u_k) \\ +f(x_{k-1}, u_{k-1}) \end{bmatrix} \\
& \eta_k = \eta_{k-1} + \frac{\tau}{2} \begin{bmatrix} A_\eta(\eta_k + \eta_{k-1}) \\ +B_\eta(\alpha_k + \alpha_{k-1}) \end{bmatrix} \\
& c_e(x_k, \eta_k, u_k, \alpha_k) = 0 \\
& \underline{x}_k \leq x_k \leq \bar{x}_k \\
& \underline{u}_k \leq u_k \leq \bar{u}_k \\
& \underline{\eta}_k \leq \eta_k \leq \bar{\eta}_k \\
& \underline{\alpha}_k \leq \alpha_k \leq \bar{\alpha}_k \\
& c_i(x_k, \eta_k, u_k, \alpha_k) \leq 0
\end{aligned} \tag{3.59}$$

where $x_{t-1} = x_0$, $u_{t-1} = u_0$, $\eta_{t-1} = \eta_0$, $\alpha_{t-1} = \alpha_0$

In this formulation, the inequality constraint vector may be augmented as

$$c_i(\rho, \xi, \zeta, \eta) = \begin{bmatrix} g_c(\{d_{ij}(\rho)\}_{(i,j) \in \mathcal{P}}) \\ g_{cor}(\rho, \eta_1) \\ g_{user}(\rho, \xi, \zeta, \eta) \end{bmatrix} \tag{3.60}$$

where $g_{user}(\rho, \xi, \zeta, \eta)$ represents additional user-defined constraints. Similarly, $c_e = 0$ represents user-defined equality constraints, thereby generalizing the optimal control problem.

Here, $P_\theta \geq 0$ is the scalar terminal path-progression reward weight, while $Q_\theta \geq 0$ is the weight on the inter-cycle path-position regularization term. The term $-P_\theta \eta_{1,t+n_p}$ rewards forward progression along the path when terminal alignment is unspecified. The negative sign is used because the optimization problem is written as a minimization; therefore, increasing terminal path progress decreases the objective. The term $\|\eta_{1,t} - \eta_{1,t-1}\|_{Q_\theta}^2$ discourages abrupt changes in the path-

position variable between successive optimization cycles.

The state and input bounds $(\underline{x}_k, \bar{x}_k)$ and $(\underline{u}_k, \bar{u}_k)$ represent physical or stability constraints. Path variable bounds $\underline{\eta}_k \leq \eta_k \leq \bar{\eta}_k$ and timing-input bounds $\underline{\alpha}_k \leq \alpha_k \leq \bar{\alpha}_k$ allow curvature or speed-based restrictions along the path [16]. Rate penalties on states or controls are excluded to maintain block-diagonal Hessian structure essential for sparse MPC [46]. If additional smoothness is required, it may be introduced through auxiliary state dynamics or filters, thereby preserving separability and enabling efficient optimization solver implementation.

3.9 Unified MPPFC Challenges

This chapter formulated the unified MPPFC problem by combining the robot kinematic representation, path-progress dynamics, physical constraints, and collision-avoidance constraints within a single finite-horizon NMPC problem. The resulting formulation provides the control structure used throughout the dissertation.

The main computational implication is that the collision constraints depend on minimum-distance functions between configuration-dependent convex bodies. These functions must be evaluated and differentiated at each prediction stage. Therefore, the next chapter develops analytical gradients and Hessians of the minimum-distance function, while Chapter 6 develops the structure-exploiting interior-point solver used to solve the resulting NMPC problem efficiently.

3.10 Supplementary Analytical Derivatives for MPPFC

3.10.1 Link Pose Map Derivatives

This section develops the recursive link pose map and its first- and second-order derivatives. The formulation propagates the translation and rotation of each link directly from the previous link pose, which provides a compact structure for implementation in recursive forward-kinematics and optimization routines. Throughout this section, $\mathbf{1}_{(\cdot)}$ denotes the indicator function, which equals one when the stated condition is true and zero otherwise. The operator $\circledast$ appearing in the section recursions denotes the structured matrix product introduced in Chapter 2.

Consider a serial chain with frames $F_0, \dots, F_k$. Specializing the relative transform in (3.15) to the factorized DH form gives

$$H_k^{k-1} = \begin{bmatrix} A_k F_k & A_k p_k(q_k) \\ 0 & 1 \end{bmatrix}, \quad A_k = R_z(\varphi_k), \quad F_k = R_x(\alpha_k) \quad (3.61)$$

where $A_k F_k \in SO(3)$ is the relative rotation, $A_k p_k(q_k) \in \mathbb{R}^3$ is the corresponding DH translation column, $p_k(q_k)$ is the factorized local translation, $\varphi_k \in \mathbb{R}^{n_{\varphi_k}}$, and $q_k \in \mathbb{R}^{n_{q_k}}$. For prismatic coordinates, the translation map is assumed affine in q_k , so that

$$\nabla_{q_k q_k}^2 p_k(q_k) = 0$$

In the Denavit–Hartenberg parametrization, the translation column is commonly written as

$$p_k^{DH}(q_k, \varphi_k) = \begin{bmatrix} l_k \cos \varphi_k \\ l_k \sin \varphi_k \\ q_k \end{bmatrix} \quad (3.62)$$

Equivalently, one may define the factorized local translation

$$p_k(q_k) = \begin{bmatrix} l_k \\ 0 \\ q_k \end{bmatrix}, \quad p_k^{DH}(q_k, \phi_k) = R_z(\phi_k) p_k(q_k) \quad (3.63)$$

Thus, the angular dependence of the DH translation is carried by the relative

rotation, while $p_k(q_k)$ remains independent of φ_k . Therefore,

$$D_{\phi_k} p_k(q_k) = 0 \quad (3.64)$$

Let $P_{0:k}$ denote the cumulative rotation from the base frame to frame F_k , with

$$P_{0:k} = P_{0:k-1} A_k F_k$$

Let S_k denote the constant rotation offset from frame F_k to the link geometry frame and let y_{ls_k} denote the local reference point associated with link k . The cumulative translation and rotation maps are respectively,

$$T_k = \sum_{l=0}^k (P_{0:l-1} A_l p_l) + P_{0:k} y_{ls_k} \quad (3.65)$$

$$R_k = P_{0:k} S_k \quad (3.66)$$

For the base-link term, $A_0 = I$ is used when p_0 denotes the base translation. For link $k-1$,

$$T_{k-1} = \sum_{l=0}^{k-1} P_{0:l-1} A_l p_l + P_{0:k-1} y_{ls_{k-1}}$$

Using $P_{0:k} = P_{0:k-1} A_k F_k$, the translation map can be written as

$$T_k = T_{k-1} + P_{0:k-1} [A_k (p_k + F_k y_{ls_k}) - y_{ls_{k-1}}]$$

Since $R_{k-1} = P_{0:k-1} S_{k-1}$, it follows that

$$P_{0:k-1} = R_{k-1} S_{k-1}^T$$

Define the local displacement and relative link-geometry rotation as

$$\rho_k \triangleq S_{k-1}^T [A_k (p_k + F_k y_{ls_k}) - y_{ls_{k-1}}], \quad E_k \triangleq S_{k-1}^T A_k F_k S_k$$

Then the recursive link pose map is

$$T_k = T_{k-1} + R_{k-1} \rho_k, \quad R_k = R_{k-1} E_k$$

This recursion propagates the link pose directly from link $k - 1$ to link k , while retaining the same forward kinematics as the cumulative-product representation.

Local Derivatives of ρ_k and E_k

The local displacement ρ_k depends only on q_k and φ_k . Since S_{k-1} , y_{ls_k} , and $y_{ls_{k-1}}$ are constant offsets, we obtain

$$D_{q_k} \rho_k = S_{k-1}^T A_k D_{q_k} p_k$$

$$D_{\varphi_k} \rho_k = S_{k-1}^T D_{\varphi_k} A_k \left(I_{n_{\varphi_k}} \otimes (p_k + F_k y_{ls_k}) \right)$$

The corresponding second derivatives are

$$\nabla_{q_k q_k}^2 \rho_k = 0, \quad \nabla_{q_k \varphi_k}^2 \rho_k = S_{k-1}^T D_{\varphi_k} A_k \left(I_{n_{\varphi_k}} \otimes D_{q_k} p_k \right)$$

$$\nabla_{\varphi_k \varphi_k}^2 \rho_k = \left(I_{n_{\varphi_k}} \otimes S_{k-1}^T \right) \nabla_{\varphi_k \varphi_k}^2 A_k \left(I_{n_{\varphi_k}} \otimes (p_k + F_k y_{ls_k}) \right)$$

Similarly, since

$$E_k = S_{k-1}^T A_k F_k S_k$$

its first derivative is

$$D_{\varphi_k} E_k = S_{k-1}^T D_{\varphi_k} A_k \left(I_{n_{\varphi_k}} \otimes (F_k S_k) \right)$$

and its gradient form is

$$\nabla_{\varphi_k} E_k = \left(I_{n_{\varphi_k}} \otimes S_{k-1}^T \right) (\nabla_{\varphi_k} A_k) F_k S_k$$

The second derivative is

$$\nabla_{\varphi_k \varphi_k}^2 E_k = \left(I_{n_{\varphi_k}} \otimes S_{k-1}^T \right) \nabla_{\varphi_k \varphi_k}^2 A_k \left(I_{n_{\varphi_k}} \otimes F_k S_k \right)$$

First Derivatives with Respect to Control Variables

From $T_k = T_{k-1} + R_{k-1} \rho_k$, the derivative with respect to q_i and φ_i are

$$D_{q_i}T_k = \mathbf{1}_{i < k} D_{q_i}T_{k-1} + \mathbf{1}_{i=k} R_{k-1} D_{q_k} \rho_k \quad (3.67)$$

$$D_{\varphi_i}T_k = \mathbf{1}_{i < k} \left[D_{\varphi_i}T_{k-1} + D_{\varphi_i}R_{k-1} \left(I_{n_{\varphi_i}} \otimes \rho_k \right) \right] + \mathbf{1}_{i=k} R_{k-1} D_{\varphi_k} \rho_k \quad (3.68)$$

The rotation map does not depend on the prismatic coordinates, and since $R_k = R_{k-1}E_k$

$$D_{q_i}R_k = 0 \quad (3.69)$$

$$D_{\varphi_i}R_k = \mathbf{1}_{i < k} D_{\varphi_i}R_{k-1} \left(I_{n_{\varphi_i}} \otimes E_k \right) + \mathbf{1}_{i=k} R_{k-1} D_{\varphi_k} E_k \quad (3.70)$$

Let $B_k(s_k)$ denote a point on the local surface or geometry of link k . The world-coordinate point map is

$$y_k = T_k + R_k B_k(s_k)$$

and the point map first derivatives are

$$D_{q_i}y_k = D_{q_i}T_k \quad (3.71)$$

$$D_{\varphi_i}y_k = D_{\varphi_i}T_k + D_{\varphi_i}R_k \left(I_{n_{\varphi_i}} \otimes B_k(s_k) \right) \quad (3.72)$$

Second Derivatives with Respect to Control Variables

The second derivative of T and R can be summarized as:

$$\nabla_{q_i q_j}^2 T_k = 0 \quad (3.73)$$

$$\begin{aligned} \nabla_{q_i \varphi_j}^2 T_k = & \mathbf{1}_{i < k} \mathbf{1}_{j < k} \nabla_{q_i \varphi_j}^2 T_{k-1} + \mathbf{1}_{i=k} \mathbf{1}_{j < k} \left(\nabla_{\varphi_j} R_{k-1} \right) D_{q_k} \rho_k \\ & + \mathbf{1}_{i=k} \mathbf{1}_{j=k} R_{k-1} \nabla_{q_k \varphi_k}^2 \rho_k \end{aligned} \quad (3.74)$$

$$\begin{aligned} \nabla_{\varphi_i \varphi_j}^2 T_k = & \mathbf{1}_{i < k} \mathbf{1}_{j < k} \left[\nabla_{\varphi_i \varphi_j}^2 T_{k-1} + \nabla_{\varphi_i \varphi_j}^2 R_{k-1} \left(I_{n_{\varphi_i}} \otimes \rho_k \right) \right] \\ & + \mathbf{1}_{i < k} \mathbf{1}_{j=k} \left[D_{\varphi_i} R_{k-1} \otimes \nabla_{\varphi_k} \rho_k \right] + \mathbf{1}_{i=k} \mathbf{1}_{j < k} \left[\left(\nabla_{\varphi_j} R_{k-1} \right) D_{\varphi_k} \rho_k \right] \\ & + \mathbf{1}_{i=k} \mathbf{1}_{j=k} \left[\left(I_{n_{\varphi_k}} \otimes R_{k-1} \right) \nabla_{\varphi_k \varphi_k}^2 \rho_k \right] \end{aligned} \quad (3.75)$$

$$\nabla_{q_i q_j}^2 R_k = 0 \quad (3.76)$$

$$\nabla_{q_i \varphi_j}^2 R_k = 0 \quad (3.77)$$

$$\begin{aligned} \nabla_{\varphi_i \varphi_j}^2 R_k = & \mathbf{1}_{i < k} \mathbf{1}_{j < k} \left[\nabla_{\varphi_i \varphi_j}^2 R_{k-1} \left(I_{n_{\varphi_i}} \otimes E_k \right) \right] \\ & + \mathbf{1}_{i < k} \mathbf{1}_{j = k} \left[D_{\varphi_i} R_{k-1} \odot \nabla_{\varphi_k} E_k \right] + \mathbf{1}_{i = k} \mathbf{1}_{j < k} \left[\left(\nabla_{\varphi_j} R_{k-1} \right) D_{\varphi_k} E_k \right] \\ & + \mathbf{1}_{i = k} \mathbf{1}_{j = k} \left[\left(I_{n_{\varphi_k}} \otimes R_{k-1} \right) \nabla_{\varphi_k \varphi_k}^2 E_k \right] \end{aligned} \quad (3.78)$$

and the corresponding point-map second derivatives are

$$\nabla_{q_i q_j}^2 y_k = \nabla_{q_i q_j}^2 T_k \quad (3.79)$$

$$\nabla_{q_i \varphi_j}^2 y_k = \nabla_{q_i \varphi_j}^2 T_k \quad (3.80)$$

$$\nabla_{\phi_i \varphi_j}^2 y_k = \nabla_{\phi_i \varphi_j}^2 T_k + \nabla_{\phi_i \varphi_j}^2 R_k \left(I_{n_{\phi_i}} \otimes B_k(s_k) \right) \quad (3.81)$$

Therefore, all derivatives of the world-coordinate point map can be computed by recursively propagating the derivatives of T_k and R_k from the previous link.

3.10.2 Normal-Dominant Path-Position Cost Derivatives

This subsection records the derivative blocks used in the implementation for the normal-dominant path-position cost defined in Section 3.8.4. The quantities J_p , e_{pos} , z , $N(\eta_1)$, $U(\eta_1)$, Q_p , and Q_n are used as defined there. The first-order derivatives of the positional error are

$$D_x e_{pos} = \Pi_p D_x h_e(x) \quad (3.82)$$

$$D_{\eta_1} e_{pos} = -\Pi_p D_{\eta_1} r_p(\eta_1) \quad (3.83)$$

The first-order derivatives of the projected normal error are

$$D_x z = N^\top D_x e_{pos} \quad (3.84)$$

$$D_{\eta_1} z = (D_{\eta_1} N)^\top e_{pos} + N^\top D_{\eta_1} e_{pos} \quad (3.85)$$

Therefore, the first-order derivatives of J_p are

$$D_x J_p = 2z^\top Q_p D_x z + 2Q_n e_{pos}^\top D_x e_{pos} \quad (3.86)$$

$$D_{\eta_1} J_p = 2z^\top Q_p D_{\eta_1} z + 2Q_n e_{pos}^\top D_{\eta_1} e_{pos} \quad (3.87)$$

Using the stacked Hessian convention introduced in Chapter 2, the second-order derivatives of the positional error are

$$\nabla_{xx}^2 e_{pos} = (I_{n_x} \otimes \Pi_p) \nabla_{xx}^2 h_e(x) \quad (3.88)$$

$$\nabla_{x\eta_1}^2 e_{pos} = 0 \quad (3.89)$$

$$\nabla_{\eta_1\eta_1}^2 e_{pos} = -\Pi_p \nabla_{\eta_1\eta_1}^2 r_p(\eta_1) \quad (3.90)$$

The second-order derivatives of the projected normal error are

$$\nabla_{xx}^2 z = (I_{n_x} \otimes N^\top) \nabla_{xx}^2 e_{pos} \quad (3.91)$$

$$\nabla_{x\eta_1}^2 z = (D_{\eta_1} N)^\top (D_x e_{pos}) \quad (3.92)$$

$$\nabla_{\eta_1\eta_1}^2 z = (\nabla_{\eta_1\eta_1}^2 N)^\top e_{pos} + 2(D_{\eta_1} N)^\top (D_{\eta_1} e_{pos}) + N^\top \nabla_{\eta_1\eta_1}^2 e_{pos} \quad (3.93)$$

The second-order derivative blocks of J_p are then

$$\begin{aligned} \nabla_{xx}^2 J_p &= 2 \left[(D_x z)^\top Q_p (D_x z) + (I_{n_x} \otimes (z^\top Q_p)) \nabla_{xx}^2 z \right] \\ &\quad + 2Q_n [(D_x e_{pos})^\top (D_x e_{pos}) + (I_{n_x} \otimes e_{pos}^\top) \nabla_{xx}^2 e_{pos}] \end{aligned} \quad (3.94)$$

$$\begin{aligned} \nabla_{x\eta_1}^2 J_p &= 2 \left[(D_{\eta_1} z)^\top Q_p (D_x z) + z^\top Q_p \nabla_{x\eta_1}^2 z \right] \\ &\quad + 2Q_n [(D_{\eta_1} e_{pos})^\top (D_x e_{pos}) + e_{pos}^\top \nabla_{x\eta_1}^2 e_{pos}] \end{aligned} \quad (3.95)$$

$$\begin{aligned} \nabla_{\eta_1\eta_1}^2 J_p &= 2 \left[(D_{\eta_1} z)^\top Q_p (D_{\eta_1} z) + z^\top Q_p \nabla_{\eta_1\eta_1}^2 z \right] \\ &\quad + 2Q_n [(D_{\eta_1} e_{pos})^\top (D_{\eta_1} e_{pos}) + e_{pos}^\top \nabla_{\eta_1\eta_1}^2 e_{pos}] \end{aligned} \quad (3.96)$$

3.10.3 Tangential Velocity-Consistency Cost Derivatives

This subsection records the derivative blocks used in the implementation for the tangential velocity-consistency cost defined in Section 3.8.4. The quantities $v_\parallel$, e_v , e_θ , $Q_{v,r}$, and $Q_{v,c}$ are used as defined there. Let

$$y(x) = \dot{h}_{e,pos}(\rho) = \Pi_p \frac{\partial h_e(\rho)}{\partial \rho} f_\rho(\rho, \xi),$$

so that

$$v_\parallel(x, \eta_1) = U^\top(\eta_1)y(x).$$

The first-order derivatives of $v_\parallel$ are

$$D_x v_\parallel = U^\top D_x y, \quad D_{\eta_1} v_\parallel = (D_{\eta_1} U)^\top y, \quad D_{\eta_2} v_\parallel = 0 \quad (3.97)$$

The residual derivatives are

$$D_x e_v = D_x e_\theta = D_x v_\parallel \quad (3.98)$$

$$D_{\eta_1} e_v = D_{\eta_1} v_\parallel - D_{\eta_1} r_v(\eta_1), \quad D_{\eta_1} e_\theta = D_{\eta_1} v_\parallel \quad (3.99)$$

$$D_{\eta_1} e_\theta = D_{\eta_1} v_\parallel, \quad D_{\eta_1} e_\theta = D_{\eta_1} v_\parallel \quad (3.100)$$

Therefore, the first-order derivatives of J_v are

$$D_x J_v = 2Q_{v,r} e_v D_x e_v + 2Q_{v,c} e_\theta D_x e_\theta \quad (3.101)$$

$$D_{\eta_1} J_v = 2Q_{v,r} e_v D_{\eta_1} e_v + 2Q_{v,c} e_\theta D_{\eta_1} e_\theta \quad (3.102)$$

$$D_{\eta_2} J_v = -2Q_{v,c} e_\theta \quad (3.103)$$

The required second-order derivatives of $v_\parallel$ are

$$\nabla_{xx}^2 v_\parallel = (I_{n_x} \otimes U^\top) \nabla_{xx}^2 y \quad (1.104)$$

$$\nabla_{x\eta_1}^2 v_\parallel = (D_{\eta_1} U)^\top D_x y \quad (1.105)$$

$$\nabla_{\eta_1\eta_1}^2 v_\parallel = (\nabla_{\eta_1\eta_1}^2 U)^\top y \quad (1.106)$$

The corresponding residual Hessians are

$$\nabla_{xx}^2 e_v = \nabla_{xx}^2 e_\theta = \nabla_{xx}^2 v_\parallel \quad (3.107)$$

$$\nabla_{xx}^2 e_v = \nabla_{xx}^2 e_\theta = \nabla_{xx}^2 v_\parallel \quad (3.108)$$

$$\nabla_{\eta_1\eta_1}^2 e_v = \nabla_{\eta_1\eta_1}^2 v_\parallel - \nabla_{\eta_1\eta_1}^2 r_v(\eta_1), \quad \nabla_{\eta_1\eta_1}^2 e_\theta = \nabla_{\eta_1\eta_1}^2 v_\parallel \quad (3.109)$$

All second derivatives of the residuals involving η_2 are zero. Hence, the nonzero Hessian blocks of J_v are

$\nabla_{xx}^2 J_v = 2Q_{v,r}[(D_x e_v)^\top D_x e_v + e_v \nabla_{xx}^2 e_v] + 2Q_{v,c}[(D_x e_\theta)^\top D_x e_\theta + e_\theta \nabla_{xx}^2 e_\theta]$	(3.110)
$\nabla_{x\eta_1}^2 J_v = 2Q_{v,r}[(D_{\eta_1} e_v) D_x e_v + e_v \nabla_{x\eta_1}^2 e_v] + 2Q_{v,c}[(D_{\eta_1} e_\theta) D_x e_\theta + e_\theta \nabla_{x\eta_1}^2 e_\theta]$	(3.111)
$\nabla_{\eta_1\eta_1}^2 J_v = 2Q_{v,r}[(D_{\eta_1} e_v)^2 + e_v \nabla_{\eta_1\eta_1}^2 e_v] + 2Q_{v,c}[(D_{\eta_1} e_\theta)^2 + e_\theta \nabla_{\eta_1\eta_1}^2 e_\theta]$	(3.112)
$\nabla_{x\eta_2}^2 J_v = -2Q_{v,c} D_x e_\theta$	(3.113)
$\nabla_{\eta_1\eta_2}^2 J_v = -2Q_{v,c} D_{\eta_1} e_\theta$	(3.114)
$\nabla_{\eta_2\eta_2}^2 J_v = 2Q_{v,c}$	(3.115)

3.11 Chapter Summary

This chapter defined the notation and finite-dimensional rigid-body system representation used throughout the dissertation. Generalized configuration coordinates, nonlinear configuration manifolds, rigid-body transformations, forward kinematics, actuator and internal-state augmentation, and task-output maps were summarized to establish a common modeling basis for the later optimization problems.

Using this representation, the chapter formulated the unified MPPFC problem by augmenting the physical system with path-progression variables. The resulting formulation supports path following, trajectory tracking, and stabilization within a single NMPC structure. Collision avoidance was incorporated through differentiable clearance constraints between convex body representations, while the path-following objective was decomposed into normal-dominant position tracking, weak tangential regularization, tangential velocity consistency, and

orientation tracking.

The formulation developed in this chapter provides the foundation for the remaining technical developments. In particular, the next chapter derives analytical gradients and Hessians of the minimum-distance functions used in the collision constraints, and Chapter 6 develops the structure-exploiting interior-point NMPC solver used to solve the resulting finite-horizon optimization problem efficiently.

CHAPTER 4 - ANALYTICAL DISTANCE DERIVATIVES FOR ARTICULATED ROBOTIC SYSTEMS

4.1 Introduction and Motivation

Chapter 3 formulated the unified MPPFC problem and showed that collision avoidance enters the finite-horizon optimization through minimum-distance constraints between configuration-dependent robot and obstacle bodies. This chapter derives analytical gradients and Hessians of these distance functions for smooth convex bodies and extends the resulting expressions to articulated robotic systems.

Consider two rigid bodies i and j , represented by closed, bounded convex sets C_i and C_j . Their boundaries are denoted by

$$C_{b_i} = \partial C_i, \quad C_{b_j} = \partial C_j \quad (4.1)$$

The convexity assumption ensures a well-defined closest-point structure. In practical robotic systems, nonconvex bodies can be replaced by smooth conservative convex hulls that enclose the original geometry. The derivative development in this chapter can then be applied to each resulting convex body pair.

Let body i belong to kinematic chain A , whose configuration is described by the translational and rotational coordinates

$$\rho_a = (q_a, \varphi_a) \in R^{n_{q_a}} \times R^{n_{\varphi_a}} \quad (4.2)$$

and let body j belong to kinematic chain B , whose configuration is

$$\rho_b = (q_b, \varphi_b) \in R^{n_{q_b}} \times R^{n_{\varphi_b}} \quad (4.3)$$

For the pairwise distance calculation, define the combined variables

$$q = (q_a, q_b), \quad \varphi = (\varphi_a, \varphi_b) \quad (4.4)$$

which collect the translational and rotational coordinates relevant to the two bodies. Here q and φ may represent independent robot chains, a robot and an obstacle, or two bodies within a larger articulated system. If the bodies belong to the same kinematic chain, the corresponding shared coordinates are handled through the forward-kinematic mappings introduced in Chapter 3. The minimum Euclidean distance between the bodies is then defined as

$$d_{ij}(q, \varphi) = \begin{cases} \min_{y_i, y_j} \|y_i - y_j\| \\ \text{s.t. } y_i \in C_{b_i}(q_a, \varphi_a) \\ y_j \in C_{b_j}(q_b, \varphi_b) \end{cases} \quad (4.5)$$

The points y_i and y_j are boundary points expressed in the world frame. They are obtained by transforming body-fixed surface points through the rigid-body transformations and kinematic-chain mappings associated with the configuration variables. When the bodies are nonintersecting and the closest-point pair is unique, d_{ij} defines the local clearance between the two bodies.

Let $\mathcal{P}$ denote the set of relevant body pairs considered for collision avoidance. For each $(i, j) \in \mathcal{P}$, the distance may be used as an inequality constraint

$$d_{ij}(q, \varphi) \geq \delta, \quad \forall (i, j) \in \mathcal{P} \quad (4.6)$$

or as part of a penalty, barrier, or smooth aggregate clearance function. In the MPPFC formulation developed in Chapter 3, such distance functions appear repeatedly across prediction stages and body pairs. Consequently, the solver requires not only d_{ij} , but also its derivatives with respect to the variables that determine the body

poses. The required quantities are the gradient $\nabla_{(q,\varphi)} d_{ij}(q, \varphi)$ and, for second-order optimization, the Hessian $\nabla_{(q,\varphi)}^2 d_{ij}(q, \varphi)$.

The numerical evaluation of the distance may be performed using established proximity-query algorithms. The contribution of this chapter is not a new distance-evaluation algorithm, but the derivation of analytical derivatives after the closest-point pair has been determined. This distinction is important because proximity-query methods provide the closest points and distance value, whereas the NMPC solver requires consistent gradient and Hessian information with respect to the configuration variables.

The main difficulty is that d_{ij} is the value of an inner closest-point problem. As the bodies move, the optimal surface parameters also change, and this variation contributes to the distance derivatives. Therefore, differentiating only the rigid-body transformations while holding the closest points fixed is not sufficient. Instead, the closest-point optimality conditions of (4.5) are differentiated implicitly to obtain the sensitivities of the optimal surface parameters with respect to configuration.

The resulting pairwise derivative expressions provide the collision-gradient and collision-Hessian building blocks used later in the dissertation. For articulated systems, the pose-level derivatives are propagated through the kinematic chain to obtain derivatives with respect to generalized coordinates. For multi-link robots or multi-obstacle environments, the pairwise derivatives are assembled across active body pairs and supplied to the structure-exploiting NMPC solver developed in Chapter 6.

4.2 Minimum Distance Formulation

The minimum distance $d_{ij}(q, \varphi)$ in (4.5) is the minimum Euclidean distance between points on the body boundaries C_{b_i} and C_{b_j} . This section rewrites that distance using the parametric boundary representation introduced in Chapter 3, so that the closest-point problem can be differentiated with respect to the configuration variables.

Recall from eq. (3.23) that a point on the boundary of rigid body $k \in \{i, j\}$ can be written in world coordinates as

$$y_k(q, \varphi, s_k) = T_k(q, \varphi) + R_k(\varphi)B_k(s_k), \quad s_k \in [0, 1]^2 \quad (4.7)$$

where (q, φ) denotes the stacked vector of translational and rotational variables associated with the articulated system containing body k , as defined in Eqn. (3.21), $T_k(q, \varphi)$ is the translational component of the rigid-body transformation as given in Eq. (3.24), $R_k(\varphi)$ is the rotation matrix defined in Eq. (3.25), and $B_k(s_k)$ is the body-fixed surface parameterization, as defined in (3.11).

Substituting this representation into the distance definition in Eq. (4.5) yields the parametric optimization problem

$$d_{ij}(q, \varphi) = \begin{cases} \min_{s_i, s_j} \|T_i(q, \varphi) - T_j(q, \varphi) + R_i(\varphi)B_i(s_i) - R_j(\varphi)B_j(s_j)\| \\ s.t. \quad 0 \leq s_i \leq 1 \\ \quad \quad 0 \leq s_j \leq 1 \end{cases} \quad (4.8)$$

This notation applies to self-collision, robot-robot, and robot-obstacle distances by associating each body with the relevant subset of configuration variables. If a body is static, its transformation is treated as constant.

To facilitate the use of unconstrained optimization methods, the bounded surface parameters are mapped to unconstrained variables. Define the scalar maps

$$s(\vartheta) = \frac{1}{2}(1 - \cos \vartheta), \quad \vartheta = \cos^{-1}(1 - 2s), \quad \vartheta \in [0, 2\pi) \quad (4.9)$$

which maps $\vartheta \in \mathbb{R}$ to the interval $[0, 1]$. For $k \in \{i, j\}$, let

$$\vartheta_k = \text{vec}(\vartheta_{k_1}, \vartheta_{k_2}), \quad s_k = \text{vec}(s_{k_1}, s_{k_2})$$

with

$$s_k = s(\vartheta_k)$$

Using this transformation, the constrained optimization problem (4.8) can be written equivalently as the unconstrained problem:

$d_{ij}(q, \varphi) = \min_{\vartheta} \ y_{ij}(q, \varphi, \vartheta)\ \quad (4.10)$
where:
$y_{ij}(q, \varphi, \vartheta) = I_{\pm}(T(q, \varphi) + R(\varphi)B(\vartheta)) \quad (4.11)$

with the following definitions:

- $T(q, \varphi) = \begin{bmatrix} T_i(q, \varphi) \\ T_j(q, \varphi) \end{bmatrix}, \quad R(\varphi) = \begin{bmatrix} R_i(\varphi) & 0 \\ 0 & R_j(\varphi) \end{bmatrix}$
- $B(\vartheta) = \text{vec}\left((B_i \circ s)(\vartheta_i), (B_j \circ s)(\vartheta_j)\right)$
- $s = \text{vec}(s_i, s_j), \quad \vartheta = \text{vec}(\vartheta_i, \vartheta_j)$
- $I_{\pm} = \begin{bmatrix} I & -I \end{bmatrix}$ is the block difference operator

This reformulation expresses the minimum-distance problem entirely in terms of the parameter vector ϑ , enabling the application of unconstrained optimization techniques and providing a differentiable structure suitable for

deriving gradient and Hessian expressions.

Fig. 4.1 shows the closest-point vector for a selected link–obstacle pair; and its norm is the pairwise minimum distance.

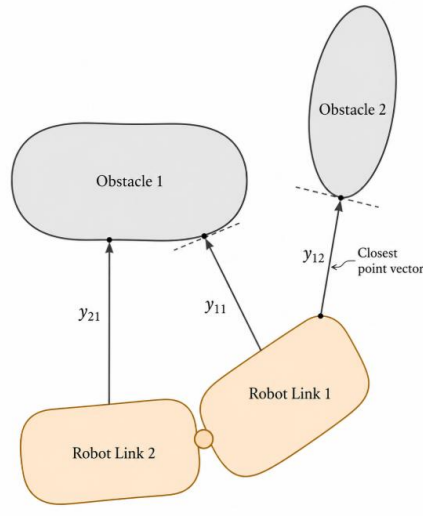


Figure 4.1: Pairwise Distance Vectors

Lemma 4.1 – Closest Points Lie on the Boundaries

Let $C_i, C_j \subset \mathbb{R}^3$ be non-empty closed, bounded, convex sets such that $C_i \cap C_j = \emptyset$. Define the closest point pair as:

$$(y_i^*, y_j^*) = \underset{y_i \in C_i, y_j \in C_j}{\operatorname{argmin}} \|y_i - y_j\| \quad (4.12)$$

Then, the minimizer lies on the boundaries of their respective sets

$$y_i^* \in C_{b_i}, \quad y_j^* \in C_{b_j}$$

where C_{b_i} and C_{b_j} are the boundaries of C_i and C_j , respectively

Proof

Suppose, for contradiction, that y_i^* lies in the interior of C_i . Then there exist an $\varepsilon > 0$ such that the open ball $B_\varepsilon(y_i^*) \subset C_i$. Let

$$u = \frac{y_j^* - y_i^*}{\|y_i^* - y_j^*\|}$$

be a unit vector from y_i^* to y_j^* . For any $0 < \delta < \varepsilon$, the point $y_i' = y_i^* + \delta u \in C_i$. Then

$$\|y_i' - y_j^*\| = \|y_i^* - y_j^*\| - \delta < \|y_i^* - y_j^*\|$$

which contradicts the optimality of y_i^* . An identical argument applies to y_j^* .

Therefore, both y_i^* and y_j^* must lie on the boundaries of their respective sets, given by C_{b_i} and C_{b_j} respectively.

This is a standard result in convex analysis: for any two non-intersecting, closed convex sets, the minimum distance is always achieved at a pair of boundary points [77]. These results guarantee the existence and boundary localization of the closest points between convex sets, providing the theoretical foundation for the formulation in Eq. (4.8) and Eq. (4.10) of this chapter.

Theorem 4.2 – Closest-Point Problem and First-Order Stationarity

Let C_i and C_j be two closed, bounded convex sets in $\mathbb{R}^3$ with smooth differentiable boundaries C_{b_i} and C_{b_j} , whose world-frame boundary points are parameterized by Eq. (4.7). Suppose the following are satisfied:

1. Non-intersection: $C_i \cap C_j = \emptyset$.
2. Smoothness: The boundary mappings $B_i(\cdot)$ and $B_j(\cdot)$ are C^1 smooth.
3. Strict convexity: In addition to the convexity of both sets stated above, at least one of C_i or C_j is strictly convex.
4. Smooth kinematics: The mappings $T_k(q, \varphi)$ and $R_k(\varphi)$, for $k \in \{i, j\}$, are C^1 smooth in (q, φ) .

Then:

1. The spatial closest-point pair (y_i^*, y_j^*) between C_i and C_j exists and is unique.
2. Eq. (4.10) has at least one global minimizer $\vartheta^*(q, \varphi)$ that maps to this unique spatial closest-point pair.
3. Any such global minimizer $\vartheta^*(q, \varphi)$ satisfies the first-order stationarity condition

$$(\nabla_{\vartheta} B^{*\top}) R^{\top} I_{\pm}^{\top} y_{ij}^* = 0 \quad (4.13)$$

where

$$y_{ij}^*(q, \varphi) = y_{ij}(q, \varphi, \vartheta^*(q, \varphi)), \quad B^*(q, \varphi) = (B \circ \vartheta^*)(q, \varphi) \quad (4.14)$$

Proof:

Existence and uniqueness of the spatial closest-point pair

Since C_i and C_j are closed and bounded, they are compact subsets of $\mathbb{R}^3$.

Therefore, the Cartesian product $C_i \times C_j$ is compact. The function

$$(y_i, y_j) \mapsto \|y_i - y_j\|$$

is continuous on $C_i \times C_j$. Hence, there exists at least one pair $(y_i^*, y_j^*) \in C_i \times C_j$ that attains the minimum distance between C_i and C_j . Because the sets are non-intersecting, Lemma 4.1 implies that any closest-point pair lies on the boundaries C_{b_i} and C_{b_j} . Since both sets are convex and at least one is strictly convex, the spatial closest-point pair (y_i^*, y_j^*) is unique [77].

Connection to the parameterized problem

By Eq. (4.7), every boundary point $y_k^* \in C_{b_k}$, $k \in \{i, j\}$, can be represented by

at least one surface parameter value. Therefore, there exists at least one parameter vector $\vartheta^*(q, \varphi)$ that maps to the unique spatial closest-point pair (y_i^*, y_j^*) . Since (y_i^*, y_j^*) realizes the minimum Euclidean distance between C_i and C_j , any parameter vector $\vartheta^*(q, \varphi)$ that maps to this pair is a global minimizer of Eq. (4.10). The parameter vector itself need not be unique because the surface parameterization may be periodic, but all such parameter vectors produce the same closest-point pair and the same minimum distance.

First-order stationarity

Define

$$g(y_{ij}) = y_{ij}^\top y_{ij}, \quad f(g) = g^{1/2}$$

so that

$$\|y_{ij}\| = (f \circ g \circ y_{ij})(q, \varphi, \vartheta)$$

For non-intersecting bodies, $y_{ij} \neq 0$, so g is differentiable with respect to ϑ . From the chain rule for matrix differentials (2.57):

$$\nabla_{\vartheta} \|y_{ij}\| = \|y_{ij}\|^{-1} (\nabla_{\eta} y_{ij}^\top) y_{ij}$$

Since:

$$\nabla_{\vartheta} y_{ij}^\top = \nabla_{\vartheta} (B^\top(\vartheta) R^\top(\varphi) I_{\pm}^\top)$$

we obtain:

$$\nabla_{\vartheta} \|y_{ij}\| = \|y_{ij}\|^{-1} (\nabla_{\vartheta} B^\top) R^\top I_{\pm}^\top y_{ij}$$

At the minimizer ϑ^* , the first order optimality condition requires:

$$\|y_{ij}^*\|^{-1} (\nabla_{\vartheta} B^{*\top}) R^\top I_{\pm}^\top y_{ij}^* = 0$$

Since C_i and C_j are non intersecting, $\|y_{ij}^*\| > 0$ and hence

$$(\nabla_{\vartheta} B^{*\top}) R^{\top} I_{\pm}^{\top} y_{ij}^* = 0$$

4.3 Analytical Distance Gradients

Theorem 4.3 – Gradient of Minimum Distance Between Two Smooth Parametric Convex Surfaces

Under the assumptions of theorem 4.2, consider the minimum distance,

$$d_{ij}(q, \varphi) = \|y_{ij}^*(q, \varphi)\| > 0 \quad (4.15)$$

where the optimal separation vector is

$$y_{ij}^*(q, \varphi) = I_{\pm}(T(q, \varphi) + R(\varphi)B^*(q, \varphi)), \quad B^*(q, \varphi) = (B \circ \vartheta^*)(q, \varphi) \quad (4.16)$$

Then the gradients of d_{ij} with respect to translation q and rotation parameters φ are

$$\nabla_q d_{ij}(q, \varphi) = \|y_{ij}^*\|^{-1} (\nabla_q T^{\top}) I_{\pm}^{\top} y_{ij}^* \quad (4.17)$$

$$\nabla_{\varphi} d_{ij}(q, \varphi) = \|y_{ij}^*\|^{-1} \left(\nabla_{\varphi} T^{\top} + (I_{n_{\varphi}} \otimes B^{*\top}) \nabla_{\varphi} R^{\top} \right) I_{\pm}^{\top} y_{ij}^* \quad (4.18)$$

Proof

Derivative with respect to q

Taking derivative of (4.15) and using chain rule (2.57)

$$\nabla_q d_{ij} = \|y_{ij}^*\|^{-1} (\nabla_q y_{ij}^{*\top}) y_{ij}^*$$

Using the expression (4.16)

$$y_{ij}^* = I_{\pm}(T + RB^*) \Rightarrow y_{ij}^{*\top} = (T^{\top} + B^{*\top} R^{\top}) I_{\pm}^{\top}$$

Differentiate

$$\nabla_q y_{ij}^{*\top} = (\nabla_q T^{\top} + (\nabla_q \vartheta^{*\top})(\nabla_{\vartheta} B^{*\top}) R^{\top}) I_{\pm}^{\top}$$

Using chain rule

$$\nabla_q B^{*\top} = (\nabla_q \vartheta^{*\top})(\nabla_\vartheta B^{*\top})$$

Hence

$$\nabla_q d_{ij} = \|y_{ij}^*\|^{-1} (\nabla_q T^\top + (\nabla_q \vartheta^{*\top})(\nabla_\vartheta B^{*\top}) R^\top) I_\pm^\top y_{ij}^*$$

But from the first-order KKT condition (Eq. 4.13)

$$(\nabla_\vartheta B^{*\top}) R^\top I_\pm^\top y_{ij}^* = 0$$

Implying

$$\nabla_q d_{ij} = \|y_{ij}^*\|^{-1} (\nabla_q T^\top) I_\pm^\top y_{ij}^*$$

Derivative with respect to φ

By the chain rule

$$\nabla_\varphi d_{ij} = \|y_{ij}^*\|^{-1} (\nabla_\varphi y_{ij}^{*\top}) y_{ij}^*$$

Using the product rule and the chain rule

$$\nabla_\varphi y_{ij}^{*\top} = (\nabla_\varphi T^\top + (\nabla_\varphi \vartheta^{*\top})(\nabla_\vartheta B^{*\top}) R^\top + (I_{n_\varphi} \otimes B^{*\top}) \nabla_\varphi R^\top) I_\pm^\top$$

Substitute

$$\nabla_\varphi B^{*\top} = (\nabla_\varphi \vartheta^{*\top})(\nabla_\vartheta B^{*\top})$$

So that

$$\nabla_\varphi d_{ij} = \|y_{ij}^*\|^{-1} (\nabla_\varphi T^\top + (\nabla_\varphi \vartheta^{*\top})(\nabla_\vartheta B^{*\top}) R^\top + (I_{n_\varphi} \otimes B^{*\top}) \nabla_\varphi R^\top) I_\pm^\top y_{ij}^*$$

Again, the KKT condition (Eq. 4.13), eliminates the second term resulting in

$$\nabla_\varphi d_{ij} = \|y_{ij}^*\|^{-1} (\nabla_\varphi T^\top + (I_{n_\varphi} \otimes B^{*\top}) \nabla_\varphi R^\top) I_\pm^\top y_{ij}^*$$

The derivation of $\nabla_\varphi \vartheta^{*\top}$ and $\nabla_\varphi \vartheta^{*\top}$ shall be discussed in the next section.

The first-order stationarity condition in Eq. (4.13) can also be differentiated

to obtain the sensitivity of the optimal closest-point vector with respect to the configuration variables. This sensitivity is required for the Hessian calculation and is derived next.

Lemma 4.4 - Gradient of the Optimal Closest-Point Vector

Let

$$y_{ij}^*(q, \varphi) = I_{\pm}(T(q, \varphi) + R(\varphi)(B \circ \vartheta^*)(q, \varphi)) \quad (4.19)$$

where $\vartheta^*(q, \varphi)$ is the optimizer of the non-intersecting convex surface distance problem defined in (4.10). Assume the conditions of Theorem 4.2 are satisfied. In addition, assume that, at the closest points y_i^* and y_j^* , the active boundary patches of C_i and C_j are C^2 , regular, and locally strictly convex. Equivalently, the closest points lie on smooth, curved boundary patches, not on edges, corners, flat faces, or parameter singularities. Define the following matrices:

$$J_{\vartheta} = (\nabla_{\vartheta} B^{*\top}) R^{\top} I_{\pm}^{\top} \quad (4.20)$$

$$C = [I_{n_{\vartheta}} \otimes (y_{ij}^{*\top} I_{\pm} R)] \nabla_{\vartheta\vartheta}^2 B^* \quad (4.21)$$

$$A = J_{\vartheta} J_{\vartheta}^{\top} + C \quad (4.22)$$

$$V = -(\nabla_q T^{\top}) I_{\pm}^{\top} J_{\vartheta}^{\top} \quad (4.23)$$

$$W = -(\nabla_{\varphi} T^{\top}) I_{\pm}^{\top} J_{\vartheta}^{\top} - [I_{n_{\varphi}} \otimes B^{*\top}] (\nabla_{\varphi} R^{\top}) I_{\pm}^{\top} J_{\vartheta}^{\top} - [I_{n_{\varphi}} \otimes (y_{ij}^{*\top} I_{\pm})] (\nabla_{\varphi} R) (\nabla_{\vartheta} B^{*\top})^{\top} \quad (4.24)$$

Under these local boundary-shape conditions, the stationarity matrix A is nonsingular at the closest-point solution. Then the gradients of the optimal closest point function $y_{ij}^*(q, \varphi)$ are given by:

$$\nabla_q y_{ij}^{*\top}(q, \varphi) = [\nabla_q T^{\top} + V A^{-1} (\nabla_{\vartheta} B^{*\top}) R^{\top}] I_{\pm}^{\top} \quad (4.25)$$

$$\nabla_{\varphi} y_{ij}^{*\top}(q, \varphi) = \left[\nabla_{\varphi} T^{\top} + W A^{-1} (\nabla_{\vartheta} B^{*\top}) R^{\top} + (I_{n_{\varphi}} \otimes B^{*\top}) \nabla_{\varphi} R^{\top} \right] I_{\pm}^{\top} \quad (4.26)$$

Proof:

Derivative with respect to q

Differentiate the transpose of Eq. (4.13) w.r.t. q

$$\nabla_q \left(\nabla_{\vartheta} B^{*\top}(q, \varphi) R^{\top}(\varphi) I_{\pm}^{\top} y_{ij}^{*}(q, \varphi) \right)^{\top} = 0$$

Expand using product rule (2.41)

$$(\nabla_q y_{ij}^{*\top}) I_{\pm} R (\nabla_{\vartheta} B^{*\top})^{\top} + (I_{n_q} \otimes (y_{ij}^{*\top} I_{\pm} R)) \nabla_q (\nabla_{\vartheta} B^{*\top})^{\top} = 0$$

Use chain rule (2.57) and the definition of second derivative (2.32),

$$\nabla_q (\nabla_{\vartheta} B^{*\top})^{\top} = (\nabla_q \vartheta^{*\top} \otimes I_{n_q}) \nabla_{\vartheta\vartheta}^2 B^*$$

and simplify using identity (2.13),

$$(I_{n_q} \otimes (y_{ij}^{*\top} I_{\pm} R)) (\nabla_q \vartheta^{*\top} \otimes I_{n_q}) = \nabla_q \vartheta^{*\top} (I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R))$$

we obtain

$$(\nabla_q y_{ij}^{*\top}) I_{\pm} R (\nabla_{\vartheta} B^{*\top})^{\top} + \nabla_q \vartheta^{*\top} (I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R)) \nabla_{\vartheta\vartheta}^2 B^* = 0$$

Differentiate $y_{ij}^{*\top}$ in (4.19)

$$\nabla_q y_{ij}^{*\top} = (\nabla_q T^{\top} + (\nabla_q \vartheta^{*\top}) (\nabla_{\vartheta} B^{*\top}) R^{\top}) I_{\pm}^{\top}$$

Substitute into the previous equation:

$$(\nabla_q T^{\top} + (\nabla_q \vartheta^{*\top}) (\nabla_{\vartheta} B^{*\top}) R^{\top}) I_{\pm}^{\top} I_{\pm} R (\nabla_{\vartheta} B^{*\top})^{\top} + \nabla_q \vartheta^{*\top} (I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R)) \nabla_{\vartheta\vartheta}^2 B^* = 0$$

Group terms

$$\begin{aligned}
& \nabla_q \vartheta^{*\top} \left[(\nabla_{\vartheta} B^{*\top} R^{\top} I_{\pm}^{\top}) (\nabla_{\vartheta} B^{*T} R^T I_{\pm}^T)^{\top} + (I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R)) \nabla_{\vartheta\vartheta}^2 B^* \right] \\
& = -\nabla_q T^{\top} I_{\pm}^T (\nabla_{\vartheta} B^{*\top} R^{\top} I_{\pm}^{\top})^{\top}
\end{aligned}$$

and define

$$\begin{aligned}
J_{\vartheta} & \triangleq (\nabla_{\vartheta} B^{*T}) R^T I_{\pm}^T \\
C & \triangleq (I_{n_s} \otimes (y_{ij}^{*T} I_{\pm} R)) \nabla_{\vartheta\vartheta}^2 B^* \\
A & \triangleq J_{\vartheta} J_{\vartheta}^{\top} + C \\
V & \triangleq -(\nabla_q T^T) I_{\pm}^T J_{\vartheta}^{\top}
\end{aligned}$$

Then

$$\nabla_q \vartheta^{*T} = V A^{-1} \quad (4.27)$$

Substitute back in the equation for $\nabla_q y_{ij}^{*T}$

$$\nabla_q y_{ij}^{*T} = (\nabla_q T^T + V A^{-1} \nabla_{\vartheta} B^{*T} R^T) I_{\pm}^T$$

The nonsingularity of A follows from the local boundary-shape assumptions and is discussed at the end of the proof.

Derivative with respect to φ

Repeat the procedure

$$\begin{aligned}
& (\nabla_{\varphi} y_{ij}^{*\top}) I_{\pm} R (\nabla_{\vartheta} B^{*\top})^{\top} + [I_{n_{\varphi}} \otimes (y_{ij}^{*\top} I_{\pm})] (\nabla_{\varphi} R) (\nabla_{\vartheta} B^{*\top})^{\top} \\
& + [I_{n_{\varphi}} \otimes (y_{ij}^{*\top} I_{\pm} R)] \nabla_{\varphi} (\nabla_{\vartheta} B^{*\top})^{\top} = 0
\end{aligned}$$

Again, by applying chain rule (2.57) and the definition of second derivative (2.32)

$$\nabla_{\varphi} (\nabla_{\vartheta} B^{*\top})^{\top} = (\nabla_{\varphi} \vartheta^{*T} \otimes I_{n_{\varphi}}) \nabla_{\vartheta\vartheta}^2 B^*$$

substituting

$$\nabla_{\varphi} y_{ij}^{*\top} = \left[\nabla_{\varphi} T^{\top} + (\nabla_{\varphi} \vartheta^{*\top})(\nabla_{\vartheta} B^{*\top}) R^{\top} + (I_{n_{\varphi}} \otimes B^{*\top}) \nabla_{\varphi} R^{\top} \right] I_{\pm}^{\top}$$

and simplifying using identity (2.13)

$$\left(I_{n_{\varphi}} \otimes (y_{ij}^{*\top} I_{\pm} R) \right) (\nabla_{\varphi} \vartheta^{*\top} \otimes I_{n_{\varphi}}) = \nabla_{\varphi} \vartheta^{*\top} (I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R))$$

we obtain

$$\begin{aligned} \nabla_{\varphi} \vartheta^{*\top} & \left[(\nabla_{\vartheta} B^{*\top} R^{\top} I_{\pm}^{\top}) (\nabla_{\vartheta} B^{*\top} R^{\top} I_{\pm}^{\top})^{\top} + (I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R)) \nabla_{\vartheta \vartheta}^2 B^* \right] \\ & = -\nabla_{\varphi} T^{\top} I_{\pm}^{\top} I_{\pm} R (\nabla_{\vartheta} B^{*\top})^{\top} - (I_{n_{\varphi}} \otimes B^{*\top}) \nabla_{\varphi} R^{\top} I_{\pm}^{\top} (\nabla_n B^{*\top} R^{\top} I_{\pm}^{\top})^{\top} \\ & \quad - (I_{n_{\varphi}} \otimes (y_{ij}^{*\top} I_{\pm})) (\nabla_{\varphi} R) (\nabla_{\vartheta} B^{*\top})^{\top} \end{aligned}$$

Define

$$W \triangleq -(\nabla_{\varphi} T^{\top}) I_{\pm}^{\top} J_{\vartheta}^{\top} - (I_{n_{\varphi}} \otimes B^{*\top}) (\nabla_{\varphi} R^{\top}) I_{\pm}^{\top} J_{\vartheta}^{\top} - (I_{n_{\varphi}} \otimes (y_{ij}^{*\top} I_{\pm})) (\nabla_{\varphi} R) (\nabla_{\vartheta} B^{*\top})^{\top}$$

Then

$$\nabla_{\varphi} \vartheta^{*\top} = W A^{-1} \tag{4.28}$$

Substitute back in the equation for $\nabla_{\varphi} y_{ij}^{*\top}$

$$\nabla_{\varphi} y_{ij}^{*\top} = \left[\nabla_{\varphi} T^{\top} + W A^{-1} (\nabla_{\vartheta} B^{*\top}) R^{\top} + (I_{n_{\varphi}} \otimes B^{*\top}) \nabla_{\varphi} R^{\top} \right] I_{\pm}^{\top}$$

Invertibility of the Matrix A

The matrix A is composed of a Gram term and a curvature term. The Gram matrix term is

$$J_{\vartheta} J_{\vartheta}^{\top} = ((\nabla_{\vartheta} B^{*\top}) R^{\top} I_{\pm}^{\top}) ((\nabla_{\vartheta} B^{*\top}) R^{\top} I_{\pm}^{\top})^{\top}$$

For any vector z ,

$$z^{\top} J_{\vartheta} J_{\vartheta}^{\top} z = \| J_{\vartheta}^{\top} z \|^2 \geq 0$$

Therefore, $J_{\vartheta}J_{\vartheta}^{\top}$ is symmetric and positive semi-definite. This means that the Gram term cannot introduce negative curvature. However, positive semi-definiteness alone does not guarantee that $J_{\vartheta}J_{\vartheta}^{\top}$, or A , is nonsingular. The nonsingularity of A must therefore be established using the curvature correction term. Now consider the curvature correction term

$$C = \left(I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R) \right) \nabla_{\vartheta\vartheta}^2 B^*$$

Use $I_{\pm} = [I \quad -I]$ and block-diagonal $R = \text{blkdiag}(R_i, R_j)$ to write

$$y_{ij}^{*\top} I_{\pm} R = [y_{ij}^{*\top} R_i \quad -y_{ij}^{*\top} R_j]$$

Under the closest-point regularity assumptions used in this section, y_{ij}^* is the locally unique shortest vector and is orthogonal to both active boundaries at the closest points; hence it is colinear with the corresponding surface normals. Define

$$n_i = \frac{R_i^{\top} y_{ij}^*}{\|y_{ij}^*\|}, \quad n_j = -\frac{R_j^{\top} y_{ij}^*}{\|y_{ij}^*\|}$$

which are unit normal in each local frame at contact points. Then

$$y_{ij}^{*\top} R_i = \|y_{ij}^*\| n_i^{\top}, \quad y_{ij}^{*\top} R_j = -\|y_{ij}^*\| n_j^{\top}$$

Since $y_{ij}^* = y_i^* - y_j^*$ points from body j toward body i , the minus sign in the definition of n_j gives the corresponding normal direction for body j . The stacked second derivative $\nabla_{\vartheta\vartheta}^2 B^*$ is block diagonal because B_i depends only on ϑ_i and B_j depends only on ϑ_j ; therefore, the mixed surface-parameter derivatives are identically zero:

$$\nabla_{\vartheta\vartheta}^2 B^* = \begin{bmatrix} \nabla_{\vartheta_i\vartheta_i}^2 B_i^* & 0 \\ 0 & \nabla_{\vartheta_j\vartheta_j}^2 B_j^* \end{bmatrix}$$

Therefore,

$$\left(I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R) \right) \nabla_{\vartheta\vartheta}^2 B^* = \begin{bmatrix} C_i & 0 \\ 0 & C_j \end{bmatrix}$$

$$C_i = [I_{n_s/2} \otimes (y_{ij}^{*\top} R_i)] \nabla_{\vartheta_i\vartheta_i}^2 B_i^*, \quad C_j = -[I_{n_s/2} \otimes (y_{ij}^{*\top} R_j)] \nabla_{\vartheta_j\vartheta_j}^2 B_j^*.$$

Using the relations derived above, for each fixed $k \in \{i, j\}$, denote the two components of ϑ_k by ϑ_1 and ϑ_2 . Then

$$C_k = \|y_{ij}^*\| \begin{bmatrix} n_k^\top \nabla_{\vartheta_1\vartheta_1}^2 B_k^* & n_k^\top \nabla_{\vartheta_1\vartheta_2}^2 B_k^* \\ n_k^\top \nabla_{\vartheta_2\vartheta_1}^2 B_k^* & n_k^\top \nabla_{\vartheta_2\vartheta_2}^2 B_k^* \end{bmatrix}, \quad k \in \{i, j\}$$

The matrix multiplying $\|y_{ij}^*\|$ is the second fundamental-form matrix of body k , expressed in its local surface parameters. Denote this matrix by

$$II_k = \begin{bmatrix} L_k & M_k \\ M_k & N_k \end{bmatrix}, \quad \begin{cases} L_k = n_k^\top \nabla_{\vartheta_1\vartheta_1}^2 B_k^* \\ M_k = n_k^\top \nabla_{\vartheta_1\vartheta_2}^2 B_k^* \\ N_k = n_k^\top \nabla_{\vartheta_2\vartheta_2}^2 B_k^* \end{cases}$$

Because the second fundamental form II_k is written in the local surface parameters, the active boundary patches are assumed to be regular at the closest points, so these parameters define valid tangent directions. Under the additional assumption of local strict convexity, the second fundamental form II_k is positive definite in those tangent directions [78]. Therefore, under the chosen normal convention, the curvature correction term

$$\left(I_{n_s} \otimes (y_{ij}^{*\top} I_{\pm} R) \right) \nabla_{\vartheta\vartheta}^2 B^*$$

is positive definite in the active surface-parameter directions. Since

$J_{\vartheta}J_{\vartheta}^{\top}$ is positive semi-definite and C is positive definite in the active surface-parameter directions, their sum is positive definite. Therefore, the stationarity matrix

$$A = J_{\vartheta}J_{\vartheta}^{\top} + C$$

is nonsingular at the closest-point solution. Hence, A^{-1} exists locally, and the sensitivity expressions in Eqs. (4.27)– (4.28) are well defined.

4.4 Analytical Distance Hessians

We now derive the Hessian of the minimum distance $d_{ij} = \|y_{ij}^*\|$ between two non-intersecting, smooth, parametric boundaries C_{b_i} and C_{b_j} of convex sets C_i and C_j , respectively. The derivation is restricted to closest-point solutions satisfying Lemma 4.4, so that the strict local convexity condition needed for the implicit derivative system holds.

4.4.1 Hessian of Distance Function

The gradients of d_{ij} with respect the configuration parameters q and φ , as shown in theorem 4.3 are

$$\nabla_q d_{ij} = \|y_{ij}^*\|^{-1} \nabla_q y_{ij}^{*\top} y_{ij}^*, \quad \nabla_{\varphi} d_{ij} = \|y_{ij}^*\|^{-1} \nabla_{\varphi} y_{ij}^{*\top} y_{ij}^*$$

Second Derivatives – General Structure

Differentiating $\nabla_q d_{ij}$ with respect to q and applying the product rule (2.41)

$$\begin{aligned} \nabla_{qq}^2 d_{ij} = & \left(\nabla_q \|y_{ij}^*\|^{-1} \right) y_{ij}^{*\top} (\nabla_q y_{ij}^{*\top})^{\top} \\ & + \|y_{ij}^*\|^{-1} \left[(\nabla_q y_{ij}^{*\top}) (\nabla_q y_{ij}^{*\top})^{\top} + (I_{n_q} \otimes y_{ij}^{*\top}) \nabla_{qq}^2 y_{ij}^* \right] \end{aligned} \quad (4.29)$$

The first term above accounts for the derivative of the normalization factor $\|y_{ij}^*\|^{-1}$,

while the second and the third terms capture the contribution from the Jacobian of y_{ij}^* and its second derivative

Derivative of the inverse of distance

To evaluate $\nabla_q \|y_{ij}^*\|^{-1}$, define

$$g(y_{ij}^*) = y_{ij}^{*\top} y_{ij}^*, \quad f(g) = g^{-0.5}$$

so that

$$\|y_{ij}^*\|^{-1} = (f \circ g \circ y_{ij}^*)(q, \varphi)$$

Using chain-rule for matrix differentials (2.57)

$$\nabla_q \|y_{ij}^*\|^{-1} = (\nabla_q y_{ij}^{*\top}) (\nabla_{y_{ij}^*} g^\top) \nabla_g f$$

Since $\nabla_g f(g) = -0.5 \|y_{ij}^*\|^{-3}$ and $\nabla_{y_{ij}^*} g^\top(y_{ij}^*) = 2y_{ij}^*$, we have

$$\nabla_q \|y_{ij}^*\|^{-1} = -\|y_{ij}^*\|^{-3} \nabla_q y_{ij}^{*\top} y_{ij}^* \quad (4.30)$$

Compact Hessian Expression

Substituting (4.30) in (4.29) and following the same derivation for φ gives:

$$\nabla_{qq}^2 d_{ij} = \|y_{ij}^*\|^{-1} \left[\nabla_q y_{ij}^{*\top} (I - \|y_{ij}^*\|^{-2} y_{ij}^* y_{ij}^{*\top}) (\nabla_q y_{ij}^{*\top})^\top + (I_{n_q} \otimes y_{ij}^{*\top}) \nabla_{qq}^2 y_{ij}^* \right] \quad (4.31)$$

$$\nabla_{q\varphi}^2 d_{ij} = \|y_{ij}^*\|^{-1} \left[\nabla_\varphi y_{ij}^{*\top} (I - \|y_{ij}^*\|^{-2} y_{ij}^* y_{ij}^{*\top}) (\nabla_q y_{ij}^{*\top})^\top + (I_{n_\varphi} \otimes y_{ij}^{*\top}) \nabla_{q\varphi}^2 y_{ij}^* \right] \quad (4.32)$$

$$\nabla_{\varphi\varphi}^2 d_{ij} = \|y_{ij}^*\|^{-1} \left[\nabla_\varphi y_{ij}^{*\top} (I - \|y_{ij}^*\|^{-2} y_{ij}^* y_{ij}^{*\top}) (\nabla_\varphi y_{ij}^{*\top})^\top + (I_{n_\varphi} \otimes y_{ij}^{*\top}) \nabla_{\varphi\varphi}^2 y_{ij}^* \right] \quad (4.33)$$

Geometric Interpretation of the Projection Matrix

In the Hessian expression, the matrix

$$P = I - \|y_{ij}^*\|^{-2} y_{ij}^* y_{ij}^{*\top}$$

plays a central role. Let

$$u = \|y_{ij}^*\|^{-1} y_{ij}^*$$

then the projection matrix can be written as

$$P = I - uu^\top$$

The matrix P is an orthogonal projection operator onto the subspace perpendicular to the closest-point separation direction u . For two disjoint convex bodies with smooth boundaries, the closest-point displacement is aligned with the outward surface normals at the two closest points [28]. Therefore, P removes the normal component of a perturbation and retains only the tangent-plane component. This term appears when differentiating the normalized closest-point vector in the distance Hessian.

Lemma 4.5 (Positive semi-definiteness of the tangent-plane projection)

The matrix

$$P = I - \|y_{ij}^*\|^{-2} y_{ij}^* y_{ij}^{*\top}$$

is positive semidefinite (PSD) with

$$\ker(P) = \text{span}\{u\}, \quad \text{rank}(P) = n - 1$$

Proof:

Let $u := \|y_{ij}^*\|^{-1} y_{ij}^*$. For any $z \in \mathbb{R}^n$,

$$z^\top P z = \|z\|^2 - (z^\top u)^2$$

Using Cauchy-Schwarz inequality,

$$|z^\top u| \leq \|z\| \|u\|$$

and $\|u\| = 1$, we obtain $z^\top P z \geq 0$

$$Pu = (I - uu^\top)u = u - u = 0$$

So $\text{span}\{u\} \subseteq \ker(P)$. Conversely, if $Pz = 0$, then $z = uu^\top z = (u^\top z)u$, so z is a scalar multiple of u . Hence $\ker(P) = \text{span}\{u\}$. By the rank-nullity theorem, $\text{rank}(P) = n - \dim \ker(P) = n - 1$. These properties are important in the present work because the projection matrix $P = I - uu^\top$ is positive semidefinite, which reveals the locally convex curvature structure of the distance function arising from the Euclidean norm. This structure leads to well-conditioned Hessian terms in the analytical derivatives used within the NMPC optimization problem, improving numerical stability and computational efficiency.

4.4.2 Second Derivatives of Closest-Point Vector

The Hessian of the distance function (4.31)- (4.33) involves second derivatives of the closest-point vector y_{ij}^* with respect to q and φ . This subsection derives the complete second-order sensitivity tensors in Eqs. (4.34)- (4.36). In addition to the assumptions of Lemma 4.4, their derivation assumes that the active boundary mapping $B(\vartheta)$ is locally C^3 at ϑ^* . This stronger regularity condition is required only for the complete second-order closest-point-vector sensitivities. Although Eqs. (4.31)- (4.33) contain second derivatives of y_{ij}^* , the terms in Eqs. (4.34)- (4.36) that require third derivatives of B^* make no contribution to the scalar distance Hessian when substituted into Eqns. (4.31)- (4.33), as shown in the remark following Eq. (4.36). Therefore, the exact scalar distance Hessian requires only C^2 boundary regularity. Using the first-derivative relation in (4.26), its transpose can be written as

$$(\nabla_\varphi \mathcal{Y}_{ij}^*)^\top = I_\pm \left[(\nabla_\varphi T^\top)^\top + R(\nabla_\varphi B^{*\top})^\top A^{-\top} W^\top + (\nabla_\varphi R^\top)^\top (I_{n_\varphi} \otimes B^*) \right]$$

Differentiate with respect to φ and use matrix product rule (2.41)

$$\begin{aligned} \nabla_{\varphi\varphi}^2 \mathcal{Y}_{ij}^* &= \left(I_{n_\varphi} \otimes I_\pm \right) \left(\nabla_{\varphi\varphi}^2 T + \nabla_\varphi R (\nabla_\varphi B^{*\top})^\top A^{-\top} W^\top \right) \\ &\quad + \left(I_{n_\varphi} \otimes I_\pm R \right) \nabla_\varphi (\nabla_\varphi B^{*\top})^\top A^{-\top} W^\top + \left(I_{n_\varphi} \otimes I_\pm R (\nabla_\varphi B^{*\top})^\top \right) \nabla_\varphi A^{-\top} W^\top \\ &\quad + \left(I_{n_\varphi} \otimes I_\pm R (\nabla_\varphi B^{*\top})^\top A^{-\top} \right) \nabla_\varphi W^\top + \left(I_{n_\varphi} \otimes I_\pm \right) \nabla_\varphi \left((\nabla_\varphi R^\top)^\top I_{n_\varphi} \otimes B^* \right) \end{aligned}$$

Using the identities:

(2.57),

$$\nabla_\varphi (\nabla_\varphi B^{*\top})^\top = (\nabla_\varphi \vartheta^\top \otimes I_{n_y}) \nabla_{\vartheta\vartheta}^2 B^*$$

(2.12),

$$\left(I_{n_\varphi} \otimes I_\pm R \right) \nabla_\varphi (\nabla_\varphi B^{*\top})^\top = (\nabla_\varphi \vartheta^\top \otimes I_\pm R) \nabla_{\vartheta\vartheta}^2 B^*$$

(2.64),

$$\nabla_\varphi \left((\nabla_\varphi R^\top)^\top (I_{n_\varphi} \otimes B^*) \right) = (\nabla_\varphi R^\top)^\top \odot \nabla_\varphi B^* + \nabla_{\varphi\varphi}^2 R (I_{n_\varphi} \otimes B^*)$$

(2.57),

$$\nabla_\varphi B^* = (\nabla_\varphi \vartheta^\top \otimes I_{n_y}) \nabla_\vartheta B^*$$

We obtain the compact form:

$$\begin{aligned} \nabla_{\varphi\varphi}^2 \mathcal{Y}_{ij}^* &= \left(I_{n_\varphi} \otimes I_\pm \right) \left[\nabla_{\varphi\varphi}^2 T + \nabla_\varphi R (\nabla_\varphi B^{*\top})^\top A^{-\top} W^\top \right] \\ &\quad + \left(\nabla_\varphi \vartheta^\top \otimes (I_\pm R) \right) \nabla_{\vartheta\vartheta}^2 B^* A^{-\top} W^\top \\ &\quad + \left(I_{n_\varphi} \otimes I_\pm R (\nabla_\varphi B^{*\top})^\top \right) \nabla_\varphi A^{-\top} W^\top \\ &\quad + \left(I_{n_\varphi} \otimes I_\pm R (\nabla_\varphi B^{*\top})^\top A^{-\top} \right) \nabla_\varphi W^\top \\ &\quad + \left(I_{n_\varphi} \otimes I_\pm \right) \left[(\nabla_\varphi R^\top)^\top \odot \left[(\nabla_\varphi \vartheta^\top \otimes I_{n_y}) \nabla_\vartheta B^* \right] + \nabla_{\varphi\varphi}^2 R (I_{n_\varphi} \otimes B^*) \right] \end{aligned} \tag{4.34}$$

Following the same steps starting from $\nabla_q y_{ij}^{*\top}$, we obtain:

$$\begin{aligned} \nabla_{qq}^2 y_{ij}^* &= \left(I_{n_q} \otimes I_{\pm} \right) \nabla_{qq}^2 T + \left(\nabla_q \vartheta^\top \otimes I_{\pm} R \right) \nabla_{\vartheta\vartheta}^2 B^* A^{-\top} V^\top \\ &\quad + \left[I_{n_q} \otimes I_{\pm} R \left(\nabla_{\vartheta} B^{*\top} \right)^\top \right] \nabla_q A^{-\top} V^\top + \left[I_{n_q} \otimes I_{\pm} R \left(\nabla_{\vartheta} B^{*\top} \right)^\top A^{-\top} \right] \nabla_q V^\top \end{aligned} \quad (4.35)$$

Similarly, differentiating $\nabla_q y_{ij}^{*\top}$ with respect to φ yields:

$$\begin{aligned} \nabla_{q\varphi}^2 y_{ij}^* &= \left(I_{n_\varphi} \otimes I_{\pm} \right) \left[\nabla_{q\varphi}^2 T + \nabla_\varphi R \left(\nabla_{\vartheta} B^{*\top} \right)^\top A^{-\top} V^\top \right] \\ &\quad + \left(\nabla_\varphi \vartheta^\top \otimes I_{\pm} R \right) \nabla_{\vartheta\vartheta}^2 B^* A^{-\top} V^\top + \left[I_{n_\varphi} \otimes I_{\pm} R \left(\nabla_{\vartheta} B^{*\top} \right)^\top \right] \nabla_\varphi A^{-\top} V^\top \\ &\quad + \left[I_{n_\varphi} \otimes I_{\pm} R \left(\nabla_{\vartheta} B^{*\top} \right)^\top A^{-\top} \right] \nabla_\varphi V^\top \end{aligned} \quad (4.36)$$

Remark:

The third and fourth terms of Eqns. (4.34)– (4.36) are required only when the complete second-order sensitivity tensors of y_{ij}^* are needed. From Eqns. (4.13) and (4.20),

$$y_{ij}^{*\top} I_{\pm} R \left(\nabla_{\vartheta} B^{*\top} \right)^\top = y_{ij}^{*\top} J_{\vartheta}^\top = \left(J_{\vartheta} y_{ij}^* \right)^\top = 0.$$

Consequently, when Eq. (4.35) is substituted into Eq. (4.31), its third and fourth terms make zero contribution. Likewise, when Eq. (4.36) is substituted into Eq. (4.32), its third and fourth terms make zero contribution, and when Eq. (4.34) is substituted into Eq. (4.33), its third and fourth terms make zero contribution.

The third terms are the only terms that require derivatives of $A^{-\top}$. Their evaluation requires $\nabla_q A^\top$ or $\nabla_\varphi A^\top$, which introduces $\nabla_{\vartheta\vartheta}^3 B^*$ through Eqns. (4.41) and (4.46). Since these terms make zero contribution to Eqns. (4.31)– (4.33), third derivatives of B^* are not required for the exact scalar distance Hessian. All remaining terms in Eqns. (4.34)– (4.36) are retained and require at most $\nabla_{\vartheta\vartheta}^2 B^*$.

4.4.3 Convexity Properties of Distance Hessians

Although the Euclidean distance is convex in the closest-point variables, the resulting distance function is not generally convex in the configuration variables. This is because the closest-point vector depends on the robot configuration through nonlinear rigid-body and articulated-kinematic mappings. Consequently, the Hessians derived in this section are not guaranteed to be positive semidefinite in general.

The Hessian contains two main effects. The first comes from changes in the direction of the closest-point vector. This effect is captured by the tangent-plane projection matrix in Lemma 4.5. The second comes from the movement of the closest points along the two body surfaces. Since these surfaces are curved and the bodies move through nonlinear kinematics, this second effect can make the Hessian indefinite.

Therefore, the analytical Hessians should be interpreted as exact local second-order sensitivity information rather than as globally convex approximations. This is sufficient for their use in Newton-type and interior-point optimization methods, where the Hessian is used to represent the local curvature of the nonlinear collision-avoidance constraints.

4.4.4 Derivatives of Auxiliary Matrices in the Distance Hessian

In this section, we compute the gradients of the matrix functions $A^\top$, $V^\top$ and $W^\top$ with respect to both the translation variables q and the rotation variables φ .

Gradients with Respect To q

The gradients with respect to q are computed as in the derivations from earlier sections, following the same chain-rule expansions and tensor reshaping steps already established. Applying these steps yields:

$$\nabla_q A^\top = (\nabla_q J_\vartheta) J_\vartheta^\top + (I_{n_q} \otimes J_\vartheta) (\nabla_q J_\vartheta^\top) + \nabla_q C^\top \quad (4.37)$$

$$\nabla_q V^\top = -(\nabla_q J_\vartheta) I_\pm (\nabla_q T^\top)^\top - [I_{n_q} \otimes (J_\vartheta I_\pm)] \nabla_{qq}^2 T \quad (4.38)$$

$$\begin{aligned} \nabla_q W^\top = & -(\nabla_q J_\vartheta) I_\pm (\nabla_\varphi T^\top)^\top - [I_{n_q} \otimes (J_\vartheta I_\pm)] \nabla_{\varphi q}^2 T \\ & - (J_\vartheta I_\pm (\nabla_\varphi R^\top)^\top) \odot [(\nabla_q \vartheta^\top \otimes I_{n_y}) \nabla_\vartheta B^*] \\ & - (\nabla_q J_\vartheta) I_\pm (\nabla_\varphi R^\top)^\top (I_{n_q} \otimes B^*) \\ & - [(\nabla_\vartheta B^{*T}) (\nabla_\varphi R)^\top (I_{n_\varphi} \otimes I_\pm^\top)] \odot \nabla_q y_{ij}^* \\ & - (\nabla_q \vartheta^\top \otimes I_{n_s}) \text{vec}_b(\nabla_{\vartheta\vartheta}^2 B^*)^\top (\nabla_\varphi R)^\top [I_{n_\varphi} \otimes (I_\pm^\top y_{ij}^*)] \end{aligned} \quad (4.39)$$

with

$$\nabla_q J_\vartheta = (\nabla_q \vartheta^\top \otimes I_{n_s}) \text{vec}_b(\nabla_{\vartheta\vartheta}^2 B^*)^\top R^\top I_\pm^\top \quad (4.40)$$

$$\nabla_q C^\top = (\nabla_{\vartheta\vartheta}^2 B^*)^\top \odot [(I_{n_q} \otimes (R^\top I_\pm^\top)) \nabla_q y_{ij}^*] + (\nabla_q \vartheta^\top \otimes I_{n_s}) \nabla_{\vartheta\vartheta\vartheta}^3 B^* (I_{n_s} \otimes R^\top I_\pm^\top y_{ij}^*) \quad (4.41)$$

Gradients with Respect To φ

The gradients with respect to φ are obtained by repeating the steps described in the q - derivative case, replacing ∇_q with throughout ∇_φ yields

$$\nabla_\varphi A^\top = (\nabla_\varphi J_\vartheta) J_\vartheta^\top + (I_{n_\varphi} \otimes J_\vartheta) (\nabla_\varphi J_\vartheta^\top) + \nabla_\varphi C^\top \quad (4.42)$$

$$\nabla_\varphi V^\top = -(\nabla_\varphi J_\vartheta) I_\pm (\nabla_q T^\top)^\top - [I_{n_\varphi} \otimes (J_\vartheta I_\pm)] \nabla_{q\varphi}^2 T \quad (4.43)$$

$$\begin{aligned}
\nabla_\varphi W^\top = & -(\nabla_\varphi J_\vartheta)I_\pm(\nabla_\varphi T^\top)^\top - [I_{n_\varphi} \otimes (J_\vartheta I_\pm)] \nabla_{\varphi\varphi}^2 T \\
& - [(\nabla_\vartheta B^{*\top} R^\top I_\pm^\top)I_\pm(\nabla_\varphi R^\top)^\top] \odot [(\nabla_\varphi \vartheta^\top \otimes I_{n_y}) \nabla_\vartheta B^*] \\
& - [(\nabla_\varphi J_\vartheta)I_\pm(\nabla_\varphi R^\top)^\top + (I_{n_\varphi} \otimes (J_\vartheta I_\pm)) \nabla_{\varphi\varphi}^2 R] (I_{\vartheta\varphi} \otimes B^*) \\
& - [(\nabla_\vartheta B^{*\top})(\nabla_\varphi R)^\top (I_{n_\varphi} \otimes I_\pm^\top)] \odot \nabla_\varphi y_{ij}^* \\
& - (\nabla_\varphi \vartheta^\top \otimes I_{n_s}) \text{vec}_b(\nabla_{\vartheta\vartheta}^2 B^*)^\top (\nabla_\varphi R)^\top [I_{n_\varphi} \otimes (I_\pm^\top y_{ij}^*)] \\
& - (I_{n_\varphi} \otimes \nabla_\vartheta B^{*\top}) (\nabla_{\varphi\varphi}^2 R)^\top [I_{n_\varphi} \otimes (I_\pm^\top y_{ij}^*)]
\end{aligned} \tag{4.44}$$

with

$$\nabla_\varphi J_\vartheta = (\nabla_\varphi \vartheta^\top \otimes I_{n_s}) \text{vec}_b(\nabla_{\vartheta\vartheta}^2 B^*)^\top R^\top I_\pm^\top + (I_{n_\varphi} \otimes \nabla_\vartheta B^{*\top}) \nabla_\varphi R^\top I_\pm^\top \tag{4.45}$$

$$\begin{aligned}
\nabla_\varphi C^\top = & (\nabla_{\vartheta\vartheta}^2 B^*)^\top \odot [(\nabla_\varphi R^\top)I_\pm^\top y_{ij}^* + (I_{n_\varphi} \otimes (R^\top I_\pm^\top)) \nabla_\varphi y_{ij}^*] \\
& + (\nabla_\varphi \vartheta^\top \otimes I_{n_s}) \nabla_{\vartheta\vartheta\vartheta}^3 B^* (I_{n_s} \otimes R^\top I_\pm^\top y_{ij}^*)
\end{aligned} \tag{4.46}$$

The expressions in this section are required only for evaluation of the complete second-order sensitivity tensors in Eqs. (4.34)– (4.36). In particular, $\nabla_q C^T$ in Eq. (4.41) and $\nabla_\varphi C^T$ in Eq. (4.46) contain $\nabla_{\vartheta\vartheta\vartheta}^3 B^*$, so their evaluation requires the active boundary mapping to be C^3 . These third-derivative terms are generally nonzero. However, as shown in the remark following Eq. (4.36), the terms containing $\nabla_q A^{-T}$ and $\nabla_\varphi A^{-T}$ make zero contribution to the scalar distance Hessian in Eqs. (4.31)– (4.33). Therefore, third derivatives of B^* are required only for the complete second-order closest-point-vector sensitivities, not for the exact scalar distance Hessian.

The matrix A may become ill-conditioned when the two surfaces are nearly parallel at the closest points. In such cases, a small regularization, such as $A + \epsilon I$, can be used to stabilize the implicit sensitivity calculation. This approximation is

especially reasonable for simple convex primitives, such as ellipses, ellipsoids, spherical patches, or planar patches, where curvature variation is limited or absent.

4.4.5 Higher-Order Surface-Boundary Derivatives

In this section, we derive expressions for the second and third derivatives of the boundary mapping B_k evaluated at the optimal surface parameters ϑ_k^* . The second derivative $\nabla_{\vartheta\vartheta}^2 B^*$ is required for the exact scalar distance Hessian, whereas the third derivative $\nabla_{\vartheta\vartheta\vartheta}^3 B^*$ is required only for the complete second-order sensitivity tensors in Eqs. (4.34)–(4.36). Recall that the boundary of each convex body C_k is described by a smooth parameterization

$$B_k: s_k \mapsto \mathbb{R}^{n_d}$$

where s_k denotes local surface parameters. In the distance formulation, the surface parameters are themselves functions of the optimization variables through the mapping

$$s_k = s_k(\vartheta_k)$$

Consequently, the boundary mapping appearing in the closest-point expressions is the composite mapping $B_k \circ s_k$. The higher-order derivatives of $B_k^* = B_k(s_k^*)$ with respect to the variables ϑ_k are therefore obtained by applying the chain rule to this composite mapping.

Determination of $\nabla_{\vartheta\vartheta}^2 B_k^*$

We begin with the chain rule (2.57) applied to composite mapping $B_k \circ s$ evaluated at ϑ_k^* :

$$\nabla_{\vartheta_k\vartheta_k}^2 B_k^* = \nabla_{\vartheta_k\vartheta_k}^2 (B_k \circ s)(\vartheta_k^*) = \nabla_{\vartheta_k} \left(\nabla_{\vartheta_k} (B_k^T \circ s)(\vartheta_k^*) \right)^\top$$

Using the Jacobian of B_k with respect to its surface parameters s_k

$$\nabla_{\vartheta_k \vartheta_k}^2 B_k^* = \nabla_{\vartheta_k} \left(\left(\nabla_{s_k} B_k^T(s_k^*) \right)^\top \left(\nabla_{\vartheta_k} s^T(\vartheta_k^*) \right)^\top \right)$$

Applying the product rule (2.41) yields:

$$\nabla_{\vartheta_k \vartheta_k}^2 B_k^* = (\nabla_{\vartheta_k} s_k^{*\top} \otimes I_{n_d}) (\nabla_{s_k s_k} B_k^*) (\nabla_{\vartheta_k} s_k^{*\top})^\top + (I_{n_s} \otimes (\nabla_{s_k} B_k^{*\top})^\top) \nabla_{\vartheta_k \vartheta_k} s_k^* \quad (4.47)$$

This expression shows that the second derivative of the boundary mapping consists of two contributions: the curvature of the surface parameterization B_k , and the second derivatives of the parameter transformation $s_k(\vartheta_k)$.

Determination of $\nabla_{\vartheta \vartheta}^3 B_k^*$

To obtain the third derivatives, take a transpose of (4.47) and differentiate both sides with respect to ϑ_k

$$\nabla_{\vartheta_k \vartheta_k \vartheta_k}^3 B_k^* = \nabla_{\vartheta_k} \left((\nabla_{\vartheta_k} s_k^{*\top}) (\nabla_{s_k s_k}^2 B_k^*)^\top ((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d}) \right) + \nabla_{\vartheta_k} (\nabla_{\vartheta_k \vartheta_k}^2 B_k^*)^\top$$

Expanding the first term of $\nabla_{\vartheta \vartheta}^3 B_k^*$ via the product rule (2.41)

$$\begin{aligned} & \nabla_{\vartheta_k} \left[(\nabla_{\vartheta_k} s_k^{*\top}) (\nabla_{s_k s_k}^2 B_k^*)^\top ((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d}) \right] \\ &= \left(\nabla_{\vartheta_k} (\nabla_{\vartheta_k} s_k^{*\top}) \right) (\nabla_{s_k s_k}^2 B_k^*)^\top \left[(\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right] \\ &+ (I_{n_s} \otimes \nabla_{\vartheta_k} s_k^{*\top}) \nabla_{\vartheta_k} (\nabla_{s_k s_k}^2 B_k^*)^\top \left[(\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right] \\ &+ \left[I_{n_s} \otimes \left((\nabla_{\vartheta_k} s_k^{*\top}) (\nabla_{s_k s_k}^2 B_k^*)^\top \right) \right] \nabla_{\vartheta_k} \left((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right) \end{aligned}$$

Using the identities

$$\nabla_{\vartheta_k} s_k^{*\top} = (\nabla_{\vartheta_k} s_k^{*\top})^\top$$

$$\nabla_{\vartheta_k} \left((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right) = \nabla_{\vartheta_k \vartheta_k}^2 s_k^{*\top} \otimes I_{n_d} \quad (\text{see 2.62})$$

$$(I_{n_s} \otimes \nabla_{\vartheta_k} s_k^{*\top}) (\nabla_{\vartheta_k} s_k^{*\top} \otimes I_{n_s n_y}) = \nabla_{\vartheta_k} s_k^{*\top} \otimes \nabla_{\vartheta_k} s_k^{*\top} \quad (\text{see 2.12})$$

$$\nabla_{s_k} (\nabla_{s_k s_k}^2 B_k^{*\top})^\top = \nabla_{s_k s_k s_k}^3 B_k^{*\top}$$

the first term simplifies to

$$\begin{aligned} & \nabla_{\vartheta_k} \left[\nabla_{\vartheta_k} s_k^{*\top} (\nabla_{s_k s_k}^2 B_k^*)^\top \left((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right) \right] \\ &= \nabla_{\vartheta_k \vartheta_k}^2 s_k^* (\nabla_{s_k s_k}^2 B_k^*)^\top \left((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right)^\top \\ &+ (\nabla_{\vartheta_k} s_k^{*\top} \otimes \nabla_{\vartheta_k} s_k^{*\top}) \nabla_{s_k s_k s_k}^3 B_k^{*\top} \left((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right) \\ &+ \left(I_{n_s} \otimes (\nabla_{\vartheta_k} s_k^{*\top}) (\nabla_{s_k s_k}^2 B_k^*)^\top \right) (\nabla_{\vartheta_k \vartheta_k}^2 s_k^* \otimes I_{n_d}) \end{aligned}$$

For the second term of $\nabla_{\vartheta \vartheta \vartheta}^3 B_k^*$, using identity (2.64)

$$\begin{aligned} & \nabla_{\vartheta_k} \left[\left(\nabla_{\vartheta_k \vartheta_k} s(s^*) \right)^\top \left(I_{n_s} \otimes \nabla_{s_k} B_k^\top(s^*) \right) \right] \\ &= (\nabla_{\vartheta_k \vartheta_k}^2 s_k^*)^\top \odot \nabla_{\vartheta_k} (\nabla_{s_k} B_k^{*\top}) + \nabla_{\vartheta_k} (\nabla_{\vartheta_k \vartheta_k}^2 s_k^{*\top})^\top (I_{n_s} \otimes \nabla_{s_k} B_k^{*\top}) \end{aligned}$$

and

$$\nabla_{\vartheta_k} (\nabla_{s_k} B_k^{*\top}) = (\nabla_{\vartheta_k} s_k^{*\top} \otimes I_{n_s}) \text{vec}_b (\nabla_{s_k s_k}^2 B_k^*)^\top$$

Combining all the terms, we obtain

$$\begin{aligned} \nabla_{\vartheta_k \vartheta_k \vartheta_k}^3 B_k^* &= (\nabla_{\vartheta_k \vartheta_k}^2 s_k^*) (\nabla_{s_k s_k}^2 B_k^*)^\top \left[(\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right]^\top \\ &+ (\nabla_{\vartheta_k} s_k^{*\top} \otimes \nabla_{\vartheta_k} s_k^{*\top}) \nabla_{s_k s_k s_k}^3 B_k^{*\top} \left((\nabla_{\vartheta_k} s_k^{*\top})^\top \otimes I_{n_d} \right) \\ &+ \left(I_{n_s} \otimes (\nabla_{\vartheta_k} s_k^{*\top}) (\nabla_{s_k s_k}^2 B_k^*)^\top \right) (\nabla_{\vartheta_k \vartheta_k}^2 s_k^* \otimes I_{n_d}) \\ &+ (\nabla_{\vartheta_k \vartheta_k}^2 s_k^*)^\top \odot \left[(\nabla_{\vartheta_k} s_k^{*\top} \otimes I_{n_s}) \text{vec}_b (\nabla_{s_k s_k}^2 B_k^*)^\top \right] \\ &+ \nabla_{\vartheta_k \vartheta_k \vartheta_k}^3 s_k^{*\top} (I_{n_s} \otimes \nabla_{s_k} B_k^{*\top}) \end{aligned} \tag{4.48}$$

4.5 Aggregated Distance Constraints

Let $d_{ij}(q, \varphi)$ denote the pairwise minimum distance between robot link i and

obstacle j , as defined in the previous sections. Enforcing all pairwise distance constraints is physically direct because each constraint represents the clearance between one link-obstacle pair. However, this approach can become computationally expensive when many links, obstacles, and prediction stages are present. Distance-based obstacle terms are widely used in optimization-based motion planning, including CHOMP [34], [79] and TrajOpt [80]. In the present work, aggregation is introduced to reduce the number of collision-avoidance constraints by replacing multiple pairwise link-obstacle distances within a group by a single link-to-group distance measure.

Let $\mathcal{O}_c$ denote the obstacle set assigned to group c , with $c = 1, \dots, N_c$. For link i , the corresponding link-to-group pair set is

$$P_{i,c} = \{(i, j) \mid j \in \mathcal{O}_c\}$$

Although the notation below is written using the obstacle set $\mathcal{O}_c$, it equivalently represents aggregation over the link-to-group pair set $P_{i,c}$. The grouping operation does not change the pairwise distance function d_{ij} . It only determines which pairwise distances enter each group. Obstacles may be grouped into geometrically meaningful sets, such as path corridors, workspace channels, or task-relevant regions, consistent with corridor- and region-based planning methods [81], [82].

For link i and group c , the physically direct link-to-group distance is the hard minimum over all pairwise distances in the group

$$d_{hard,i,c} = \min_{j \in \mathcal{O}_c} d_{ij} \tag{4.49}$$

This hard minimum identifies the closest obstacle in the group and therefore

gives the most direct clearance measure. However, the hard minimum is nonsmooth when the closest obstacle changes from one pair to another. This nonsmoothness is undesirable for the gradient-based optimization framework used in this dissertation. Therefore, the hard minimum is replaced by a smooth aggregate minimum distance.

The shifted log-sum-exp form used below is a smooth approximation of the hard minimum, consistent with the standard use of log-sum-exp as a smooth maximum/minimum approximation in optimization [77]. Thus, geometric relevance is handled by the grouping step, while differentiability is provided by the aggregate minimum distance. The corresponding link-to-group aggregate minimum distance is

$$d_{agr,i,c}(q, \varphi) = d_{hard,i,c} - \kappa \log \sum_{j \in \mathcal{O}_c} e^{-(d_{ij} - d_{hard,i,c})/\kappa} + \kappa \log |\mathcal{O}_c| \quad (4.50)$$

The shifted difference $d_{ij} - d_{hard,i,c}$ inside the exponential is used for numerical stabilization because the closest obstacle then contributes $e^0 = 1$, preventing the exponential sum from underflowing to zero. The added term $\kappa \log |\mathcal{O}_c|$ normalizes the aggregate distance so that, when all distances in the group are equal, the aggregate returns that common distance rather than an artificially reduced value. $|\mathcal{O}_c|$ denotes the number of obstacles in the obstacle group $\mathcal{O}_c$, and $\kappa > 0$ is the smoothing parameter used in the log-sum-exp approximation. A smaller value of κ makes the aggregate distance approach the hard minimum distance more closely, while a larger value of κ makes the aggregate distance smoother and allows more obstacles in the group to influence the aggregate value. Thus, κ is a numerical

smoothing parameter, not a physical clearance distance. If $|\mathcal{O}_c| = 1$, then $d_{\text{aggr},i,c} = d_{ij}$, so the formulation reduces to the pairwise distance constraint. As $\kappa \rightarrow 0^+$,

$$d_{\text{aggr},i,c}(q, \varphi; \kappa) \rightarrow d_{\text{hard},i,c}(q, \varphi)$$

The grouped clearance constraint is imposed directly on the aggregate distance as

$$d_{\text{aggr},i,c}(q, \varphi) \geq \delta \quad (4.51)$$

where $\delta > 0$ is the required minimum clearance. Groups with no assigned obstacles are omitted from the constraint set.

Using the normalized soft-min weights in (4.54), the derivative of the grouped aggregate distance is obtained as a weighted combination of the pairwise distance derivatives within the fixed obstacle group. In (4.52) and (4.53), x and y denote any of the configuration-variable blocks with respect to which derivatives are taken, i.e., $x, y \in \{q, \varphi\}$

Gradient of Grouped Aggregate Distance

$$\nabla_x d_{\text{aggr},i,c}(q, \varphi) = \sum_{j \in \mathcal{O}_c} [w_{ij}^{(c)} \nabla_x d_{ij}] \quad (4.52)$$

Mixed Hessian of Grouped Aggregate Distance

$$\begin{aligned} \nabla_{xy}^2 d_{\text{aggr},i,c}(q, \varphi) &= \sum_{j \in \mathcal{O}_c} \left[w_{ij}^{(c)} \left(\nabla_{xy}^2 d_{ij} - \frac{1}{\kappa} (\nabla_y d_{ij})(\nabla_x d_{ij})^\top \right) \right] \\ &\quad + \frac{1}{\kappa} (\nabla_y d_{\text{aggr},i,c})(\nabla_x d_{\text{aggr},i,c})^\top \end{aligned} \quad (4.53)$$

where

$$w_{ij}^{(c)} = \frac{e^{-(d_{ij}-d_{\text{hard},i,c})/\kappa}}{\sum_{l \in \mathcal{O}_c} e^{-(d_{il}-d_{\text{hard},i,c})/\kappa}}, \quad j \in \mathcal{O}_c \quad (4.54)$$

The derivatives above assume that the group membership O_c is fixed during a single derivative evaluation. In implementation, the collision-avoidance constraint is imposed directly as $d_{\text{aggr},i,c}(q, \varphi) \geq \delta$. The groups may be updated by an outer obstacle-selection procedure, such as assigning obstacles to path corridors or workspace regions. Thus, grouping determines which pairs are considered, while the shifted log-sum-exp expression provides a differentiable aggregate distance within each selected group. This grouped formulation provides a practical compromise between full pairwise constraints and a single global aggregate. Pairwise distance functions retain their physical meaning, while grouping reduces the number of competing obstacle pairs and prevents geometrically unrelated obstacles from influencing the same aggregate constraint.

4.6 Computational Evaluation of Distance Derivatives

The analytical results derived in the chapter can be assembled into a computational workflow for evaluating pairwise and link-to-group aggregated distance derivatives in articulated systems. Algorithm 4.1 summarizes the procedure and indicates the equations used at each step

Algorithm 4.1: Distance-Derivative Evaluation for Convex Body Pairs

Inputs

- Configuration variables q, φ
- Kinematic-chain description of each robot (3.23), (3.24), (3.25)
- Surface parameterizations $s_k \mapsto B_k(s_k)$, their inverse mappings $B_k^{-1}(y_k) \mapsto s_k$
- Derivatives of $B_k(s_k)$ up to second order; third-order derivatives only when

the complete second-order sensitivity tensors of y_{ij}^* are separately required

- Support functions of the convex sets defining the bodies
- Active pair set $P_{i,c} = \{(i,j): j \in \mathcal{O}_c\}$, $c = 1, \dots, N_c$
- Smoothing parameter $\kappa > 0$

Outputs

- Pairwise distances d_{ij} and their gradients and Hessians
- Aggregated distance function $d_{aggr,i,c}$ and its derivatives

Procedure

Step 1. Compute rigid-body kinematics and their derivatives

Using the forward-kinematics relations (3.24)– (3.25), evaluate for each link k

$$T_k(q, \varphi), \quad R_k(\varphi), \quad k \in \{i, j\}$$

Next compute their first- and second-order derivatives using the derivative propagation formulas from Chapter 3:

$$\nabla_q T_k, \nabla_\varphi T_k \quad (\text{Eq. (3.67), (3.68)})$$

$$\nabla_q R_k, \nabla_\varphi R_k \quad (\text{Eq. (3.69), (3.70)})$$

$$\nabla_{qq}^2 T_k, \nabla_{q\varphi}^2 T_k, \nabla_{\varphi\varphi}^2 T_k \quad (\text{Eq. (3.73)- (3.75)})$$

$$\nabla_{qq}^2 R_k, \nabla_{q\varphi}^2 R_k, \nabla_{\varphi\varphi}^2 R_k \quad (\text{Eq. (3.76)- (3.78)})$$

Step 2. Solve the closest-point problem

For each pair $(i, j) \in P_{i,c}$, compute the closest points by solving the minimum-distance problem (4.5):

$$(y_i^*, y_j^*) = \begin{cases} \min_{y_i, y_j} \|y_i - y_j\| \\ \text{s.t. } y_i \in C_{b_i}(q, \varphi), y_j \in C_{b_j}(q, \varphi) \end{cases}$$

The solution may be obtained using the GJK algorithm [28], which returns:

- the closest-point pair y_i^*, y_j^*
- the separation vector $y_{ij}^* = y_i^* - y_j^*$

Compute the pairwise distance

$$d_{ij} = \| y_{ij}^* \| \quad (\text{Eq. 4.15})$$

Next determine the corresponding surface parameters:

1. Recover (s_i^*, s_j^*) using the inverse surface mappings B_k^{-1} and Eq. (4.7),
2. Convert to the surface parameters ϑ_k^* using the transformation in Eq. (4.9).

Step 3. Compute pairwise distance gradients

Using the closest-point optimality conditions, construct the auxiliary matrices

$$A, \quad V, \quad W \quad (\text{Eqns. 4.20–4.24})$$

Then compute the derivatives of the closest-point vector

$$\nabla_q y_{ij}^*, \quad \nabla_\varphi y_{ij}^* \quad (\text{Eqns. 4.25–4.26})$$

Finally evaluate the pairwise distance gradients

$$\nabla_q d_{ij}, \quad \nabla_\varphi d_{ij} \quad (\text{Eqns. 4.17–4.18})$$

Step 4. Evaluate higher-order surface derivatives

At the optimal surface parameters ϑ_k^* , evaluate $\nabla_{\vartheta\vartheta}^2 B^*$ using Eq. (4.47). Evaluate $\nabla_{\vartheta\vartheta\vartheta}^3 B^*$ using Eq. (4.48) only when the complete second-order sensitivity tensors

$$\nabla_{qq}^2 y_{ij}^*, \quad \nabla_{q\varphi}^2 y_{ij}^*, \quad \nabla_{\varphi\varphi}^2 y_{ij}^*$$

are separately required.

Step 5. Compute closest-point and auxiliary derivatives

Using the matrices A , V , and W constructed in Step 3, compute using (4.27)–(4.28):

$$\nabla_q \vartheta^{*T} = VA^{-1}, \quad \nabla_\phi \vartheta^{*T} = WA^{-1}$$

If the complete second-order sensitivity tensors

$$\nabla_{qq}^2 \mathcal{Y}_{ij}^*, \quad \nabla_{q\phi}^2 \mathcal{Y}_{ij}^*, \quad \nabla_{\phi\phi}^2 \mathcal{Y}_{ij}^*$$

are separately required, compute using Eqns. (4.37)– (4.46):

$$\nabla_q A^T, \quad \nabla_\phi A^T, \quad \nabla_q V^T, \quad \nabla_\phi V^T, \quad \nabla_q W^T, \quad \nabla_\phi W^T$$

Step 6. Compute pairwise distance Hessians

- Evaluate the second derivatives of the closest-point vector

$\nabla_{qq}^2 \mathcal{Y}_{ij}^*, \nabla_{q\phi}^2 \mathcal{Y}_{ij}^*, \nabla_{\phi\phi}^2 \mathcal{Y}_{ij}^*$ using Eqns. (4.34)– (4.36) only when the complete second-order sensitivity tensors are required.

- For evaluation of the pairwise distance Hessians, retain only the terms that make nonzero contributions according to the remark following Eq. (4.36).

Then compute $\nabla_{qq}^2 d_{ij}, \nabla_{q\phi}^2 d_{ij}, \nabla_{\phi\phi}^2 d_{ij}$ using Eqns. (4.31)– (4.33).

Step 7. Compute link-to-group aggregated distance derivatives (optional)

If distance aggregation is used, then for each link i and obstacle group c :

1. Compute the aggregated distance $d_{aggr,i,c}$ using Eq. (4.50)
2. Compute the aggregated gradient $\nabla_x d_{aggr,i,c}, x \in \{q, \phi\}$ using Eq. (4.52)
3. Compute the aggregated Hessian $\nabla_{xy}^2 d_{aggr,i,c}, x, y \in \{q, \phi\}$ using Eq. (4.53)

When a stacked gradient or Hessian representation is required, such as for the NMPC formulations developed in Chapter 6, the derivative expressions obtained from Algorithm 4.1 may be converted to the corresponding stacked forms using the gradient and Hessian stacking conventions introduced in Chapter 2.

4.7 Numerical Results

This section evaluates the accuracy and computational efficiency of the analytical minimum-distance derivative formulation. Three studies are presented. The first validates the analytical gradient and Hessian for a tractor-trailer system in a cluttered two-dimensional environment and compares their runtime against finite-difference approximations. The second provides a preliminary assessment of global log-sum-exp aggregation of multiple pairwise distances for the same setup. The third validates the analytical derivatives for a spatial three-link robot with ellipsoidal obstacles.

4.7.1 Tractor-Trailer System in a 2D Cluttered Environment

Simulation Setup

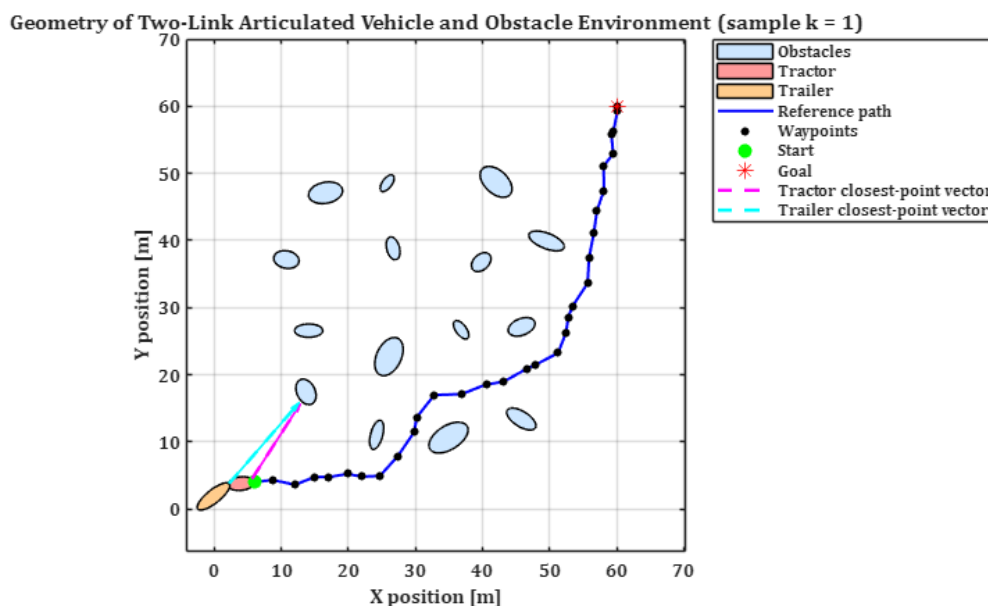


Figure 4.2: 2D Tractor-Trailer Setup for Distance-Derivative Validation

A two-link articulated vehicle was simulated in a cluttered two-dimensional

workspace, as shown in Fig. 4.2. The tractor, trailer, and static obstacles were modeled as elliptic convex sets. This case provides a controlled benchmark for validating the analytical gradient and Hessian of the minimum link–obstacle distance along a representative trajectory.

Gradient and Hessian Verification

The analytical gradient and Hessian of the pairwise link–obstacle distance were compared against central finite-difference approximations along the trajectory. For a scalar function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ evaluated at $x \in \mathbb{R}^n$, the finite-difference approximations are [36]:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_i + h_i e_i) - f(x_i - h_i e_i)}{2h_i} \quad (4.55)$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \approx \frac{\left(f(x_i + h_i e_i + h_j e_j) - f(x_i + h_i e_i - h_j e_j) - f(x_i - h_i e_i + h_j e_j) + f(x_i - h_i e_i - h_j e_j) \right)}{4h_i h_j} \quad (4.56)$$

where $h_i, h_j > 0$ are the finite-difference step sizes, and e_i denotes the i -th Euclidean basis vector. A commonly used step-size estimate for first-order central finite differences is [36]:

$$h_i = \sqrt{\varepsilon}(1 + |x_i|)$$

where $\varepsilon \approx 10^{-16}$ is machine epsilon in double precision arithmetic.

Fig. 4.3 shows the derivative verification errors as a function of finite-difference step size. For both links, the gradient and Hessian errors increase as the perturbation size is increased. This trend is expected because larger perturbations evaluate the distance function farther from the nominal configuration, where higher-

order nonlinear effects become more significant. The small errors at lower perturbation sizes indicate good agreement between the analytical derivatives and the finite-difference approximations near the nominal trajectory samples.

Computational Efficiency of Analytical Derivatives

Table 4.1 compares the computational cost of analytical and Finite-Difference (FD) derivative evaluations. The gradient and Hessian errors are measured using

$$\|\nabla d^a - \nabla d^n\|_\infty \quad (4.57)$$

$$\|\nabla^2 d^a - \nabla^2 d^n\|_\infty \quad (4.58)$$

where a and n denote analytical and FD evaluation respectively.

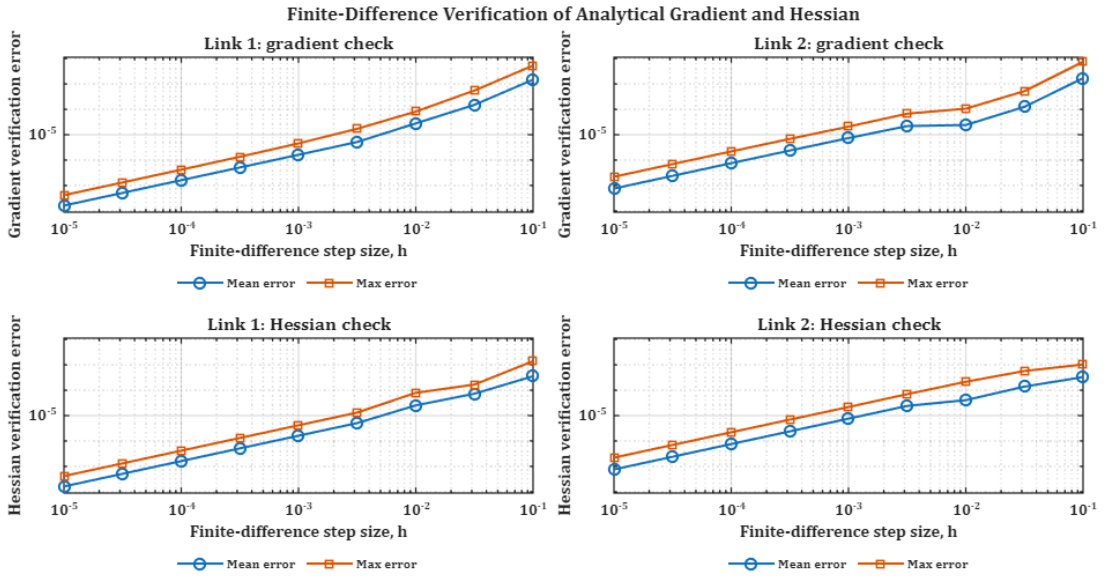


Figure 4.3: 2D Tractor-Trailer Distance-Derivative Verification

The analytical formulation provides approximately $57 \times$ speedup for gradient evaluations and more than $118 \times$ speedup for Hessian evaluations. This improvement occurs because finite differences require repeated distance evaluations for each derivative component, whereas the analytical expressions compute the derivatives directly.

Quantity	Error metric	Error	FD avg. time per link-obstacle pair per waypoint (s)	Analytical avg. time per link-obstacle pair per waypoint (s)	Speed up
Gradient	(4.57)	0.0013	0.034	0.0006	56.6
Hessian	(4.58)	0.0409	0.153	0.0013	117.7

Table 4.1: Comparison of analytical vs. FD Pairwise Distance Gradient and Hessian computations

Runtime Scaling with Number of Pairwise Distance Terms

Fig. 4.4 shows runtime scaling as the number of pairwise link–obstacle distance terms increases. The finite-difference runtime grows rapidly with the number of pairwise terms, while the analytical formulation remains significantly faster. This scaling advantage is important for cluttered environments and articulated systems, where the number of pairwise collision terms

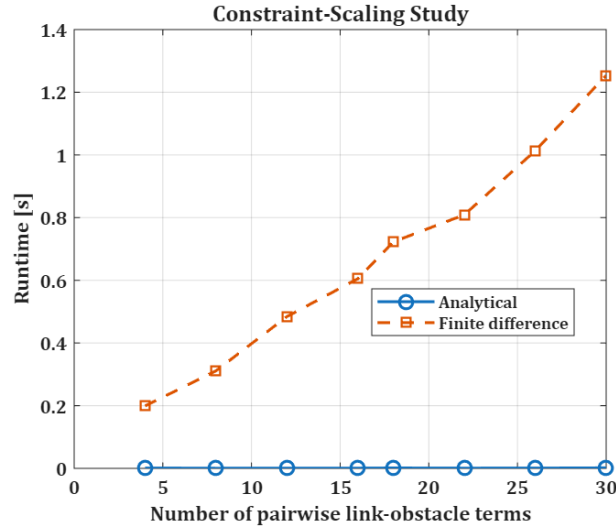


Figure 4.4: Runtime Scaling with Pairwise Distance Terms

4.7.2 Preliminary Study of Soft-Minimum Aggregation

A preliminary study was also performed to examine global log-sum-exp aggregation of multiple pairwise link–obstacle distances. This aggregation replaces

the nonsmooth hard minimum with a differentiable approximation. However, because all pairwise distances are combined into a single global function, the resulting derivative may blend obstacle interactions associated with different avoidance directions. This is especially problematic when obstacles on opposite sides of the path have comparable distances, since their gradients may point in competing lateral directions. Fig. 4.5 shows the approximation error between the hard-minimum distance and the shifted soft-minimum approximation as a function.

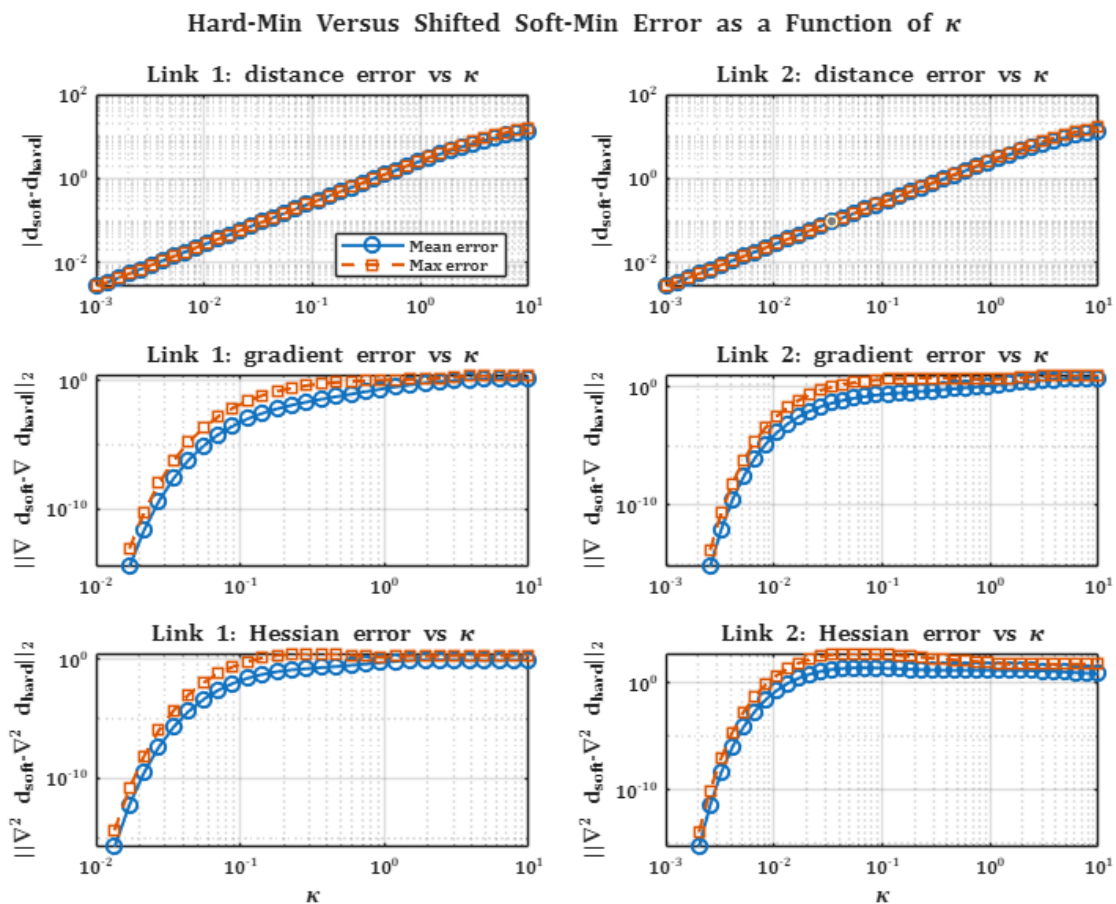


Figure 4.5: Soft-Minimum Aggregation Error

For both links, the distance error increases as κ increases, indicating that

smaller κ values provide a closer approximation to the hard minimum. The gradient and Hessian errors also increase rapidly from very small κ values and then tend to level off for larger κ , showing that the derivative approximation becomes less consistent with the hard-minimum derivative as the smoothing becomes stronger. This illustrates the basic tradeoff in the shifted soft-minimum formulation: small κ better preserves the hard-minimum distance and derivative behavior, whereas larger κ introduces stronger smoothing at the cost of larger approximation error.

Fig. 4.6 compares the hard-minimum distance and global soft-minimum distance along the trajectory for several smoothing parameters. The results show the expected tradeoff between approximation accuracy and smoothness.

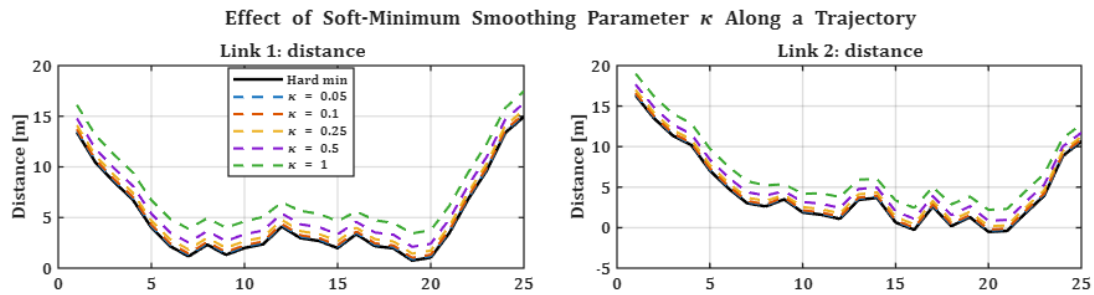


Figure 4.6 Hard-Minimum and Global Soft-Minimum Distance Along the Trajectory

These results indicate that global aggregation is useful as a diagnostic smoothing study, but it is not adopted as the final collision-avoidance formulation. In cluttered environments, grouped aggregation based on spatial region, side of the path, active interaction set, or local avoidance direction would likely provide better locality and numerical conditioning.

4.7.3 3D Validation of Analytical Distance Derivatives

A three-dimensional validation case was also performed using a mobile

three-link robot and three ellipsoidal obstacles, as shown in Fig. 4.7. Each robot link and obstacle was modeled as an oriented ellipsoid. The configuration vector included the base position, orientation, and two revolute joint angles, resulting in an eight-dimensional configuration vector. For each waypoint, the minimum distance between each robot link and obstacle was evaluated. The analytical gradients and Hessians were compared against FD approximations only on smooth exterior samples. Samples near contact or closest-point transitions were excluded because FD derivatives are unreliable at nonsmooth minimum-distance points. Out of 270 link-obstacle waypoint evaluations, 119 samples satisfied the verification checks.

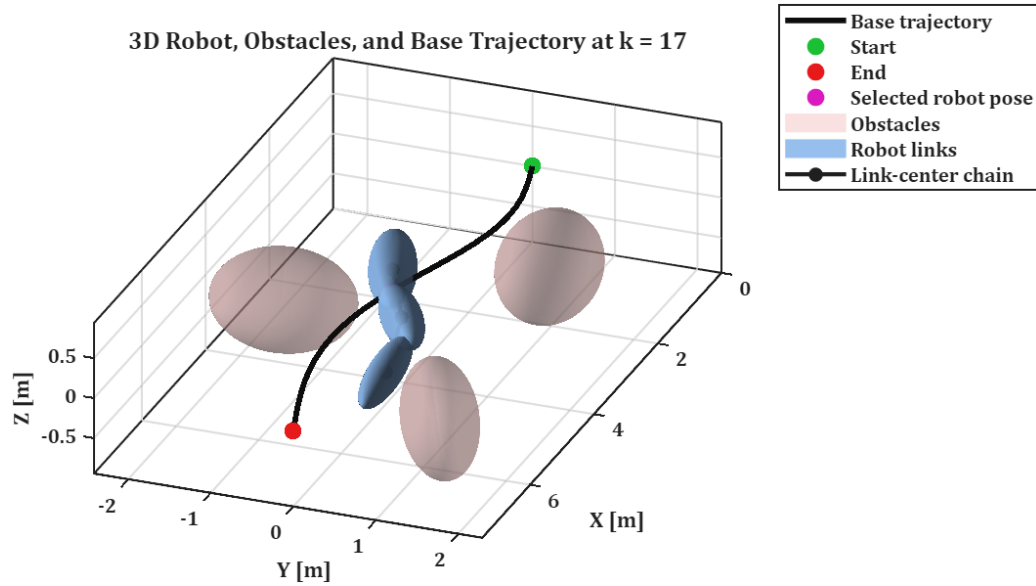


Figure 4.7: 3D Distance-Derivative Validation Scene.

Tables 4.2 and 4.3 show that the analytical gradients agree closely with finite differences, while the Hessians show acceptable agreement on smooth exterior samples. The larger Hessian errors are expected because second-order finite-difference verification is more sensitive to perturbation size and closest-point

switching. The analytical method also provides substantial computational savings, with approximately $111\times$ speedup for gradient and $577\times$ speedup for Hessian evaluations.

Quantity	Maximum error	Mean error	Median error
$\nabla_q d$	2.4738×10^{-3}	5.0958×10^{-4}	3.3977×10^{-4}
$\nabla_\varphi d$	5.7113×10^{-3}	9.8248×10^{-4}	6.8274×10^{-4}
$\nabla_{qq}^2 d$	9.4369×10^{-2}	2.4110×10^{-3}	4.0823×10^{-4}
$\nabla_{q\varphi}^2 d$	2.5942×10^{-1}	5.2889×10^{-3}	5.2277×10^{-4}
$\nabla_{\varphi\varphi}^2 d$	4.9834×10^{-1}	3.9999×10^{-2}	1.9666×10^{-2}

Table 4.2: Error summary for analytical vs FD derivatives in the 3D, 3-link robot case

Quantity	Error metric	Error	FD avg. time per link-obstacle pair per waypoint (s)	Analytical avg. time per link-obstacle pair per waypoint (s)	Speed up
Gradient	(4.57)	0.0057	0.0637	0.000573	111.2
Hessian	(4.58)	0.4983	1.0112	0.001753	576.8

Table 4.3: Comparison of analytical and FD derivative computation times for the 3D, 3-link robot case

4.8 Chapter Summary

The numerical results validate the analytical gradient and Hessian of the pairwise minimum-distance function for the tractor-trailer system in a cluttered two-dimensional environment. The analytical derivatives agree with finite-difference approximations while providing approximately $57 \times$ gradient speedup and more than $118 \times$ Hessian speedup, with increasing advantage as the number of pairwise link-obstacle terms grows. The preliminary aggregation study shows that a global log-sum-exp soft minimum can approximate the hard minimum but may blend obstacle interactions with different avoidance directions, motivating grouped

link-to-group aggregation in which only geometrically related obstacle pairs are combined. Thus, the main validated contribution is the pairwise analytical derivative framework, while grouped aggregation provides a structured extension for reducing constraint complexity. All experiments were conducted in MATLAB R2025b on a Windows 11 Pro workstation with an Intel Core i9 processor and 48 GB RAM; reported times correspond to interpreted MATLAB execution and represent unoptimized runtimes.

CHAPTER 5 - SMOOTH PATH GENERATION FROM WAYPOINTS

5.1 Introduction

In autonomous vehicle guidance and robotic motion planning problems, a desired route is often specified as a discrete sequence of waypoints obtained from a higher-level planner, digital map, or perception systems [1]. These waypoints represent ordered spatial locations that define the geometric structure of a desired path through the environment. Formally, the set of waypoints can be expressed as

$$w_i \in \mathbb{R}^d, \quad i = 0, \dots, m, \quad d \in \{2, 3\} \quad (5.1)$$

where d denotes the spatial dimension of the environment. In most vehicle navigation applications, $d = 2$ corresponds to planar motion on a road surface, while $d = 3$ may arise in aerial or robotic manipulation problems.

While waypoints define the overall geometry of a route, they do not provide the smooth representation required by control algorithms. Directly following a polyline formed by connecting waypoints leads to discontinuities in heading and curvature at waypoint locations, which can generate non-physical control commands and lead to unstable or infeasible trajectories. In this dissertation, this smoothness requirement is imposed by the MPPFC formulation introduced in Chapter 3, where the reference path is represented as a parametric function $r_p(\theta)$, and its derivatives with respect to θ are required for path projection, curvature evaluation, and collision avoidance constraints. Consequently, the discrete waypoint representation produced by higher-level planners must be converted into a smooth parametric path representation with the differentiability required by the controller.

Polynomial splines provide a convenient framework for constructing such paths because they offer closed-form analytical expressions for derivatives while allowing local control of the curve geometry [84]– [86]. In general, a polynomial spline is defined as a piecewise polynomial function over a partition of a scalar parameter domain. Without loss of generality, let the path parameter be denoted by s , defined over the interval

$$0 = s_0 < s_1 < \dots < s_{m-1} < s_m = l_s$$

where l_s denotes the total length of the path parameter domain. A spline curve $C(s)$ is defined piecewise as

$$C(s) = \begin{cases} C_1(s), & s_0 \leq s < s_1 \\ C_2(s), & s_1 \leq s < s_2 \\ \vdots & \vdots \\ C_m(s), & s_{m-1} \leq s \leq s_m \end{cases} \quad (5.2)$$

where each segment $C_i(s)$ is a polynomial of degree n ,

$$C_i(s) = \sum_{j=0}^n c_{j,i} \tau_i^j(s) \quad (5.3)$$

with coefficients $c_{j,i} \in \mathbb{R}^d$ determining the shape of the curve. The local parameter $\tau_i \in [0,1]$ is defined as

$$\tau_i = \frac{s - s_{i-1}}{l_i}, \quad l_i = s_i - s_{i-1} \quad (5.4)$$

where l_i denotes the parameter length of segment i . This normalization allows each polynomial segment to be expressed on a common unit interval regardless of the physical length of the segment.

Among polynomial spline representations, composite Bézier curves are

particularly convenient for path planning and motion control because their derivatives admit compact analytical expressions and their geometric properties are well understood. Bézier curves are constructed from Bernstein polynomial bases [83], and they form the foundation of many geometric modeling techniques used in computer-aided design and robotics [84].

For path-following control problems, it is advantageous to parameterize the curve using a scalar variable s that is closely related to the arc length. Let l_i denote the arc length of segment i . The total path length is given by

$$l_s = \sum_{i=1}^m l_i, \quad l_i := \int_{s_{i-1}}^{s_i} \left\| \frac{dC_i(s)}{ds} \right\| ds \quad (5.5)$$

When a spline is parameterized by arc length, the parameter s corresponds directly to distance travelled along the path. This simplifies the evaluation of tangents, curvature, and related geometric quantities used in path-following control and collision avoidance constraints.

In this dissertation, smooth reference paths are constructed using composite polynomial Bézier curves because they provide closed-form analytical expressions for position and derivatives, straightforward enforcement of geometric continuity conditions, local geometric control through a small set of control points, and efficient evaluation suitable for real-time control algorithms. These properties make composite Bézier curves well suited for the MPPFC framework developed in this work, where accurate evaluation of path position, tangent, curvature, and higher-order derivatives is required at each optimization step.

5.2 Bézier Curves and Composite Bézier Paths

A Bézier curve is a parametric polynomial curve that is defined using a set of control points

$$\{P_0, P_1, \dots, P_n\}, \quad P_i \in \mathbb{R}^d$$

where n denotes the degree of the curve and d is the dimension of the ambient space. A Bézier curve can be defined recursively using de Casteljau's algorithm or explicitly using Bernstein polynomials [83], [84]. The Bernstein form is used throughout this dissertation because it provides compact analytical expressions for the curve and its derivatives.

An n – th degree Bézier curve parameterized by $s \in [0,1]$, can be expressed in Bernstein form as [83]:

$$C(s): [0,1] \rightarrow \mathbb{R}^d, \quad C(s) = \sum_{j=0}^n [P_j b_{j,n}(s)] \quad (5.6)$$

The Bernstein basis polynomial $b_{j,n}(s)$ is defined as

$$b_{j,n}(s) = c(j, n) s^j (1 - s)^{n-j}, \quad c(j, n) = \binom{n}{j} = \frac{n!}{j! (n - j)!} \quad (5.7)$$

Using the binomial theorem, it can be shown that the Bernstein polynomials form a partition of unity [84]

$$\sum_{j=0}^n b_{j,n}(s) = 1 \quad (5.8)$$

which implies that a Bézier curve lies within the convex hull of its control points.

5.2.1 Parameter-Domain Transformation

Bézier curves are invariant under affine transformations of the parameter

domain [84]. As a result, the definition in (5.6) can be generalized to any interval

$$s \in [\underline{s}, \bar{s}]$$

by defining

$$\Delta s = \bar{s} - \underline{s} > 0 \quad (5.9)$$

$$b_{j,n}(s) = \frac{c(j,n)}{\Delta s^n} (s - \underline{s})^j (\bar{s} - s)^{n-j} \quad (5.10)$$

which reduces to standard Bernstein form when $\underline{s} = 0$ and $\bar{s} = 1$.

5.2.2 Composite Bézier Curve

In practice, complex paths cannot usually be represented accurately using a single polynomial segment. Instead, multiple Bézier segments are connected to form a composite Bézier curve. Let the i – th Bézier segment be defined over the interval $s \in [\bar{s}_{i-1}, \bar{s}_i]$ as

$$C_i(s) = \sum_{j=0}^n [P_j^i b_{j,n}^i(s)] \quad (5.11)$$

where P_j^i is the j – th control point of the i – th Bézier curve and $b_{j,n}^i(s)$ is the Bernstein basis over the interval $[\bar{s}_{i-1}, \bar{s}_i]$, with

$$\Delta s_i = \bar{s}_i - \bar{s}_{i-1} \quad (5.12)$$

Using m such segments, a composite Bézier curve is defined as

$$C(s) = \bigcup_{i=1}^m C_i(s) \quad (5.13)$$

where $\bar{s}_0$ and $\bar{s}_m$ are the boundary points and $\{\bar{s}_1, \dots, \bar{s}_{m-1}\}$ are referred to as knot points. The composite Bézier curve can be re-written as

$$C(s) = \sum_{i=0}^m [\alpha_i C_i(s)] \quad (5.14)$$

where

- α_i activates the segment, whose domain contains s ,
- $P^i = [P_0^i \ \dots \ P_n^i]$ is the matrix of control points for segment i ,
- $B_n^i(s) = [b_{0,n}^i(s) \ \dots \ b_{n,n}^i(s)]^T$ is a vector of Bernstein basis polynomials for segment i .

5.2.3 First Derivative of a Composite Bézier Curve

Taking the $i - th$ segment defined in (5.11) with respect to s

$$\frac{dC_i(s)}{ds} = \sum_{j=0}^n \left[P_j^i \frac{db_{j,n}^i(s)}{ds} \right]$$

Differentiating the Bernstein basis yields the identity [84]

$$\frac{db_{j,n}^i(s)}{ds} = \frac{n}{\Delta s_i} (b_{j-1,n-1}^i(s) - b_{j,n-1}^i(s)) \quad (5.15)$$

Substituting this result and rearranging terms gives

$$\frac{dC_i(s)}{ds} = \frac{n}{\Delta s_i} \sum_{j=0}^{n-1} [(P_{j+1}^i - P_j^i) b_{j,n-1}^i(s)] \quad (5.16)$$

Define the forward differences

$$\delta P_j^i = P_{j+1}^i - P_j^i \quad (5.17)$$

Then

$$\frac{dC_i(s)}{ds} = \frac{n}{\Delta s_i} \sum_{j=0}^{n-1} [\delta P_j^i b_{j,n-1}^i(s)] \quad (5.18)$$

which shows that the derivative of a Bézier curve is itself a Bézier curve of degree $n - 1$. The derivative of the composite curve follows directly

$$\frac{dC(s)}{ds} = \sum_{i=0}^m \left[\alpha_i \frac{dC_i(s)}{ds} \right] \quad (5.19)$$

5.2.4 Second Derivative of a Composite Bézier Curve

Differentiating (5.16) gives

$$\frac{d^2C_i(s)}{ds^2} = \frac{n(n-1)}{(\Delta s_i)^2} \sum_{j=0}^{n-2} [(P_{j+2}^i - 2P_{j+1}^i + P_j^i) b_{j,n-2}^i(s)] \quad (5.20)$$

Define the second forward differences

$$\delta^2 P_j^i = P_{j+2}^i - 2P_{j+1}^i + P_j^i \quad (5.21)$$

Then

$$\frac{d^2C_i(s)}{ds^2} = \frac{n(n-1)}{(\Delta s_i)^2} \sum_{j=0}^{n-2} [\delta^2 P_j^i b_{j,n-2}^i(s)] \quad (5.22)$$

and the second derivative of the composite curve becomes

$$\frac{d^2C(s)}{ds^2} = \sum_{i=0}^m \left[\alpha_i \frac{d^2C_i(s)}{ds^2} \right] \quad (5.23)$$

5.2.5 C^2 Continuity of Composite Bézier Curve

A parametric curve is C^2 continuous if the curve and its first two derivatives with respect to the parameter are continuous. Since each Bézier segment is smooth in its interior, the continuity conditions only need to be enforced at the knot points

$$\begin{aligned}
C_i(\bar{s}_i) &= C_{i+1}(\bar{s}_i) \\
\frac{dC_i(\bar{s}_i)}{ds} &= \frac{dC_{i+1}(\bar{s}_i)}{ds} \quad \forall 0 \leq i \leq m-1 \\
\frac{d^2C_i(\bar{s}_i)}{ds^2} &= \frac{d^2C_{i+1}(\bar{s}_i)}{ds^2}
\end{aligned} \tag{5.24}$$

These conditions can be expressed directly in terms of the Bézier control points. From the segment definition in (5.11),

$$C_i(s) = \sum_{j=0}^n [P_j^i b_{j,n}^i(s)]$$

evaluating the adjacent segments at the shared knot $\bar{s}_i$ gives

$$C_i(\bar{s}_i) = P_n^i, \quad C_{i+1}(\bar{s}_i) = P_0^{i+1}$$

Therefore, the position continuity condition becomes

$$P_n^i = P_0^{i+1} \tag{5.25}$$

which ensures that the two segments meet at the same point.

Evaluating the first derivative expressions (5.16) at the end points

$$\frac{dC_i(\bar{s}_i)}{ds} = \frac{n}{\Delta s_i} (P_n^i - P_{n-1}^i), \quad \frac{dC_{i+1}(\bar{s}_i)}{ds} = \frac{n}{\Delta s_{i+1}} (P_1^{i+1} - P_0^{i+1})$$

Enforcing equality of these derivatives yields

$$P_n^i - P_{n-1}^i = \frac{\Delta s_i}{\Delta s_{i+1}} (P_1^{i+1} - P_0^{i+1}) \tag{5.26}$$

which guarantees continuity of the tangent vector.

Evaluating the second derivative expression (5.20) at the end points

$$\begin{aligned}
\frac{d^2C_i(\bar{s}_i)}{ds^2} &= \frac{n(n-1)}{(\Delta s_i)^2} (P_n^i - 2P_{n-1}^i + P_{n-2}^i), \\
\frac{d^2C_{i+1}(\bar{s}_i)}{ds^2} &= \frac{n(n-1)}{(\Delta s_{i+1})^2} (P_2^{i+1} - 2P_1^{i+1} + P_0^{i+1})
\end{aligned}$$

Hence,

$$P_n^i - 2P_{n-1}^i + P_{n-2}^i = \left(\frac{\Delta s_i}{\Delta s_{i+1}} \right)^2 (P_2^{i+1} - 2P_1^{i+1} + P_0^{i+1}) \quad (5.27)$$

5.2.6 G^2 Continuity of Composite Bézier Curves

A curve is geometrically continuous of order two, i.e., G^2 continuous, if the curve, its unit tangent vector, and its curvature vector are continuous [85]. Unlike C^2 continuity, G^2 continuity enforces geometric smoothness without requiring equality of derivatives with respect to the original parameter. This distinction is important because G^2 continuity is sufficient for geometric boundary smoothing, while C^2 continuity is required when parametric derivatives appear directly in the motion-control formulation [84], [85]. Therefore, this dissertation uses C^2 continuity for reference-path smoothing in the MPPFC formulation, while the G^2 conditions are included for completeness and possible future use in smoothing polygonal shapes, obstacle boundaries, or corridor boundaries. The unit tangent vector is defined as

$$T(s) = \frac{dC(s)/ds}{\|dC(s)/ds\|}$$

Requiring equality of the unit tangents at the knot point gives

$$\frac{dC_i(\bar{s}_i)/ds}{\|dC_i(\bar{s}_i)/ds\|} = \frac{dC_{i+1}(\bar{s}_i)/ds}{\|dC_{i+1}(\bar{s}_i)/ds\|}$$

Substituting the derivative expressions leads to

$$P_n^i - P_{n-1}^i = \alpha_i (P_1^{i+1} - P_0^{i+1}) \quad (5.28)$$

$$\alpha_i = \frac{\|(P_n^i - P_{n-1}^i)\|}{\|(P_1^{i+1} - P_0^{i+1})\|} \quad (5.29)$$

which represents the ratio of the tangent magnitudes

The curvature vector of a parametric curve $C(s)$ can be expressed as [85]

$$K(s) = \frac{\frac{dC(s)}{ds} \times \frac{d^2C(s)}{ds^2} \times \frac{dC(s)}{ds}}{\|dC(s)/ds\|^4} = \kappa(s)N(s) \quad (5.30)$$

where

- $\kappa(s)$ is the magnitude of curvature,
- $N(s)$ is the unit normal vector at s

Enforcing equality of curvature vectors at the knot point yields

$$K_i(\bar{s}_i) = K_{i+1}(\bar{s}_i)$$

Using the previously derived tangent continuity condition and defining

$$T_i(s) = \frac{dC_i(\bar{s}_i)/ds}{\|dC_i(\bar{s}_i)/ds\|}$$

leads to

$$T_i(\bar{s}_i) \times \left(\frac{d^2C_i(\bar{s}_i)/ds^2}{\|dC_i(\bar{s}_i)/ds\|^2} - \frac{d^2C_{i+1}(\bar{s}_i)/ds^2}{\|dC_{i+1}(\bar{s}_i)/ds\|^2} \right) \times T_i(\bar{s}_i) = 0$$

A solution of this condition can be written as

$$\frac{d^2C_i(\bar{s}_i)/ds^2}{\|dC_i(\bar{s}_i)/ds\|^2} = \frac{d^2C_{i+1}(\bar{s}_i)/ds^2}{\|dC_{i+1}(\bar{s}_i)/ds\|^2} + k_i \frac{dC_i(\bar{s}_i)/ds}{\|dC_i(\bar{s}_i)/ds\|}$$

with $k_i \in \mathbb{R}$. Substituting the Bézier derivative expressions and simplifying yields

$$P_n^i - 2P_{n-1}^i + P_{n-2}^i = \alpha_i^2 (P_2^{i+1} - 2P_1^{i+1} + P_0^{i+1}) + \beta_i (P_n^i - P_{n-1}^i) \quad (5.31)$$

$$\beta_i = k \frac{n}{(n-1)} \|(P_n^i - P_{n-1}^i)\| \quad (5.32)$$

Although the implementation in this dissertation uses C^2 reference paths, the G^2 relations above provide a useful geometric-smoothing alternative for future extensions.

5.2.7 Translation and Rotation of Composite Bézier Curves

Bézier curves are invariant under affine transformations [84]. Therefore, applying a rigid transformation to the control points and then evaluating the curve gives the same result as evaluating the curve first and then transforming the resulting point. Consider a composite Bézier curve consisting of m Bézier segments of degree n

$$C(s) = \sum_{i=0}^m [\alpha_i C_i(s)]$$

where

$$C_i(s) = \sum_{j=0}^n [P_j^i b_{j,n}^i(s)]$$

Suppose each control point is subjected to a rigid body transformation consisting of a rotation $R(\varphi)$ and translation T . The transformed control points can be written as

$$\tilde{P}_j^i = Q + R(\varphi)P_j^i, \quad i = 1, \dots, m, \quad j = 0, \dots, n$$

Substituting the transformed control points into the Bézier representation gives

$$\tilde{C}(s) = \sum_{i=0}^m \left[\alpha_i \sum_{j=0}^n [(Q + R(\varphi)P_j^i) b_{j,n}^i(s)] \right]$$

Expanding the summation yields

$$\tilde{C}(s) = Q \sum_{i=0}^m \left[\alpha_i \sum_{j=0}^n b_{j,n}^i(s) \right] + R(\varphi) \sum_{i=0}^m \left[\alpha_i \sum_{j=0}^n [P_j^i b_{j,n}^i(s)] \right]$$

Using the Bernstein partition-of-unity property for each segment

$$\sum_{j=0}^n b_{j,n}^i(s) = 1$$

and noting that exactly one segment is active for a given parameter value s , the above expression simplifies to

$$\tilde{C}(s) = Q + R(\varphi)C(s) \quad (5.33)$$

which shows that the composite Bézier curve undergoes the same rigid transformation as its control points.

5.3 Polyline-Scaled C^2 Cubic Bézier Paths

Let a discrete set of waypoints is given by

$$\mathcal{W} = \{w_0, w_1, \dots, w_m\} \subset \mathbb{R}^d, \quad d \in \{2, 3\} \quad (5.34)$$

From this set, a composite cubic Bézier curve is constructed to interpolate all waypoints while ensuring C^2 continuity between successive segments. The continuity scaling is based on the polyline chord lengths

$$l_i = \|w_i - w_{i-1}\|, \quad i = 1, \dots, m, \quad (5.35)$$

which approximate the local arc-length spacing between waypoints.

5.3.1 Parameterization and Segment Definition

Each Bézier segment is defined as a cubic curve

$$C_i(s) = \sum_{j=0}^3 \left(P_j^i b_{j,3}^i(s) \right), \quad s \in [\bar{s}_{i-1}, \bar{s}_i] \quad (5.36)$$

and each segment endpoints interpolate the waypoints

$$P_0^i = w_{i-1}, \quad P_3^i = w_i \quad (5.37)$$

while the intermediate control points P_1^i and P_2^i determine the tangent and the

curvature continuity between adjacent segments. Define the cumulative polyline parameter

$$\bar{s}_i = \sum_{k=1}^i l_k, \quad \bar{s}_0 = 0 \quad (5.38)$$

which provides a natural parameterization consistent with waypoint spacing.

5.3.2 Continuity and Boundary Conditions

For each cubic Bézier segment C_i , $i \in [1, m-1]$, the C^2 conditions using polyline based scaling can be expressed using (1.9) as:

$$\begin{aligned} (C^0): \quad & P_3^i = P_0^{i+1} = w_i \\ (C^1): \quad & l_{i+1}(P_3^i - P_2^i) = l_i(P_1^{i+1} - P_0^{i+1}) \\ (C^2): \quad & l_{i+1}^2(P_3^i - 2P_2^i + P_1^i) = l_i^2(P_2^{i+1} - 2P_1^{i+1} + P_0^{i+1}) \end{aligned}$$

Substituting $P_3^i = P_0^{i+1} = w_i$ gives the explicit coupling between the adjacent segments

$$\begin{aligned} l_{i+1}P_2^i + l_iP_1^{i+1} &= (l_i + l_{i+1})w_i \\ l_{i+1}^2(P_1^i - 2P_2^i) + l_i^2(2P_1^{i+1} - P_2^{i+1}) &= (l_i^2 - l_{i+1}^2)w_i, \quad i = 1, \dots, m-1 \end{aligned} \quad (5.39)$$

To obtain a unique solution for all control points, boundary conditions are imposed at the endpoints. A common choice is the natural C^2 boundary condition, which enforces zero second derivative at both ends:

$$\begin{aligned} 2P_1^1 - P_2^1 &= w_0 \\ -P_1^m + 2P_2^m &= w_m \end{aligned} \quad (5.40)$$

If boundary tangents or curvatures are prescribed, these equations can be replaced by the corresponding linear constraints.

5.3.3 Block-Tridiagonal Linear System

Define the unknown control-point vectors

$$X_i = \begin{bmatrix} P_1^i \\ P_2^i \end{bmatrix}, \quad i = 1, \dots, m$$

Stacking all unknowns gives

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_m \end{bmatrix} \in \mathbb{R}^{2md}$$

The continuity and boundary equations assemble into a block-tridiagonal linear system

$$KX = R$$

with the structure

$$\begin{bmatrix} B_1 & C_1 & 0 & \cdots & 0 & 0 \\ A_2 & B_2 & C_2 & \cdots & 0 & 0 \\ 0 & A_3 & B_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & B_{m-1} & C_{m-1} \\ 0 & 0 & 0 & \cdots & A_m & B_m \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{m-1} \\ X_m \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{m-1} \\ D_m \end{bmatrix} \quad (5.41)$$

Each 2×2 block is implicitly multiplied by the identity I_d and is given by

- Left boundary

$$B_1 = \begin{bmatrix} 2I_d & -I_d \\ 0 & l_2 I \end{bmatrix}, \quad C_1 = \begin{bmatrix} 0 & 0 \\ l_1 I & 0 \end{bmatrix}, \quad D_1 = \begin{bmatrix} w_0 \\ (l_1 + l_2)w_1 \end{bmatrix}$$

- Interior junctions $i \in [2, m-1]$

$$A_i = \begin{bmatrix} l_i^2 & -2l_i^2 I \\ 0 & 0 \end{bmatrix}, \quad B_i = \begin{bmatrix} 2l_{i-1}^2 I_d & -l_{i-1}^2 I_d \\ 0 & l_{i+1} I_d \end{bmatrix}, \quad C_i = \begin{bmatrix} 0 & 0 \\ l_i I & 0 \end{bmatrix},$$

$$D_i = \begin{bmatrix} (l_{i-1}^2 - l_i^2)w_{i-1} \\ (l_i + l_{i+1})w_i \end{bmatrix}$$

- Right boundary

$$A_m = \begin{bmatrix} l_m^2 & -2l_m^2 I \\ 0 & 0 \end{bmatrix}, \quad B_m = \begin{bmatrix} 2l_{m-1}^2 I_d & -l_{m-1}^2 I_d \\ -I_d & 2I_d \end{bmatrix},$$

$$D_m = \begin{bmatrix} (l_{m-1}^2 - l_m^2)w_{m-1} \\ w_m \end{bmatrix}$$

This system is block-tridiagonal and can be solved efficiently using a block Thomas algorithm. Each coordinate axis (x, y, z) can be solved independently.

5.3.4 Arc-Length Scaling Refinement

Because the continuity scaling uses polyline chord lengths l_i , the resulting parameterization approximates but does not exactly match true arc length. A simple one step refinement can improve approximation:

1. *Initial solve:* Use $l_i = \|w_i - w_{i-1}\|$ to compute all P_1^i, P_2^i .
2. *Compute true segment lengths:*

$$L_i = \int_0^1 \left\| \frac{dC_i}{d\tau} \right\| d\tau \quad (5.42)$$

3. *Update scaling:* Set $l_i \leftarrow L_i$ and resolve the same linear system

After one refinement, the ratios l_{i+1}/l_i better approximate the true segment-length ratios while preserving the same composite curve construction. The resulting spline is C^2 continuous, analytically differentiable, and suitable for optimization, collision avoidance, and curvature-based constraints in MPPFC. Because the control points are obtained from a block-tridiagonal system, the interpolation remains efficient for online path updates.

5.3.5 Solution of the Block-Tridiagonal System

The linear system (5.41) possesses a block tri-diagonal structure without

symmetry. Such systems can be solved effectively using the block Thomas algorithm with $O(md^3)$ computational complexity (where each block is $d \times d$) [88]. Block Thomas algorithm is as follows:

Schur-complement elimination: Write the first two block equations:

$$\begin{aligned} B_1 X_1 + C_1 X_2 &= D_1 \\ A_2 X_1 + B_2 X_2 + C_2 X_3 &= D_2 \end{aligned}$$

From the first equation,

$$X_1 = B_1^{-1}(D_1 - C_1 X_2)$$

Substitute this into the second:

$$(B_2 - A_2 B_1^{-1} C_1) X_2 + C_2 X_3 = D_2 - A_2 B_1^{-1} D_1$$

Define the Schur updates

$$\tilde{B}_i \triangleq B_i - A_i B_{i-1}^{-1} C_{i-1}, \quad \tilde{D}_i \triangleq D_i - A_i B_{i-1}^{-1} D_{i-1}, \quad \forall i \in [2, m] \quad (5.43)$$

so the second equation becomes

$$\tilde{B}_2 X_2 + C_2 X_3 = \tilde{D}_2$$

Using the Schur updates, and repeated elimination forward for $k = 1, \dots, m-1$

yields the reduced upper block-triangular system

$$\begin{bmatrix} B_1 & C_1 & 0 & \cdots & 0 & 0 \\ 0 & \tilde{B}_2 & C_2 & \cdots & 0 & 0 \\ 0 & 0 & \tilde{B}_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \tilde{B}_{m-1} & C_{m-1} \\ 0 & 0 & 0 & \cdots & 0 & \tilde{B}_m \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{m-1} \\ X_m \end{bmatrix} = \begin{bmatrix} D_1 \\ \tilde{D}_2 \\ \tilde{D}_3 \\ \vdots \\ \tilde{D}_{m-1} \\ \tilde{D}_m \end{bmatrix}$$

Solve the last block and back-substitute:

$$\tilde{B}_m X_m = \tilde{D}_m$$

$$X_k = \tilde{B}_k^{-1}(\tilde{D}_k - C_k X_{k+1}), \quad k = m-1, \dots, 1$$

Local Factorization and Numerical Issues: In practice we never form inverse. We apply $\tilde{B}_k^{-1}$ using triangular solves with a local LU factorization:

$$\Pi_k \tilde{B}_k = L_k U_k \quad (5.44)$$

where

- Π_k is a permutation matrix implementing partial pivoting,
- L_k is a lower triangular matrix with ones on its diagonal
- U_k is upper triangular.

We need Π_k because LU factorization of general nonsymmetric matrix

$$B_k = L_k U_k$$

may fail or be numerically unstable if a pivot is zero or very small. Small pivoting ensures that:

- The decomposition always exists
- Numerical stability is significantly improved

Solving $\tilde{B}_k^{-1}v$ becomes

$$\tilde{B}_k^{-1}v = U_k^{-1}L_k^{-1}(\Pi_k v)$$

Interface Operators and Back Substitution: Compute the interface operators via triangular solves (no explicit inverse):

$$Y_k = \tilde{B}_k^{-1}C_k, \quad W_k = \tilde{B}_k^{-1}\tilde{D}_k$$

These represent:

- Y_k : forward coupling operator (how X_{k+1} influences X_k)
- W_k : local solution contribution

The back substitution step becomes:

$$X_k = W_k - Y_k X_{k+1}, \quad k = m-1, \dots, 1 \quad (5.45)$$

Note also that:

$$\tilde{B}_k = B_k - A_k B_{k-1}^{-1} C_{k-1} = B_k - A_k Y_{k-1}$$

Remarks

- No matrix inverse is formed explicitly. All solutions use the local LU factors L_k, U_k .
- Numerical stability is ensured via pivoting Π_k .
- For symmetric $A_k = C_m^T$, one may replace LU with LDL^T factorization and eliminate pivoting except in the indefinite case.
- A compact pseudo-code formulation of Thomas method for block tridiagonal linear systems is outlined in algorithm 5.1, at the end of this section.

5.3.6 Waypoint Reduction by Polyline Simplification

Before fitting a smooth composite Bézier path, the raw waypoint sequence is simplified to remove redundant nearly collinear points while preserving the reference-path geometry within a prescribed tolerance. Let the set of waypoints W in (5.34) define the polyline path

$$\mathcal{C} = \bigcup_{k=0}^{m-1} [w_k, w_{k+1}]$$

where

$$[w_k, w_{k+1}] := \{w_k + \lambda(w_{k+1} - w_k) \mid \lambda \in [0,1]\}$$

denotes the closed line segment between w_k and w_{k+1} . Given a tolerance $\varepsilon > 0$, the aim is to construct a reduced set of waypoints

$$\tilde{w}_j = w_{i_j}, \quad j = 0, \dots, m, \quad 0 = i_0 < i_1 < \dots < i_m = n$$

Such that

1. The end points are preserved:

$$\tilde{w}_0 = w_0, \quad \tilde{w}_m = w_n$$

2. The simplified polygonal path

$$\tilde{\mathcal{C}} = \bigcup_{k=0}^{m-1} [\tilde{w}_k, \tilde{w}_{k+1}]$$

remains uniformly close to the original polyline, i.e.,

$$\max_{k=0, \dots, n} d(w_k, \tilde{\mathcal{C}}) \leq \varepsilon \tag{5.46}$$

$$s. t. \quad d(w_k, \tilde{\mathcal{C}}) = \min_{y \in \tilde{\mathcal{C}}} \|w_k - y\|_2 \tag{5.47}$$

The tolerance ε is chosen in physical units and controls the trade-off between fidelity to the original sampled path and the number of retained waypoints. In the context of MPPFC, this step reduces the number of segments for which composite Bézier control points must be computed, thereby decreasing the size of the block-tridiagonal system in the fitting stage while keeping the geometric error bounded.

Point-segment distance in $\mathbb{R}^d$

A key ingredient of the simplification algorithm is the distance between a point and a line segment. Given two distinct points $A, B \in \mathbb{R}^d$ and a point $P \in \mathbb{R}^d$, the distance from P to the segment $[A, B]$ is

$$d(P, [A, B]) = \min_{\lambda \in [0, 1]} \|A + \lambda(B - A) - P\|_2 \tag{5.48}$$

Let $v := B - A$, $w := P - A$. Then the unconstrained minimizer of $\lambda \rightarrow$

$\|A + \lambda v - P\|_2^2$ is given by

$$\lambda_{line}^* = \frac{w^T v}{v^T v}$$

Clamping to the segment yields

$$\lambda^* = \min(1, \max(0, \lambda_{line}^*))$$

and the squared distance is

$$d^2(P, [A, B]) = \|A + \lambda^* v - P\|_2^2$$

5.3.7 Ramer–Douglas–Peucker simplification

The reduced waypoint subset can be obtained using the Ramer-Douglas-Peucker (RDP) algorithm, [86], [87]. The algorithm recursively approximates a polyline with a fewer segments, while ensuring that the geometric deviation from the original path remains below a specified tolerance. Consider a contiguous index range $[k_0, k_1]$ with $0 \leq k_0 < k_1 \leq n$. The corresponding polyline sub-curve is approximated by the single segment $[w_{k_0}, w_{k_1}]$. The algorithm proceeds as follows:

1. *Initialization:* Mark w_{k_0} and w_{k_1} as “kept”
2. *Maximum deviation:* For all intermediate indices $k \in \{k_0 + 1, \dots, k_1 - 1\}$, compute the point-segment distance

$$d_k = d(w_k, [w_{k_0}, w_{k_1}])$$

$$d_{max} := \max_{k_0 < k < k_1} d_k, \quad k^* := \arg \max_{k_0 < k < k_1} d_k$$

3. *Termination or split:*

- If $d_{max} \leq \varepsilon$, then the straight segment $[w_{k_0}, w_{k_1}]$ is an acceptable approximation of the entire sub-curve and no intermediate points are

required. Only w_{k_0} and w_{k_1} are kept for this interval.

- If $d_{max} > \varepsilon$, then the approximation is too coarse. The point w_{k^*} must be retained to respect the error bound. The interval is split at k^* , and the procedure is applied recursively to the two sub-intervals $[k_0, k^*]$ and $[k^*, k_1]$.

Algorithmically, starting from the full index range $[0, n]$, the above procedure is applied recursively until all sub-intervals satisfy to yield the subsequence $\{i_0, \dots, i_m\}$ defining the reduced waypoint set $\{\tilde{w}_j\}$. A compact pseudo-code formulation is outlined in algorithm 5.2.

By construction, $\tilde{f}$ uses a subsequence of the original points and bounds the maximum deviation by ε . This is the standard form used in cartography and robotics [86]. The algorithm is invariant under rigid translations and rotations and extends directly to any dimension d because it uses only Euclidean point-segment distances in $\mathbb{R}^d$.

5.3.8 Computational Complexity

The reference-path construction consists of RDP waypoint reduction followed by composite cubic Bézier fitting with C^2 continuity. Let the original waypoint sequence contain m points and let the simplified sequence contain $\bar{m}$ retained waypoints, with $\bar{m} \leq m$. The classical recursive RDP algorithm has worst-case complexity $O(m^2)$, although typical geometric paths are closer to $O(m \log m)$ due to early termination [86], [87]. The Bézier fitting stage solves the block-tridiagonal system in (5.41), where each unknown block corresponds to the control

vector X_i , and the block matrices A_i, B_i, C_i have dimension $2d \times 2d$. Consequently, the expected runtime is approximately

$$O(RefPath) = O(n \log n) + O(\bar{m}d^2) \quad (5.49)$$

which scales nearly linearly with the number of input waypoints.

In real-time MPPFC applications, successive waypoint sets change gradually between control cycles. Therefore, the retained waypoint indices from the previous cycle can be used as a warm start and locally updated as points enter or leave the horizon window. This reduces the effective RDP recursion depth and keeps the Bézier fitting problem size nearly constant.

The overall implementation pipeline is summarized conceptually in Fig. 5.1.

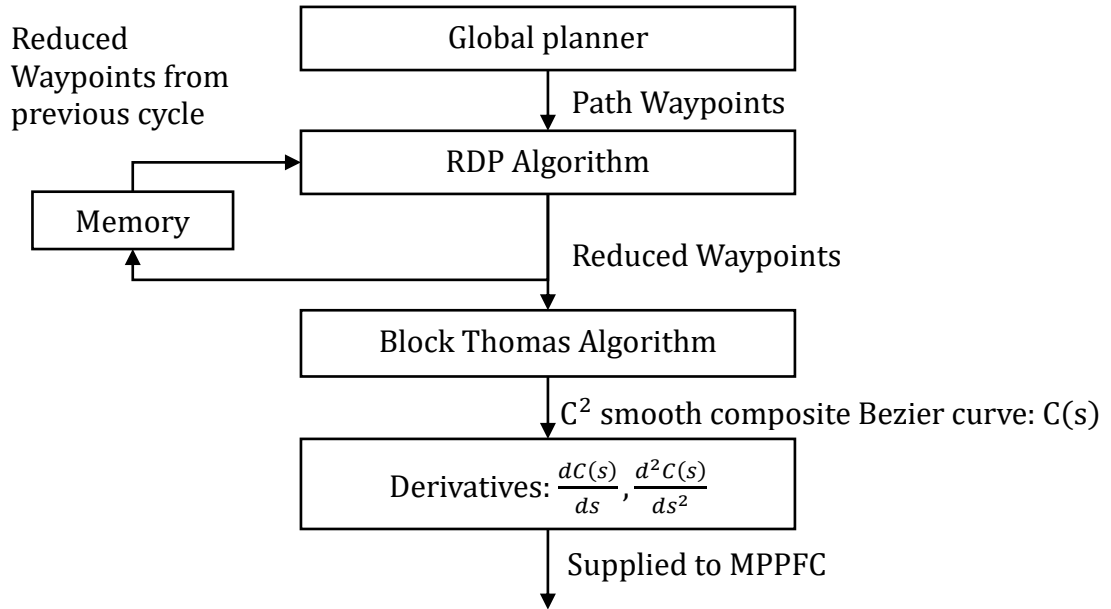


Figure 5.1: Real-Time Smooth Reference Path Generation Pipeline

Algorithm 5.1: Thomas Method for Block Tridiagonal Linear Systems

Consider the block tridiagonal linear system

$$\begin{bmatrix} B_1 & C_1 & 0 & \cdots & 0 & 0 \\ 0 & B_2 & C_2 & \cdots & 0 & 0 \\ 0 & 0 & B_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & B_{m-1} & C_{m-1} \\ 0 & 0 & 0 & \cdots & 0 & B_m \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{m-1} \\ X_m \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{m-1} \\ D_m \end{bmatrix}$$

where each block A_k, B_k, C_k and X_k, D_k is of dimension $n \times n$. The Block Thomas Algorithm computes its solution in $O(mn^3)$ flops without any matrix inverse.

Inputs:

- Block matrices $A_k, B_k, C_k \in \mathbb{R}^{n \times n}$, $k = 1, \dots, m$
- Right and side blocks $D_k \in \mathbb{R}^n$

Outputs: Solution blocks $X_k \in \mathbb{R}^n$

Procedure:

Stage 1: Forward Elimination (Schur Complement Recursion)

1. Initialize $\tilde{B}_1 = B_1$, $\tilde{D}_1 = D_1$
2. For $k = 2, \dots, m$

Perform the following steps

- a) Factorization of previous block:

$$\Pi_{k-1} \tilde{B}_{k-1} = L_{k-1} U_{k-1}$$

where Π_{k-1} denotes the partial-pivoting permutation

- b) Interface operator:

$$Y_{k-1} = \tilde{B}_{k-1}^{-1} C_{k-1} = U_{k-1}^{-1} L_{k-1}^{-1} \Pi_{k-1} C_{k-1}$$

- c) Modified right-hand side:

$$W_{k-1} = \tilde{B}_{k-1}^{-1} \tilde{D}_{k-1} = U_{k-1}^{-1} L_{k-1}^{-1} \Pi_{k-1} \tilde{D}_{k-1}$$

d) Schur complement updates

$$\tilde{B}_k = B_k - A_k Y_{k-1}, \quad \tilde{D}_k = D_k - A_k W_{k-1}$$

Stage 2: Backward Substitution

1. Solve the last block system: $X_m = \tilde{B}_m^{-1} \tilde{D}_m$
2. For $k = m - 1, \dots, 1$, compute: $X_k = W_k - Y_k X_{k+1}$

Algorithm 5.2: Ramer–Douglas–Peucker Polyline Simplification

Inputs: Ordered points $w_0, \dots, w_n \in \mathbb{R}^d$, tolerance ε

Output: Reduced index set $I \subset \{0, \dots, n\}$

Procedure:

1. Initialize: $I \leftarrow \emptyset$
2. Call $RDP(0, n)$, where $RDP(k_0, k_1)$ is:
 - If $k_1 \leq k_0 + 1$, add k_0 and k_1 to I and return
 - Compute $d_k = d(w_k, [w_{k_0}, w_{k_1}])$ for all $k_0 < k < k_1$.
 - Let $d_{max} = \max d_k$ and $k^* = \arg \max d_k$
 - If $d_{max} \leq \varepsilon$, add k_0 and k_1 to I and return.
 - Else, recursively call $RDP(k_0, k^*)$ and $RDP(k^*, k_1)$
3. After recursion, remove duplicate indices from I and sort in ascending order

5.4 Composite Bézier Path Examples

This section illustrates the waypoint-reduction and composite cubic Bézier fitting procedure developed in the preceding sections. The examples show that the same formulation applies to both planar and spatial waypoint sequences because the derivation is written for curves in $\mathbb{R}^d$, with $d \in \{2, 3\}$.

Planar Example

For a planar waypoint sequence $\{w_0, w_1, \dots, w_n\} \subset \mathbb{R}^2$, the raw waypoint polyline is first simplified using the RDP algorithm with tolerance ε . The retained waypoint set $\{\tilde{w}_0, \tilde{w}_1, \dots, \tilde{w}_{\tilde{n}}\}$ is then used to construct a C^2 -continuous composite cubic Bézier curve. Figure 5.2 shows the original waypoint polyline, the reduced waypoint set, and the fitted composite Bézier curve.

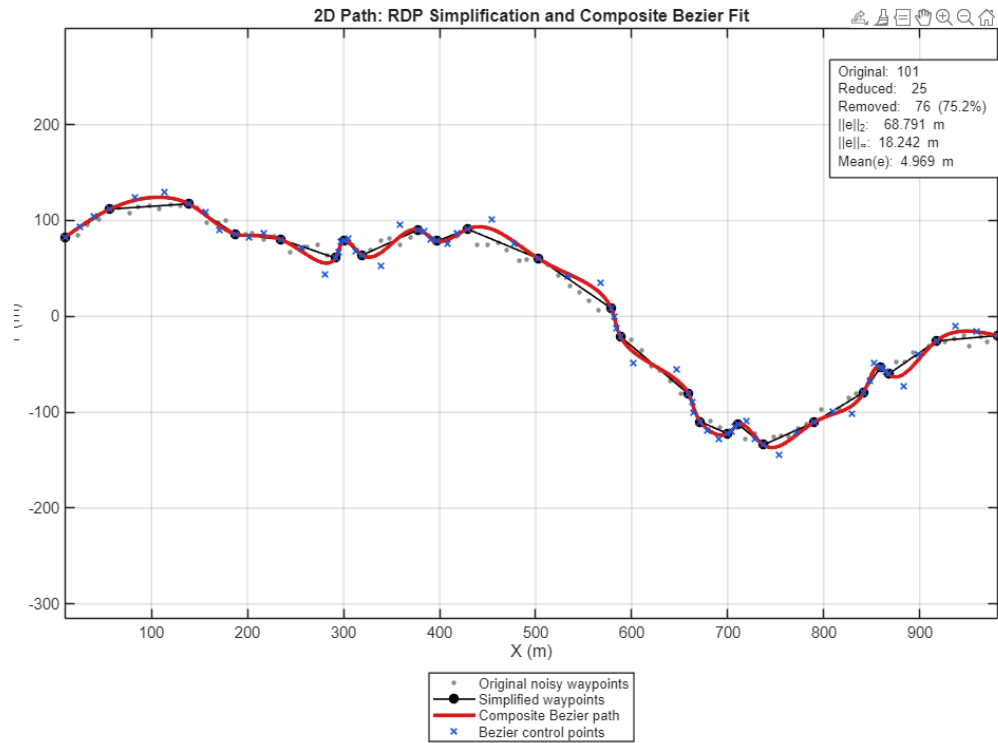


Figure 5.2: Smooth Composite Cubic Bézier Path Construction Example in $\mathbb{R}^2$

Spatial Example

The same procedure applies directly to spatial waypoint sequences $\{w_0, w_1, \dots, w_n\} \subset \mathbb{R}^3$. Since the derivation is formulated in $\mathbb{R}^d$, no separate three-dimensional construction is required. Figure 5.3 shows the spatial composite Bézier curve obtained from the reduced waypoint set.

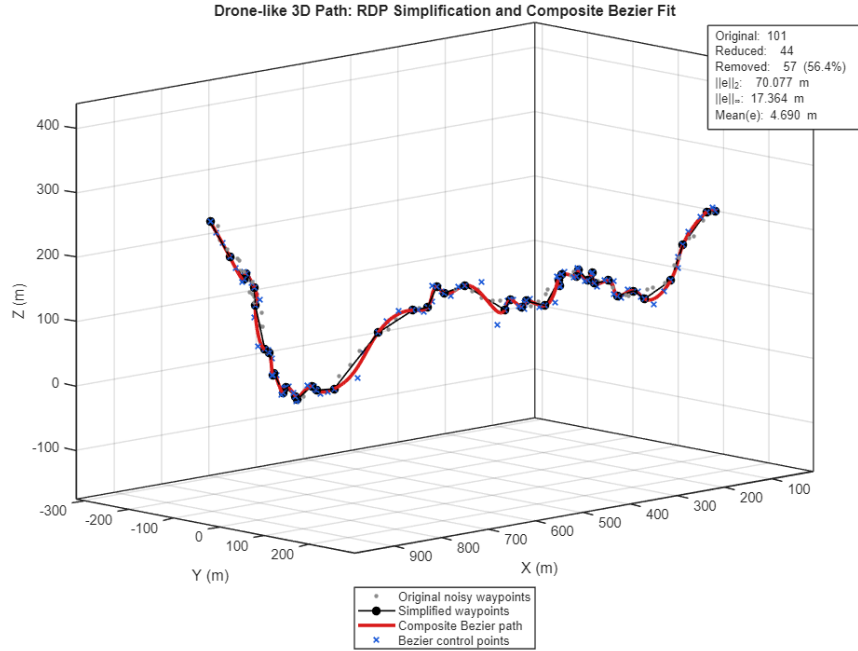


Figure 5.3: Smooth Composite Cubic Bézier Path Construction Example in $\mathbb{R}^3$

5.5 Chapter Summary

This chapter developed the smooth reference-path generation method used by the MPPFC framework. Dense waypoint sequences are first reduced using Ramer–Douglas–Peucker polyline simplification, which preserves the geometric structure of the path while reducing the number of interpolation points. The retained waypoints are then fitted with a composite cubic Bézier curve satisfying C^2 continuity conditions based on polyline-scaled segment lengths.

The resulting path representation is compact, analytically differentiable, and suitable for real-time optimization. Closed-form expressions for the curve derivatives provide the tangent, curvature, and higher-order path quantities required by the path-following and collision-avoidance formulations developed in the subsequent chapters.

CHAPTER 6 - STRUCTURE-EXPLOITING PRIMAL-DUAL INTERIOR-POINT METHOD FOR UNIFIED MPPFC

This chapter develops a structure-exploiting numerical method for solving the unified MPPFC problem formulated in Chapter 3. Starting from the discrete-time constrained optimal control problem (3.59), the derivation introduces a scaled primal-dual interior-point formulation and develops the Newton system, line-search and filter-based globalization safeguards for convergence from nonideal initial guesses, and computational accelerations used in the solver implementation in this dissertation. Interior-point methods are well suited for large-scale MPC problems because their Newton systems preserve the stagewise sparsity of the prediction model and can be solved efficiently using structured linear algebra techniques [47], [49], [53], [54].

6.1 Softened Unified MPPFC Formulation

A discrete-time unified MPPFC optimization problem with collision avoidance was formulated in Chapter 3, Eq. (3.59). Such optimization-based formulations of path-following control are widely used in nonlinear MPC frameworks [8], [9], [11], [12]. For notational convenience, the optimization variables are grouped as:

$$z_k = [x_k \quad u_k]^\top, \quad p_k = [\eta_k \quad \alpha_k]^\top \quad (6.1)$$

where z_k collects the system states and control inputs and p_k represents the path state vector consisting of the path coordinate and derivatives up-to a user desired degree. Furthermore, let

$$c_z(z_k, z_{k-1}) = -x_k + x_{k-1} + \frac{\tau}{2} [f(x_k, u_k) + f(x_{k-1}, u_{k-1})] \quad (6.2)$$

$$c_p(p_k, p_{k-1}) = -\eta_k + \eta_{k-1} + \frac{\tau}{2} [A_\eta(\eta_k + \eta_{k-1}) + B_\eta(\alpha_k + \alpha_{k-1})] \quad (6.3)$$

For compactness in the remainder of this chapter, J_k denotes the complete stage-assigned objective contribution, with the terminal path-progress term $-P_\theta \eta_{1,t+n_p}$ assigned to stage $k = t + n_p$ and the path-coordinate continuity term $\|\eta_{1,t} - \eta_{1,t-1}\|_{Q_\theta}^2$ assigned to stage $k = t$. Using these definitions, the optimal control problem can be written compactly as

$$\begin{aligned} \{z_k^*, p_k^*\}_{k=t}^{t+n_p} = & \underset{\{z_k, p_k\}_{k=t}^{t+n_p}}{\operatorname{argmin}} \sum_{k=t}^{t+n_p} J_k(z_k, p_k) \\ & \text{s.t.} \\ & \forall k \in [t, t + n_p] \quad \begin{aligned} c_z(z_k, z_{k-1}) &= 0, & z_{t-1} &= z_0 \\ c_p(p_k, p_{k-1}) &= 0, & p_{t-1} &= p_0 \\ c_e(z_k, p_k) &= 0 \\ \underline{z}_k &\leq z_k \leq \bar{z}_k \\ \underline{p}_k &\leq p_k \leq \bar{p}_k \\ c_i(z_k, p_k) &\leq 0 \end{aligned} \end{aligned} \quad (6.4)$$

The partition z_k and p_k separates the physical system variables from the path-state variables. This distinction is useful numerically because the two groups generally have different physical scales, bounds, and dynamic structures, allowing tailored primal scaling and softening parameters. It also exposes the structure of the equality constraints: c_z and c_p couple consecutive prediction stages, whereas c_e couples z_k and p_k only within each stage. Consequently, the linearized KKT system inherits a sparse stagewise structure that can be exploited in Jacobian assembly and structured linear-system solution [47], [52]- [54].

6.1.1 Slack and Softening Variables

The inequality constraints in (6.4) consist of simple variable bounds and the nonlinear inequalities $c_i(z_k, p_k) \leq 0$. Because strict initialization of the nonlinear inequalities may be difficult, particularly in collision-avoidance problems, nonnegative slack variables are introduced:

$$c_i(z_k, p_k) + s_k = 0, \quad s_k \geq 0 \quad (6.5)$$

This equality-plus-slack reformulation is standard in nonlinear programming and primal-dual interior-point methods [36], [49], [54]. To reduce infeasibility caused by restrictive inequality constraints and improve practical MPC solvability, the bounds are softened using relaxation variables ε_k [53]:

$$\underline{z}_k - \underline{w}_z \varepsilon_k \leq z_k \leq \bar{z}_k + \bar{w}_z \varepsilon_k \quad (6.6)$$

$$\underline{p}_k - \underline{w}_p \varepsilon_k \leq p_k \leq \bar{p}_k + \bar{w}_p \varepsilon_k \quad (6.7)$$

$$-\underline{w}_s \varepsilon_k \leq s_k \quad (6.8)$$

Here, the nonnegative weighting vectors $\underline{w}_z, \bar{w}_z, \underline{w}_p, \bar{w}_p, \underline{w}_s$ determine how relaxation is distributed among the softened inequalities. For a feasible point of the softened formulation, $\varepsilon_k = 0$ recovers the original inequalities; positive components of ε_k permit constraint relaxation, whereas negative components tighten inequalities having positive relaxation weights. Relaxation is discouraged through the quadratic penalty

$$\frac{1}{2} \varepsilon_k^T Q_{\varepsilon,k} \varepsilon_k, \quad Q_{\varepsilon,k} \geq 0$$

which encourages satisfaction of the original constraints whenever feasible [53].

6.1.2 Softened MPC Reformulation

Using slack and relaxation variables, the reformulated MPC problem is:

$$\begin{aligned}
 \{z_k^*, p_k^*, s_k^*, \varepsilon_k^*\}_{k=t}^{t+n_p} = & \underset{\{z_k, p_k, s_k, \varepsilon_k\}_{k=t}^{t+n_p}}{\operatorname{argmin}} \sum_{k=t}^{t+n_p} \left[J_k(z_k, p_k) + \frac{1}{2} \|\varepsilon_k\|_{Q_{\varepsilon,k}}^2 \right] \\
 \text{s.t.} & \\
 \forall k \in [t, t+n_p] & \quad c_z(z_k, z_{k-1}) = 0, z_{t-1} = z_0 \\
 & \quad c_p(p_k, p_{k-1}) = 0, p_{t-1} = p_0 \\
 & \quad c_e(z_k, p_k) = 0 \\
 & \quad c_i(z_k, p_k) + s_k = 0 \\
 & \quad \underline{z}_k - \underline{w}_z \varepsilon_k \leq z_k \leq \bar{z}_k + \bar{w}_z \varepsilon_k \\
 & \quad \underline{p}_k - \underline{w}_p \varepsilon_k \leq p_k \leq \bar{p}_k + \bar{w}_p \varepsilon_k \\
 & \quad -\underline{w}_s \varepsilon_k \leq s_k
 \end{aligned} \tag{6.9}$$

The resulting formulation converts the general nonlinear inequalities into equality constraints through the slack variables, while retaining the softened variable and slack bounds as simple inequalities. This structure is particularly well suited for primal-dual interior-point algorithms, which solve a sequence of barrier-perturbed optimality conditions approaching the solution of the original nonlinear program [36], [49], [54]. Such formulations are widely used in nonlinear model predictive control because they support structured numerical solution over prediction horizons of practical interest [46], [47], [53].

6.2 Scaled Interior-Point Reformulation

The softened MPC problem in (6.9) contains equality constraints together with simple inequalities. A logarithmic barrier incorporates these inequalities into the objective while retaining the equality constraints. For an inequality residual $r > 0$, the logarithmic barrier term is defined as $-\mu \log(r)$, where $\mu > 0$ is the barrier

parameter. For fixed μ , the barrier problem remains strictly interior; as $\mu \rightarrow 0$, its perturbed primal–dual optimality conditions approach the KKT conditions of the original constrained problem under standard regularity assumptions [36], [49].

6.2.1 Log-Barrier Form of Inequality Constraints

To construct the barrier formulation, the softened inequality constraints introduced earlier are first rewritten in residual form:

$$z_{l,k} = z_k - \underline{z}_k + \underline{w}_z \varepsilon_k, \quad z_{u,k} = -z_k + \bar{z}_k + \bar{w}_z \varepsilon_k \quad (6.10)$$

$$p_{l,k} = p_k - \underline{p}_k + \underline{w}_p \varepsilon_k, \quad p_{u,k} = -p_k + \bar{p}_k + \bar{w}_p \varepsilon_k \quad (6.11)$$

$$s_{l,k} = s_k + \underline{w}_s \varepsilon_k \quad (6.12)$$

These vectors denote residual margins associated with softened inequalities. In the original MPC problem, these residuals are nonnegative. In barrier formulation, they are restricted to be strictly positive so that the logarithmic terms are well defined.

Using the logarithmic barrier function, the MPC problem can be reformulated as

$$\begin{aligned} \{z_k^*, p_k^*, s_k^*, \varepsilon_k^*\}_{k=t}^{t+n_p} = \underset{\{z_k, p_k, s_k, \varepsilon_k\}_{k=t}^{t+n_p}}{\operatorname{argmin}} \quad & \sum_{k=t}^{t+n_p} \left[J_k(z_k, p_k) + \frac{1}{2} \|\varepsilon_k\|_{Q_{\varepsilon,k}}^2 - \right. \\ & \mu \sum_{j=1}^{n_x+n_u} \left(\log z_{l,k,j} + \log z_{u,k,j} \right) - \\ & \mu \sum_{j=1}^{n_\theta+1} \left(\log p_{l,k,j} + \log p_{u,k,j} \right) - \\ & \left. \mu \sum_{j=1}^{n_s} \log s_{l,k,j} \right] \quad (6.13) \\ \text{s.t.} \quad & \\ \forall k \in [t, t+n_p] \quad & c_z(z_k, z_{k-1}) = 0, z_{t-1} = z_0 \\ & c_p(p_k, p_{k-1}) = 0, p_{t-1} = p_0 \\ & c_e(z_k, p_k) = 0 \\ & c_i(z_k, p_k) + s_k = 0 \end{aligned}$$

In (6.13), $z_{l,k,j}$, $z_{u,k,j}$, $p_{l,k,j}$, $p_{u,k,j}$ and $s_{l,k,j}$ denote the j -th components of the corresponding residual vectors. Problem (6.13) is referred to as the MPC barrier problem. Its logarithmic barrier terms provide the basis for the perturbed primal-dual representation developed in Section 6.3 [46], [49].

6.2.2 Lagrangian Function of Barrier MPC Problem

The Lagrangian of the barrier MPC problem can be written as

$$L_\mu = \sum_{k=t}^{t+n_p} L_{\mu,k}(z_k, p_k, \varepsilon_k, s_k, \lambda_{z,k}, \lambda_{p,k}, \lambda_{e,k}, \lambda_{i,k}) \quad (6.14)$$

where

$$\begin{aligned} L_{\mu,k} = & J_k + \frac{1}{2} \|\varepsilon_k\|_{Q_{e,k}}^2 - \mu \sum_{j=1}^{n_x+n_u} \log z_{l,k,j} - \mu \sum_{j=1}^{n_x+n_u} \log z_{u,k,j} - \mu \sum_{j=1}^{n_\theta+1} \log p_{l,k,j} \\ & - \mu \sum_{j=1}^{n_\theta+1} \log p_{u,k,j} - \mu \sum_{j=1}^{n_s} \log s_{l,k,j} + \lambda_{z,k}^\top c_{z,k} + \lambda_{p,k}^\top c_{p,k} + \lambda_{e,k}^\top c_{e,k} \\ & + \lambda_{i,k}^\top (c_{i,k} + s_k) \end{aligned}$$

In this expression,

$$J_k \triangleq J(z_k, p_k)$$

$$c_{z,k} \triangleq c_z(z_k, z_{k-1}), \quad c_{p,k} \triangleq c_p(p_k, p_{k-1}), \quad c_{e,k} \triangleq c_e(z_k, p_k), \quad c_{i,k} \triangleq c_i(z_k, p_k)$$

and $\lambda_{z,k}$, $\lambda_{p,k}$, $\lambda_{e,k}$, $\lambda_{i,k}$ are the Lagrange multipliers associated with the corresponding equality constraints.

6.2.3 Variable and Constraint Scaling

The numerical performance of a Newton-based nonlinear optimization method depends strongly on the relative magnitudes of its primal variables,

constraint residuals, gradients, and Lagrange multipliers. If different components of the optimization problem vary significantly in magnitude, the resulting KKT linear systems can become poorly conditioned, producing inaccurate Newton directions and deteriorated solver convergence. Scaling is therefore introduced to improve the numerical conditioning of the primal-dual system without changing the underlying optimization problem [36], [49].

For a vector v , let v^{-1} represent its component wise inverse. All entries of the scaling vectors introduced below are assumed to be strictly positive. Let σ_z, σ_p , and σ_s denote the scaling vectors associated with the primal variables z_k, p_k , and s_k respectively. The scaled primal variables are defined as

$$z_k = \sigma_z^{-1} \odot \tilde{z}_k, \quad p_k = \sigma_p^{-1} \odot \tilde{p}_k, \quad s_k = \sigma_s^{-1} \odot \tilde{s}_k \quad (6.15)$$

The softening variable ε_k is not scaled in the present formulation. Let ρ_z, ρ_p, ρ_e and ρ_i denote the scaling vectors associated with the equality-constraint residuals. The scaled equality-constraint residuals are defined by

$$c_{z,k} = \rho_z^{-1} \odot \tilde{c}_{z,k}, \quad c_{p,k} = \rho_p^{-1} \odot \tilde{c}_{p,k}, \quad c_{e,k} = \rho_e^{-1} \odot \tilde{c}_{e,k},$$

$$c_{i,k} + s_k = \rho_i^{-1} \odot (\tilde{c}_{i,k} + \rho_i \odot \sigma_s^{-1} \odot \tilde{s}_k) \quad (6.16)$$

Since every component of ρ_z, ρ_p, ρ_e , and ρ_i is strictly positive, an equality-constraint residual is zero in the unscaled formulation if and only if the corresponding scaled residual is zero. To preserve the equality-constraint contributions to the Lagrangian under the scaling transformation, the Lagrange multipliers are defined according to

$$\lambda_{z,k} = \rho_z \odot \tilde{\lambda}_{z,k}, \quad \lambda_{p,k} = \rho_p \odot \tilde{\lambda}_{p,k}, \quad \lambda_{e,k} = \rho_e \odot \tilde{\lambda}_{e,k}, \quad \lambda_{i,k} = \rho_i \odot \tilde{\lambda}_{i,k} \quad (6.17)$$

Since the inequality residuals correspond directly to softened bounds on the primal variables, they are scaled using the associated primal-variable scaling:

$$\begin{aligned} z_{l,k} &= \sigma_z^{-1} \odot \tilde{z}_{l,k} \quad , \quad p_{u,k} = \sigma_p^{-1} \odot \tilde{p}_{u,k} \\ p_{l,k} &= \sigma_p^{-1} \odot \tilde{p}_{l,k} \quad , \quad z_{u,k} = \sigma_z^{-1} \odot \tilde{z}_{u,k} \\ s_{l,k} &= \sigma_s^{-1} \odot \tilde{s}_{l,k} \end{aligned} \quad (6.18)$$

Using (6.10)– (6.12) together with (6.15) and (6.18), the scaled inequality residuals are equivalently given by

$$\begin{aligned} \tilde{z}_{l,k} &= \tilde{z}_k - \sigma_z \odot \underline{z}_k + \sigma_z \odot (\underline{w}_z \varepsilon_k) \\ \tilde{z}_{u,k} &= -\tilde{z}_k + \sigma_z \odot \bar{z}_k + \sigma_z \odot (\bar{w}_z \varepsilon_k) \\ \tilde{p}_{l,k} &= \tilde{p}_k - \sigma_p \odot \underline{p}_k + \sigma_p \odot (\underline{w}_p \varepsilon_k) \\ \tilde{p}_{u,k} &= -\tilde{p}_k + \sigma_p \odot \bar{p}_k + \sigma_p \odot (\bar{w}_p \varepsilon_k) \\ \tilde{s}_{l,k} &= \tilde{s}_k + \sigma_s \odot (\underline{w}_s \varepsilon_k) \end{aligned} \quad (6.19)$$

Because all scaling factors are strictly positive, inequality feasibility and strict interiority are preserved by the scaling transformation. Using (6.16)– (6.19), the stagewise barrier Lagrangian in terms of the scaled variables can be written as

$$\begin{aligned} L_{\mu,k} &= J_k(\sigma_z^{-1} \odot \tilde{z}_k, \sigma_p^{-1} \odot \tilde{p}_k) + \frac{1}{2} \|\varepsilon_k\|_{Q_{\varepsilon,k}}^2 - \mu \sum_{j=1}^{n_x+n_u} \log(\sigma_{z,j}^{-1} \tilde{z}_{l,k,j}) \\ &\quad - \mu \sum_{j=1}^{n_x+n_u} \log(\sigma_{z,j}^{-1} \tilde{z}_{u,k,j}) - \mu \sum_{j=1}^{n_\theta+1} \log(\sigma_{p,j}^{-1} \tilde{p}_{l,k,j}) - \mu \sum_{j=1}^{n_\theta+1} \log(\sigma_{p,j}^{-1} \tilde{p}_{u,k,j}) \\ &\quad - \mu \sum_{j=1}^{n_s} \log(\sigma_{s,j}^{-1} \tilde{s}_{l,k,j}) + \tilde{\lambda}_{z,k}^\top \tilde{c}_{z,k} + \tilde{\lambda}_{p,k}^\top \tilde{c}_{p,k} + \tilde{\lambda}_{e,k}^\top \tilde{c}_{e,k} \\ &\quad + \tilde{\lambda}_{i,k}^\top (\tilde{c}_{i,k} + \rho_i \odot \sigma_s^{-1} \odot \tilde{s}_k) \end{aligned} \quad (6.20)$$

In (6.20), all nonlinear objective and constraint functions are evaluated using the corresponding unscaled variables recovered through (6.15). Because the scaling

vectors are component wise positive, the transformations in (6.15)– (6.19) are invertible and preserve equality feasibility, inequality feasibility, and strict interiority. Therefore, (6.20) is the exact scaled representation of the barrier Lagrangian associated with the softened MPC problem.

6.3 Perturbed Primal–Dual KKT Conditions

6.3.1 Dual Variables and Perturbed Complementarity

As the solution of the MPC barrier problem (6.13) approaches boundary of feasible region, one or more of the scaled inequality residuals $\tilde{z}_{l,k}$, $\tilde{z}_{u,k}$, $\tilde{p}_{l,k}$, $\tilde{p}_{u,k}$, $\tilde{s}_{l,k}$ may become small. Direct differentiation of the logarithmic barrier terms in (6.20) then produces reciprocal residual terms in the gradient and squared-reciprocal residual terms in the Hessian. Primal-dual interior-point methods avoid explicitly retaining these increasingly ill-conditioned reciprocal expressions by introducing additional variables associated with the inequality residuals [36], [49]. Define the lower-bound dual variables as

$$\tilde{\gamma}_{z,k} = \mu \tilde{z}_{l,k}^{-1}, \quad \tilde{\gamma}_{p,k} = \mu \tilde{p}_{l,k}^{-1}, \quad \tilde{\gamma}_{s,k} = \mu \tilde{s}_{l,k}^{-1}$$

and define the upper-bound dual variables as

$$\tilde{\beta}_{z,k} = \mu \tilde{z}_{u,k}^{-1}, \quad \tilde{\beta}_{p,k} = \mu \tilde{p}_{u,k}^{-1}$$

Here $\tilde{\gamma}$ and $\tilde{\beta}$ denote scaled dual variables associated with lower- and upper-bound residuals, respectively. Let e denote a vector of ones of dimension required by the corresponding residual vector:

$$e = [1 \quad \dots \quad 1]^T \tag{6.21}$$

The dual-variable definitions are equivalently expressed by the perturbed

complementarity equations

$$\tilde{\gamma}_{z,k} \odot \tilde{z}_{l,k} - \mu e = 0 \quad (6.22)$$

$$\tilde{\beta}_{z,k} \odot \tilde{z}_{u,k} - \mu e = 0 \quad (6.23)$$

$$\tilde{\gamma}_{p,k} \odot \tilde{p}_{l,k} - \mu e = 0 \quad (6.24)$$

$$\tilde{\beta}_{p,k} \odot \tilde{p}_{u,k} - \mu e = 0 \quad (6.25)$$

$$\tilde{\gamma}_{s,k} \odot \tilde{s}_{l,k} - \mu e = 0 \quad (6.26)$$

For $\mu > 0$, equations (6.22)–(6.26), together with the maintained interior conditions

$$\tilde{z}_{l,k}, \tilde{z}_{u,k}, \tilde{p}_{l,k}, \tilde{p}_{u,k}, \tilde{s}_{l,k} > 0, \quad \tilde{\gamma}_{z,k}, \tilde{\beta}_{z,k}, \tilde{\gamma}_{p,k}, \tilde{\beta}_{p,k}, \tilde{\gamma}_{s,k} > 0$$

define the perturbed complementarity system solved by the primal–dual algorithm.

As $\mu \rightarrow 0$, these relations approach the exact complementarity conditions of the original softened MPC problem [36], [49].

6.3.2 Lagrangian of the Softened MPC Problem

The barrier Lagrangian L_μ defined in Section 6.2.2 corresponds to the barrier reformulation of the softened MPC problem (6.13), for a fixed value of $\mu > 0$. The exact first- and second-order optimality conditions of the softened MPC problem in (6.9) are instead stated using its constrained-problem Lagrangian, denoted by L . In scaled variables, L is defined as

$$L = \sum_{k=t}^{t+n_p} L_k \quad (6.27)$$

where

$$\begin{aligned}
L_k = J_k + \frac{1}{2} \|\varepsilon_k\|_{Q_{\varepsilon,k}}^2 - \tilde{\gamma}_{z,k}^\top \tilde{z}_{l,k} - \tilde{\beta}_{z,k}^\top \tilde{z}_{u,k} - \tilde{\gamma}_{p,k}^\top \tilde{p}_{l,k} - \tilde{\beta}_{p,k}^\top \tilde{p}_{u,k} - \tilde{\gamma}_{s,k}^\top \tilde{s}_{l,k} + \tilde{\lambda}_{z,k}^\top \tilde{c}_{z,k} \\
+ \tilde{\lambda}_{p,k}^\top \tilde{c}_{p,k} + \tilde{\lambda}_{e,k}^\top \tilde{c}_{e,k} + \tilde{\lambda}_{i,k}^\top (\tilde{c}_{i,k} + \rho_i \odot \sigma_s^{-1} \odot \tilde{s}_k)
\end{aligned}$$

In contrast to L_μ , which contains logarithmic barrier terms, L contains explicit dual-variable terms associated with the softened-bound residuals. The exact complementarity conditions of the softened MPC problem are obtained from the perturbed relations (6.22)– (6.26) by replacing μe with zero. Accordingly, L is used to state the KKT and second-order optimality conditions of the constrained softened MPC problem, whereas (6.22)– (6.26) provide the perturbed complementarity relations used by the primal–dual interior-point algorithm for $\mu > 0$ [36], [49].

6.3.3 Lagrangian Gradient and Stationarity

The first-order KKT equations are obtained from the exact constrained-problem Lagrangian L defined in (6.27). Since the dynamic equality constraints couple successive stages, the stationarity equations associated with $\tilde{z}_k$ and $\tilde{p}_k$ contain multiplier contributions from both stages k and $k + 1$. Let

$$E_z = [I_{n_x} \quad 0_{n_x \times n_u}], \quad E_p = [I_{n_\eta} \quad 0_{n_\eta \times n_\alpha}] \quad (6.28)$$

The dynamics equality constraint is

$$c_{z,k} = c_z(z_k, z_{k-1}) = -x_k + x_{k-1} + \frac{\tau}{2} [f(x_k, u_k) + f(x_{k-1}, u_{k-1})]$$

Hence

$$D_{z,k} c_{z,k} = [D_{x,k} c_{z,k} \quad D_{u,k} c_{z,k}] = \left[-I + \frac{\tau}{2} D_{x,k} f_k \quad \frac{\tau}{2} D_{u,k} f_k \right] \quad (6.29)$$

Furthermore,

$$c_{z,k+1} = c_z(z_{k+1}, z_k) = -x_{k+1} + x_k + \frac{\tau}{2} [f(x_{k+1}, u_{k+1}) + f(x_k, u_k)]$$

and therefore

$$D_{z,k}c_{z,k+1} = 2E_{n_z} + D_{z,k}c_{z,k} \quad (6.30)$$

The path-dynamics equality constraint is

$$c_{p,k} = c_p(p_k, p_{k-1}) = -\eta_k + \eta_{k-1} + \frac{\tau}{2} [A_\eta(\eta_k + \eta_{k-1}) + B_\eta(\alpha_k + \alpha_{k-1})]$$

Hence,

$$D_{p,k}c_{p,k} = \left[-I + \frac{\tau}{2} A_\eta \quad \frac{\tau}{2} B_\eta \right] \quad (6.31)$$

$$D_{p,k}c_{p,k+1} = 2E_{n_p} + D_{p,k}c_{p,k} \quad (6.32)$$

For notational convenience, define $\tilde{\lambda}_{z,t+n_p+1} = 0$, $\tilde{\lambda}_{p,t+n_p+1} = 0$. Then, for all $t \leq$

$k \leq t + n_p$,

$$\begin{aligned} \nabla_{\tilde{z},k} L = & \sigma_z^{-1} \odot [\nabla_{z,k} J_k + (\nabla_{z,k} c_{z,k}^\top)(\rho_z \odot \tilde{\lambda}_{z,k}) + (2E_z^\top + \nabla_{z,k} c_{z,k}^\top)(\rho_z \odot \tilde{\lambda}_{z,k+1}) \\ & + (\nabla_{z,k} c_{e,k}^\top)(\rho_e \odot \tilde{\lambda}_{e,k}) + (\nabla_{z,k} c_{i,k}^\top)(\rho_i \odot \tilde{\lambda}_{i,k})] - \tilde{\gamma}_{z,k} + \tilde{\beta}_{z,k} \end{aligned} \quad (6.33)$$

$$\begin{aligned} \nabla_{\tilde{p},k} L = & \sigma_p^{-1} \odot [\nabla_{p,k} J_k + (\nabla_{p,k} c_{p,k}^\top)(\rho_p \odot \tilde{\lambda}_{p,k}) + (2E_p^\top + \nabla_{p,k} c_{p,k}^\top)(\rho_p \odot \tilde{\lambda}_{p,k+1}) \\ & + (\nabla_{p,k} c_{e,k}^\top)(\rho_e \odot \tilde{\lambda}_{e,k}) + (\nabla_{p,k} c_{i,k}^\top)(\rho_i \odot \tilde{\lambda}_{i,k})] - \tilde{\gamma}_{p,k} + \tilde{\beta}_{p,k} \end{aligned} \quad (6.34)$$

$$\nabla_{\tilde{s},k} L = \sigma_s^{-1} \odot \rho_i \odot \tilde{\lambda}_{i,k} - \tilde{\gamma}_{s,k} \quad (6.35)$$

$$\begin{aligned} \nabla_{\varepsilon,k} L = & Q_{\varepsilon,k} \varepsilon_k - \underline{w}_z^\top (\sigma_z \odot \tilde{\gamma}_{z,k}) - \bar{w}_z^\top (\sigma_z \odot \tilde{\beta}_{z,k}) - \underline{w}_p^\top (\sigma_p \odot \tilde{\gamma}_{p,k}) - \bar{w}_p^\top (\sigma_p \odot \tilde{\beta}_{p,k}) \\ & - \underline{w}_s^\top (\sigma_s \odot \tilde{\gamma}_{s,k}) \end{aligned} \quad (6.36)$$

$$\nabla_{\tilde{\lambda}_{z,k}} L = \rho_z \odot c_{z,k} \quad (6.37)$$

$$\nabla_{\tilde{\lambda}_{p,k}} L = \rho_p \odot c_{p,k} \quad (6.38)$$

$$\nabla_{\tilde{\lambda}_{e,k}} L = \rho_e \odot c_{e,k} \quad (6.39)$$

$$\nabla_{\tilde{\lambda}_{i,k}} L = \rho_i \odot (c_{i,k} + \sigma_s^{-1} \odot \tilde{s}_k) \quad (6.40)$$

Since $c_{z,k}$, $c_{p,k}$, $c_{e,k}$, and $c_{i,k}$ are vector-valued functions, the conversion between Jacobian and gradient-transpose notation is given by

$$\nabla_{z,k} c_{z,k}^\top = (D_{z,k} c_{z,k})^\top$$

with analogous definitions for all other vector-valued constraint functions appearing above.

Equality-Constraint Jacobian Structure

Let $\sigma = \text{vec}(\sigma_z, \sigma_p, \sigma_s, e_\varepsilon)$ and $\rho = \text{vec}(\rho_z, \rho_p, \rho_e, \rho_i)$, where e_ε denotes a vector of ones having the same dimension as ε_k , since ε_k is not scaled. All primal-variable transformations and residuals are implemented using the component wise multiplication operator $\odot$. In the following expressions, diagonal-matrix notation is introduced only to represent the assembled Jacobian compactly. The diagonal block of scaled equality-constraint Jacobian associated with stage k is

$$G_{k,k} = \text{diag}(\rho) \begin{bmatrix} (\nabla_{z,k} c_{z,k}^\top)^\top & 0 & 0 & 0 \\ 0 & (\nabla_{p,k} c_{p,k}^\top)^\top & 0 & 0 \\ (\nabla_{z,k} c_{e,k}^\top)^\top & (\nabla_{p,k} c_{e,k}^\top)^\top & 0 & 0 \\ (\nabla_{z,k} c_{i,k}^\top)^\top & (\nabla_{p,k} c_{i,k}^\top)^\top & I & 0 \end{bmatrix} \text{diag}(\sigma^{-1}) \quad (6.41)$$

For $t \leq k < t + n_p$, the sub diagonal block arising from the dynamic coupling between stages k and $k + 1$ is

$$G_{k+1,k} = \text{diag}(\rho) \begin{bmatrix} 2E_z + (\nabla_{z,k} c_{z,k}^\top)^\top & 0 & 0 & 0 \\ 0 & 2E_p + (\nabla_{p,k} c_{p,k}^\top)^\top & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{diag}(\sigma^{-1}) \quad (6.42)$$

Only the dynamic constraints $c_{z,k+1}$ and $c_{p,k+1}$ depend on the primal variables of stage k . The stagewise algebraic equality constraints $c_{e,k}$ and $c_{i,k} + s_k$ therefore contribute only to the diagonal blocks of the equality-constraint Jacobian. The complete equality-constraint Jacobian has the lower block-bidiagonal structure

$$G = \begin{bmatrix} G_{t,t} & 0 & \dots & 0 & 0 \\ G_{t+1,t} & G_{t+1,t+1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & G_{t+n_p-1,t+n_p-1} & 0 \\ 0 & 0 & \dots & G_{t+n_p,t+n_p-1} & G_{t+n_p,t+n_p} \end{bmatrix} \quad (6.43)$$

Lemma 6.1: First-Order Necessary Optimality Conditions for MPC Problem

(6.9)

Let $G^* = G(\tilde{z}^*, \tilde{p}^*)$ denote the scaled equality-constraint Jacobian evaluated at a candidate local solution. G_A^* denote the Jacobian obtained by augmenting G^* with the gradients of all inequality residuals that are active at the candidate solution. Suppose that:

1. J, f, c_e, c_i are continuously differentiable;
2. G_A^* has full row rank, so that the linear independence constraint qualification holds at the candidate solution;
3. $(\tilde{z}^*, \tilde{p}^*, \tilde{s}^*, \varepsilon^*)$ is a local minimizer of the scaled representation of the softened MPC problem (6.9).

Then there exist scaled equality-constraint Lagrange multipliers $\tilde{\lambda}_z^*, \tilde{\lambda}_p^*, \tilde{\lambda}_e^*$, $\tilde{\lambda}_i^*$ and scaled bound dual variables $\tilde{\gamma}_z^*, \tilde{\beta}_z^*, \tilde{\gamma}_p^*, \tilde{\beta}_p^*, \tilde{\gamma}_s^*$ such that the following conditions hold for all $t \leq k \leq t + n_p$, with all quantities evaluated at the

corresponding optimal primal-dual point:

Stationarity

$$\nabla_{\tilde{z},k} L = 0, \quad \nabla_{\tilde{p},k} L = 0, \quad \nabla_{\tilde{s},k} L = 0, \quad \nabla_{\varepsilon,k} L = 0 \quad (6.44)$$

Equality Feasibility

$$\nabla_{\tilde{\lambda}_z,k} L = 0, \quad \nabla_{\tilde{\lambda}_p,k} L = 0, \quad \nabla_{\tilde{\lambda}_e,k} L = 0, \quad \nabla_{\tilde{\lambda}_i,k} L = 0 \quad (6.45)$$

Inequality Feasibility

$$\tilde{z}_{l,k} \geq 0, \quad \tilde{z}_{u,k} \geq 0, \quad \tilde{p}_{l,k} \geq 0, \quad \tilde{p}_{u,k} \geq 0, \quad \tilde{s}_{l,k} \geq 0 \quad (6.46)$$

Dual Feasibility:

$$\tilde{\gamma}_{z,k} \geq 0, \quad \tilde{\beta}_{z,k} \geq 0, \quad \tilde{\gamma}_{p,k} \geq 0, \quad \tilde{\beta}_{p,k} \geq 0, \quad \tilde{\gamma}_{s,k} \geq 0 \quad (6.47)$$

Complementarity

$$\begin{aligned} \tilde{\gamma}_{z,k} \odot \tilde{z}_{l,k} &= 0, & \tilde{\beta}_{z,k} \odot \tilde{z}_{u,k} &= 0, \\ \tilde{\gamma}_{p,k} \odot \tilde{p}_{l,k} &= 0, & \tilde{\beta}_{p,k} \odot \tilde{p}_{u,k} &= 0, \\ \tilde{\gamma}_{s,k} \odot \tilde{s}_{l,k} &= 0 \end{aligned} \quad (6.48)$$

Proof:

The scaling transformations introduced in Section 6.2.3 are positive and invertible; therefore, feasible points and KKT points correspond one-to-one between the scaled and unscaled formulations. Under the linear independence constraint qualification (LICQ), the existence of multipliers satisfying the stated first-order conditions follows from Theorem 12.1 in [36].

Equations (6.44)– (6.48) are the exact KKT conditions of the softened MPC problem in scaled variables. In the primal–dual interior-point method, only the exact complementarity conditions in (6.48) are replaced by the perturbed relations

(6.22)– (6.26).

6.4 Second-Order Optimality Conditions

6.4.1 Second-Order Lagrangian Derivatives

Let $\tilde{v}_k = \text{vec}(\tilde{z}_k, \tilde{p}_k, \tilde{s}_k, \varepsilon_k)$ denote the scaled primal decision vector at stage k . Define the stagewise primal Hessian block of the exact constrained-problem Lagrangian L as

$$Q_k \triangleq \nabla_{\tilde{v},k}(D_{\tilde{v},k}L)$$

Partition Q_k according to the stagewise primal variables as

$$Q_k = \begin{bmatrix} Q_{zz,k} & Q_{zp,k}^\top & Q_{zs,k}^\top & Q_{z\varepsilon,k}^\top \\ Q_{zp,k} & Q_{pp,k} & Q_{ps,k}^\top & Q_{p\varepsilon,k}^\top \\ Q_{zs,k} & Q_{ps,k} & Q_{ss,k} & Q_{s\varepsilon,k}^\top \\ Q_{z\varepsilon,k} & Q_{p\varepsilon,k} & Q_{s\varepsilon,k} & Q_{\varepsilon\varepsilon,k} \end{bmatrix}$$

For $(x, y) \in \{z, p, s, \varepsilon\}$, define the component Hessian blocks as

$$Q_{xy,k} \triangleq \nabla_{\tilde{y},k}(D_{\tilde{x},k}L)$$

The direct second-derivative block associated with $\tilde{z}_k$ is

$$\begin{aligned} Q_{zz,k} = & [\sigma_z^{-1} \sigma_z^{-\top}] \odot \left[\nabla_{zz,k}^2 J_k + \left(I_{(n_x+n_u)} \otimes (\rho_z \odot (\tilde{\lambda}_{z,k} + \tilde{\lambda}_{z,k+1}))^\top \right) \nabla_{zz,k}^2 c_{z,k} \right. \\ & + \left(I_{(n_x+n_u)} \otimes (\rho_e \odot \tilde{\lambda}_{e,k})^\top \right) \nabla_{zz,k}^2 c_{e,k} \\ & \left. + \left(I_{(n_x+n_u)} \otimes (\rho_i \odot \tilde{\lambda}_{i,k})^\top \right) \nabla_{zz,k}^2 c_{i,k} \right] \end{aligned} \quad (6.49)$$

where $\tilde{\lambda}_{z,t+n_p+1} = 0$. The mixed second-derivative associated with $\tilde{p}_k$ and $\tilde{z}_k$ is

$$\begin{aligned} Q_{zp,k} = & [\sigma_p^{-1} \sigma_z^{-\top}] \odot \left[\nabla_{zp,k}^2 J_k + \left(I_{(n_\theta+1)} \otimes (\rho_e \odot \tilde{\lambda}_{e,k})^\top \right) \nabla_{zp,k}^2 c_{e,k} \right. \\ & \left. + \left(I_{(n_\theta+1)} \otimes (\rho_i \odot \tilde{\lambda}_{i,k})^\top \right) \nabla_{zp,k}^2 c_{i,k} \right] \end{aligned} \quad (6.50)$$

Neither dynamic constraint contributes to $Q_{zp,k}$, since neither contains joint dependence on z_k and p_k . The direct second-derivative block associated with $\tilde{p}_k$ is

$$Q_{pp,k} = [\sigma_p^{-1} \sigma_p^{-\top}] \odot \left[\nabla_{pp,k}^2 J_k + \left(I_{(n_\theta+1)} \otimes (\rho_e \odot \tilde{\lambda}_{e,k})^\top \right) \nabla_{pp,k}^2 c_{e,k} + \left(I_{(n_\theta+1)} \otimes (\rho_i \odot \tilde{\lambda}_{i,k})^\top \right) \nabla_{pp,k}^2 c_{i,k} \right] \quad (6.51)$$

The path-dynamics constraint $c_{p,k}$ is linear in p_k and p_{k-1} , and therefore contributes no second-derivative term to $Q_{pp,k}$.

The scaled slack variable $\tilde{s}_k$ appears linearly in L , while ε_k appears quadratically only through the softening penalty and linearly in the softened bound residuals. Therefore,

$$Q_{zs,k} = 0, \quad Q_{ps,k} = 0, \quad Q_{ss,k} = 0, \quad Q_{s\varepsilon,k} = 0, \quad Q_{z\varepsilon,k} = 0, \quad Q_{p\varepsilon,k} = 0, \\ Q_{\varepsilon\varepsilon,k} = Q_{\varepsilon,k}$$

Consequently, the stagewise primal Hessian block reduces to

$$Q_k = \begin{bmatrix} Q_{zz,k} & Q_{zp,k}^\top & 0 & 0 \\ Q_{zp,k} & Q_{pp,k} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{\varepsilon,k} \end{bmatrix}$$

Because the stage objective is separable, the algebraic constraints are stagewise, and the trapezoidal dynamic constraints are additively separable in consecutive-stage variables, no cross-stage second derivatives arise in the present formulation. Hence, the full primal Hessian is block diagonal

$$Q = \text{blkdiag} \left(Q_t, Q_{t+1}, \dots, Q_{t+n_p} \right)$$

Inter-stage penalties, such as control-rate or path-control-rate penalties, would

generally introduce off-diagonal Hessian blocks. Exploiting the block-diagonal Hessian together with the block-bidiagonal equality-constraint Jacobian is central to efficient structured MPC implementations [46], [47], [53], [54].

6.4.2 Necessary and Sufficient Second-Order Conditions

Let $\tilde{v} \triangleq \text{vec}(\tilde{v}_t, \tilde{v}_{t+1}, \dots, \tilde{v}_{t+n_p})$ denote the complete scaled primal-variable vector over the prediction horizon. Define the stagewise softened-bound residual vector and its associated bound-dual vector as

$$\tilde{r}_k \triangleq \text{vec}(\tilde{z}_{l,k}, \tilde{z}_{u,k}, \tilde{p}_{l,k}, \tilde{p}_{u,k}, \tilde{s}_{l,k}), \quad \tilde{\pi}_k \triangleq \text{vec}(\tilde{\gamma}_{z,k}, \tilde{\beta}_{z,k}, \tilde{\gamma}_{p,k}, \tilde{\beta}_{p,k}, \tilde{\gamma}_{s,k})$$

and their corresponding horizon-level vectors as

$$\tilde{r} \triangleq \text{vec}(\tilde{r}_t, \tilde{r}_{t+1}, \dots, \tilde{r}_{t+n_p}), \quad \tilde{\pi} \triangleq \text{vec}(\tilde{\pi}_t, \tilde{\pi}_{t+1}, \dots, \tilde{\pi}_{t+n_p})$$

At a KKT point, define the set of active softened-bound residuals as

$$A^* \triangleq \{j: \tilde{r}_j^* = 0\}$$

and partition this set according to the associated bound-dual variables:

$$A_+^* \triangleq \{j \in A^*: \tilde{\pi}_j^* > 0\}, \quad A_0^* \triangleq \{j \in A^*: \tilde{\pi}_j^* = 0\}.$$

Here, A^* is the active-set index set, while A_+^* and A_0^* denote its positive-multiplier and zero-multiplier partitions, respectively. Let ξ denote a direction in the space of $\tilde{v}$. Define

$$D^* \triangleq \left\{ \xi: \begin{array}{ll} G^* \xi = 0, \\ (\nabla_{\tilde{v}} \tilde{r}_j^{*\top})^\top \xi = 0, & j \in A_+^*, \\ (\nabla_{\tilde{v}} \tilde{r}_j^{*\top})^\top \xi \geq 0, & j \in A_0^* \end{array} \right\} \quad (6.52)$$

The set D^* consists of directions that satisfy the linearized equality constraints and the linearized active-bound conditions associated with the multiplier partition in

(6.52). Let

$$Q^* \triangleq \nabla_{\tilde{v}}^2 L(\tilde{v}^*, \tilde{\lambda}^*, \tilde{\pi}^*)$$

denote the primal Hessian of the exact constrained-problem Lagrangian L , evaluated at the KKT point.

Lemma 6.2: Second-Order Necessary Optimality Conditions for MPC Problem

Suppose that J , f , c_e , and c_i are twice continuously differentiable, G_A^* has full row rank, and $\tilde{v}^*$ is a local minimizer of the scaled representation of the softened MPC problem (6.9), with associated multipliers satisfying the KKT conditions in Lemma 6.1. Then

$$\xi^\top Q^* \xi \geq 0, \quad \forall \xi \in \mathcal{D}^*, \quad \xi \neq 0 \quad (6.53)$$

Proof:

Since G_A^* has full row rank, the linear independence constraint qualification holds at the candidate solution. The result follows by applying the second-order necessary optimality condition in Theorem 12.5 of [36] to the scaled representation of the MPC problem. Since the scaling transformations in Section 6.2.3 are positive and invertible, the condition is equivalent to the corresponding condition for the original unscaled MPC problem.

Lemma 6.3: Second-Order Sufficient Optimality Conditions for MPC Problem

Suppose that J , f , c_e , and c_i are twice continuously differentiable and that a candidate primal-dual point satisfies the KKT conditions in Lemma 6.1. If

$$\xi^\top Q^* \xi > 0, \quad \forall \xi \in \mathcal{D}^*, \quad \xi \neq 0 \quad (6.54)$$

then $\tilde{v}^*$ is a strict local minimizer of the scaled representation of the MPC problem.

Consequently, the corresponding unscaled primal point is a strict local minimizer of the softened MPC problem (6.9).

Proof:

The result follows by applying the second-order sufficient optimality condition in Theorem 12.6 of [36] to the scaled representation of the MPC problem. Equivalence of strict local optimality in the scaled and unscaled variables follows from the positive and invertible scaling transformations in Section 6.2.3.

Lemmas 6.2 and 6.3 characterize the curvature of the exact Lagrangian over the critical-direction set D^* : nonnegative curvature is necessary for local optimality, whereas strictly positive curvature is sufficient for a strict local minimizer [36]. In this work, the MPC solution is obtained by applying Newton iterations to the perturbed primal–dual system for $\mu > 0$, together with step-size control and globalization to preserve strict interiority and promote residual reduction. The following sections derive these algorithmic components.

6.5 Structure-Exploiting Newton Step Computation

6.5.1 Reduced Primal-Dual Newton System

For a fixed barrier parameter $\mu > 0$, the primal-dual interior-point method solves the stationarity and equality-feasibility conditions (6.33)–(6.40) associated with the exact constrained-problem Lagrangian L , together with the perturbed complementarity equations (6.22)–(6.26). These equations form a nonlinear algebraic system in the primal variables, equality-constraint multipliers, and bound dual variables. A Newton search direction is obtained by applying a first-order

Taylor expansion to this system at the current primal-dual iterate. In Section 6.4.1, the primal Hessian blocks of the exact Lagrangian were denoted by

$$Q_{zz,k}, \quad Q_{zp,k}, \quad Q_{pp,k}, \quad Q_{\varepsilon,k}$$

where the notation Q distinguishes the exact Lagrangian curvature from the modified curvature obtained in this section after eliminating the bound-dual search directions. Define the component wise complementarity-ratio vectors as

$$\begin{aligned} \Sigma_{l,z,k} &\triangleq \tilde{z}_{l,k}^{-1} \odot \tilde{\gamma}_{z,k} & , & & \Sigma_{u,z,k} &\triangleq \tilde{z}_{u,k}^{-1} \odot \tilde{\beta}_{z,k} \\ \Sigma_{l,p,k} &\triangleq \tilde{p}_{l,k}^{-1} \odot \tilde{\gamma}_{p,k} & , & & \Sigma_{u,p,k} &\triangleq \tilde{p}_{u,k}^{-1} \odot \tilde{\beta}_{p,k} \\ \Sigma_{l,s,k} &\triangleq \tilde{s}_{l,k}^{-1} \odot \tilde{\gamma}_{s,k} \end{aligned} \quad (6.55)$$

Linearization of (6.22)- (6.26) allows the dual-variable search directions to be eliminated in favor of the primal search directions:

$$\begin{aligned} d\tilde{\gamma}_{z,k} &= -\left[\Sigma_{l,z,k} \odot \left(d\tilde{z}_k + \sigma_z \odot (\underline{w}_z d\varepsilon_k)\right) + \tilde{\gamma}_{z,k} - \mu \tilde{z}_{l,k}^{-1}\right] \\ d\tilde{\beta}_{z,k} &= -\left[\Sigma_{u,z,k} \odot \left(-d\tilde{z}_k + \sigma_z \odot (\bar{w}_z d\varepsilon_k)\right) + \tilde{\beta}_{z,k} - \mu \tilde{z}_{u,k}^{-1}\right] \\ d\tilde{\gamma}_{p,k} &= -\left[\Sigma_{l,p,k} \odot \left(d\tilde{p}_k + \sigma_p \odot (\underline{w}_p d\varepsilon_k)\right) + \tilde{\gamma}_{p,k} - \mu \tilde{p}_{l,k}^{-1}\right] \\ d\tilde{\beta}_{p,k} &= -\left[\Sigma_{u,p,k} \odot \left(-d\tilde{p}_k + \sigma_p \odot (\bar{w}_p d\varepsilon_k)\right) + \tilde{\beta}_{p,k} - \mu \tilde{p}_{u,k}^{-1}\right] \\ d\tilde{\gamma}_{s,k} &= -\left[\Sigma_{l,s,k} \odot \left(d\tilde{s}_k + \sigma_s \odot (\underline{w}_s d\varepsilon_k)\right) + \tilde{\gamma}_{s,k} - \mu \tilde{s}_{l,k}^{-1}\right] \end{aligned} \quad (6.56)$$

To express the remaining Newton equations compactly, define the individual scaled equality-constraint Jacobian sub blocks as

$$\begin{aligned}
G_{zz,k,k} &\triangleq \text{diag}(\rho_z)(\nabla_{z,k}c_{z,k}^\top)^\top \text{diag}(\sigma_z^{-1}) \\
G_{zz,k+1,k} &\triangleq \text{diag}(\rho_z)(2E_z + (\nabla_{z,k}c_{z,k}^\top)^\top) \text{diag}(\sigma_z^{-1}) \\
G_{ze,k} &\triangleq \text{diag}(\rho_e)(\nabla_{z,k}c_{e,k}^\top)^\top \text{diag}(\sigma_z^{-1}) \\
G_{zi,k} &\triangleq \text{diag}(\rho_i)(\nabla_{z,k}c_{i,k}^\top)^\top \text{diag}(\sigma_z^{-1}) \\
G_{pp,k,k} &\triangleq \text{diag}(\rho_p)(\nabla_{p,k}c_{p,k}^\top)^\top \text{diag}(\sigma_p^{-1}) \\
G_{pp,k+1,k} &\triangleq \text{diag}(\rho_p)(2E_p + (\nabla_{p,k}c_{p,k}^\top)^\top) \text{diag}(\sigma_p^{-1}) \\
G_{pe,k} &\triangleq \text{diag}(\rho_e)(\nabla_{p,k}c_{e,k}^\top)^\top \text{diag}(\sigma_p^{-1}) \\
G_{pi,k} &\triangleq \text{diag}(\rho_i)(\nabla_{p,k}c_{i,k}^\top)^\top \text{diag}(\sigma_p^{-1}) \\
G_{si} &\triangleq \text{diag}(\rho_i \odot \sigma_s^{-1})
\end{aligned} \tag{6.57}$$

The blocks $G_{zz,k,k}$ and $G_{pp,k,k}$ are contained in the diagonal Jacobian block $G_{k,k}$, while $G_{zz,k+1,k}$ and $G_{pp,k+1,k}$ are contained in the subdiagonal block $G_{k+1,k}$ defined in (6.41) and (6.42) respectively. Substitution of the eliminated bound-dual search directions in (6.56) into the linearized stationarity equations introduces additional diagonal and primal-softening coupling terms. Define modified curvature blocks as

$$\begin{aligned}
H_{zz,k} &\triangleq Q_{zz,k} + \text{diag}(\Sigma_{l,z,k} + \Sigma_{u,z,k}) \\
H_{pp,k} &\triangleq Q_{pp,k} + \text{diag}(\Sigma_{l,p,k} + \Sigma_{u,p,k}) \\
H_{ss,k} &\triangleq \text{diag}(\Sigma_{l,s,k})
\end{aligned} \tag{6.58}$$

$$\begin{aligned}
H_{ze,k} &\triangleq \underline{w}_z^\top \text{diag}(\Sigma_{l,z,k} \odot \sigma_z) - \bar{w}_z^\top \text{diag}(\Sigma_{u,z,k} \odot \sigma_z) \\
H_{pe,k} &\triangleq \underline{w}_p^\top \text{diag}(\Sigma_{l,p,k} \odot \sigma_p) - \bar{w}_p^\top \text{diag}(\Sigma_{u,p,k} \odot \sigma_p) \\
H_{se,k} &\triangleq \underline{w}_s^\top \text{diag}(\Sigma_{l,s,k} \odot \sigma_s)
\end{aligned} \tag{6.59}$$

$$\begin{aligned}
H_{\varepsilon\varepsilon,k} &\triangleq Q_{\varepsilon,k} + \underline{w}_z^\top \text{diag}(\sigma_z \odot \Sigma_{l,z,k} \odot \sigma_z) \underline{w}_z + \bar{w}_z^\top \text{diag}(\sigma_z \odot \Sigma_{u,z,k} \odot \sigma_z) \bar{w}_z \\
&\quad + \underline{w}_p^\top \text{diag}(\sigma_p \odot \Sigma_{l,p,k} \odot \sigma_p) \underline{w}_p + \bar{w}_p^\top \text{diag}(\sigma_p \odot \Sigma_{u,p,k} \odot \sigma_p) \bar{w}_p \\
&\quad + \underline{w}_s^\top \text{diag}(\sigma_s \odot \Sigma_{l,s,k} \odot \sigma_s) \underline{w}_s
\end{aligned} \tag{6.60}$$

The residual terms introduced by the bound-dual elimination are

$$\begin{aligned}
r_{z,k} &\triangleq \nabla_{\tilde{z},k} L + \tilde{\gamma}_{z,k} - \mu \tilde{z}_{l,k}^{-1} - \tilde{\beta}_{z,k} + \mu \tilde{z}_{u,k}^{-1} \\
r_{p,k} &\triangleq \nabla_{\tilde{p},k} L + \tilde{\gamma}_{p,k} - \mu \tilde{p}_{l,k}^{-1} - \tilde{\beta}_{p,k} + \mu \tilde{p}_{u,k}^{-1} \\
r_{s,k} &\triangleq \nabla_{\tilde{s},k} L + \tilde{\gamma}_{s,k} - \mu \tilde{s}_{l,k}^{-1}
\end{aligned} \tag{6.61}$$

$$\begin{aligned}
r_{\varepsilon,k} &\triangleq \nabla_{\varepsilon,k} L + \underline{w}_z^\top [\sigma_z \odot (\tilde{\gamma}_{z,k} - \mu \tilde{z}_{l,k}^{-1})] + \bar{w}_z^\top [\sigma_z \odot (\tilde{\beta}_{z,k} - \mu \tilde{z}_{u,k}^{-1})] \\
&\quad + \underline{w}_p^\top [\sigma_p \odot (\tilde{\gamma}_{p,k} - \mu \tilde{p}_{l,k}^{-1})] + \bar{w}_p^\top [\sigma_p \odot (\tilde{\beta}_{p,k} - \mu \tilde{p}_{u,k}^{-1})] \\
&\quad + \underline{w}_s^\top [\sigma_s \odot (\tilde{\gamma}_{s,k} - \mu \tilde{s}_{l,k}^{-1})]
\end{aligned} \tag{6.62}$$

Finally, define the equality-feasibility residuals as

$$r_{\lambda_z,k} \triangleq \nabla_{\tilde{\lambda}_z,k} L, \quad r_{\lambda_p,k} \triangleq \nabla_{\tilde{\lambda}_p,k} L, \quad r_{\lambda_e,k} \triangleq \nabla_{\tilde{\lambda}_e,k} L, \quad r_{\lambda_i,k} \triangleq \nabla_{\tilde{\lambda}_i,k} L \tag{6.63}$$

For notational convenience, let

$$d\tilde{\lambda}_{z,t+n_p+1} = 0, \quad d\tilde{\lambda}_{p,t+n_p+1} = 0, \quad d\tilde{z}_{t-1} = 0, \quad d\tilde{p}_{t-1} = 0$$

here the last two conditions follow from the fixed initial conditions. The reduced

Newton equations associated with stationarity and equality feasibility are then

$$\begin{aligned}
H_{zz,k} d\tilde{z}_k + Q_{zp,k}^\top d\tilde{p}_k + H_{z\varepsilon,k}^\top d\varepsilon_k \\
+ G_{zz,k,k}^\top d\tilde{\lambda}_{z,k} + G_{zz,k+1,k}^\top d\tilde{\lambda}_{z,k+1} + G_{ze,k}^\top d\tilde{\lambda}_{e,k} + G_{zi,k}^\top d\tilde{\lambda}_{i,k} = -r_{z,k}
\end{aligned} \tag{6.64}$$

$$\begin{aligned}
Q_{zp,k} d\tilde{z}_k + H_{pp,k} d\tilde{p}_k + H_{p\varepsilon,k}^\top d\varepsilon_k \\
+ G_{pp,k,k}^\top d\tilde{\lambda}_{p,k} + G_{pp,k+1,k}^\top d\tilde{\lambda}_{p,k+1} + G_{pe,k}^\top d\tilde{\lambda}_{e,k} + G_{pi,k}^\top d\tilde{\lambda}_{i,k} = -r_{p,k}
\end{aligned} \tag{6.65}$$

$$H_{ss,k} d\tilde{s}_k + H_{s\varepsilon,k}^\top d\varepsilon_k + G_{si,k}^\top d\tilde{\lambda}_{i,k} = -r_{s,k} \tag{6.66}$$

$$H_{z\varepsilon,k} d\tilde{z}_k + H_{p\varepsilon,k} d\tilde{p}_k + H_{s\varepsilon,k} d\tilde{s}_k + H_{\varepsilon\varepsilon,k} d\varepsilon_k = -r_{\varepsilon,k} \tag{6.67}$$

$$G_{zz,k,k-1} d\tilde{z}_{k-1} + G_{zz,k,k} d\tilde{z}_k = -r_{\lambda_z,k} \tag{6.68}$$

$$G_{pp,k,k-1} d\tilde{p}_{k-1} + G_{pp,k,k} d\tilde{p}_k = -r_{\lambda_p,k} \tag{6.69}$$

$$G_{ze,k} d\tilde{z}_k + G_{pe,k} d\tilde{p}_k = -r_{\lambda_e,k} \tag{6.70}$$

$$G_{zi,k} d\tilde{z}_k + G_{pi,k} d\tilde{p}_k + G_{si,k} d\tilde{s}_k = -r_{\lambda_i,k} \tag{6.71}$$

Equations (6.64)–(6.71) define the reduced primal-dual Newton system

obtained after elimination of the bound-dual search directions. Stacking the stagewise primal directions and equality-multiplier directions over the prediction horizon gives.

$$\begin{bmatrix} H_{zz} & Q_{zp}^\top & 0 & H_{z\epsilon}^\top & G_{zz}^\top & 0 & G_{ze}^\top & G_{zi}^\top \\ Q_{zp} & H_{pp} & 0 & H_{p\epsilon}^\top & 0 & G_{pp}^\top & G_{pe}^\top & G_{pi}^\top \\ 0 & 0 & H_{ss} & H_{s\epsilon}^\top & 0 & 0 & 0 & G_{si}^\top \\ H_{z\epsilon} & H_{p\epsilon} & H_{s\epsilon} & H_{\epsilon\epsilon} & 0 & 0 & 0 & 0 \\ G_{zz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & G_{pp} & 0 & 0 & 0 & 0 & 0 & 0 \\ G_{ze} & G_{pe} & 0 & 0 & 0 & 0 & 0 & 0 \\ G_{zi} & G_{pi} & G_{si} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d\tilde{z} \\ d\tilde{p} \\ d\tilde{s} \\ d\tilde{\epsilon} \\ d\tilde{\lambda}_z \\ d\tilde{\lambda}_p \\ d\tilde{\lambda}_e \\ d\tilde{\lambda}_i \end{bmatrix} = - \begin{bmatrix} r_z \\ r_p \\ r_s \\ r_\epsilon \\ r_{\lambda_z} \\ r_{\lambda_p} \\ r_{\lambda_e} \\ r_{\lambda_i} \end{bmatrix} \quad (6.72)$$

In the above system, the modified curvature blocks and residual vectors are assembled from their stagewise components. For example,

$$H_{zz} = \text{blkdiag}(H_{zz,t}, \dots, H_{zz,t+n_p}), \quad r_z = \text{vec}(r_{z,t}, \dots, r_{z,t+n_p})$$

Similarly, H_{pp} , H_{ss} , $H_{z\epsilon}$, $H_{p\epsilon}$, $H_{s\epsilon}$, $H_{\epsilon\epsilon}$, Q_{zp} , together with the remaining residual vectors, are assembled analogously. Since the algebraic equality constraints are stagewise, G_{ze} , G_{pe} , G_{zi} , G_{pi} , and G_{si} are block diagonal. In contrast, the dynamic-constraint Jacobians G_{zz} and G_{pp} retain lower block-bidiagonal inter-stage coupling.

In particular,

$$G_{zz} = \begin{bmatrix} G_{zz,t,t} & 0 & \dots & 0 & 0 \\ G_{zz,t+1,t} & G_{zz,t+1,t+1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & G_{zz,t+n_p-1,t+n_p-1} & 0 \\ 0 & 0 & \dots & G_{zz,t+n_p,t+n_p-1} & G_{zz,t+n_p,t+n_p} \end{bmatrix} \quad (6.73)$$

with G_{pp} defined analogously.

Let H denote the assembled modified primal-curvature matrix and G denote

the assembled equality-constraint Jacobian matrix appearing in the reduced Newton system. Since the resulting KKT matrix may fail to possess the non-singularity and curvature properties required for computing a reliable Newton direction, structured regularization is introduced before solving the system. Motivated by the primal- and constraint-block perturbation strategy used in IPOPT [49], define the reduced primal-dual KKT matrix used in the present formulation as

$$K \triangleq \begin{bmatrix} H + \Delta_h & G^T \\ G & -\Delta_c \end{bmatrix}, \quad \Delta_h \geq 0, \quad \Delta_c \geq 0 \quad (6.74)$$

where Δ_h and Δ_c are block diagonal structured primal and equality regularization matrices, respectively. Consistent with the primal-variable partition in the reduced Newton system, define

$$\Delta_h \triangleq \text{blkdiag}(\Delta_z, \Delta_p, 0, 0), \quad \Delta_z \geq 0, \quad \Delta_p \geq 0 \quad (6.75)$$

where

$$\Delta_z = \text{blkdiag}(\Delta_{z,t}, \dots, \Delta_{z,t+n_p}), \quad \Delta_p = \text{blkdiag}(\Delta_{p,t}, \dots, \Delta_{p,t+n_p}) \quad (6.76)$$

When regularization is required, the nonzero stagewise blocks $\Delta_{z,k}$ and $\Delta_{p,k}$ are selected to be symmetric positive definite. Since, at strictly interior iterates, $H_{ss,k} = \text{diag}(\Sigma_{l,s,k}) > 0$, and, provided $Q_{\varepsilon,k} > 0$, $H_{\varepsilon\varepsilon,k} > 0$, no independent regularization is introduced in the $\tilde{s}$ or ε -blocks. Constraint-block regularization is applied only to the dynamic and user-defined equality constraints

$$\Delta_c \triangleq \text{blkdiag}(\Delta_{c_z}, \Delta_{c_p}, \Delta_{c_e}, 0), \quad \Delta_{c_z} \geq 0, \quad \Delta_{c_p} \geq 0, \quad \Delta_{c_e} \geq 0 \quad (6.77)$$

No constraint-block regularization is introduced for the equality constraint formed from the softened inequality constraint, since its Jacobian with respect to $\tilde{s}_k$

contains the nonsingular block G_{si} , as defined in (6.57). Thus, the regularization preserves the block-sparse structure of the reduced Newton system while modifying only the primal-curvature blocks for which additional curvature may be required.

For the perturbed system with $\mu > 0$, the corresponding reduced-space curvature requirement is imposed on the regularized primal block:

$$\xi^\top (H + \Delta_h) \xi > 0, \quad \forall \xi \neq 0 : G\xi = 0$$

The above condition is the finite- μ counterpart of the sufficient curvature condition in Lemma 6.3: Q^* characterizes the exact Lagrangian curvature of the softened MPC problem, whereas $H + \Delta_h$ is the regularized primal-curvature matrix used in the perturbed Newton system. Under the appropriate rank assumptions, this reduced-space curvature requirement can be assessed through the inertia of the KKT matrix K [36]. The structure-exploiting computation of the Newton direction is developed next, followed by the inertia condition used to select the required regularization.

6.5.2 Efficient Search-Direction Computation

The reduced Newton system in Section 6.5.1 can be solved efficiently by first eliminating the search directions associated with $\tilde{s}_k$, ε_k , and $\tilde{\lambda}_{i,k}$. The resulting condensed system contains only the directions associated with $\tilde{z}_k$, $\tilde{p}_k$, $\tilde{\lambda}_{z,k}$, $\tilde{\lambda}_{p,k}$, and $\tilde{\lambda}_{e,k}$. Because no constraint-block regularization is applied to the equality constraint formed from the softened inequality constraint, and since G_{si} is invertible, its feasibility equation gives

$d\tilde{s}_k = -(\rho_i^{-1} \odot \sigma_s) \odot (G_{zi,k} d\tilde{z}_k + G_{pi,k} d\tilde{p}_k + r_{\lambda_{i,k}})$	(6.78)
--	----------

Substituting (6.78) into the stationarity equation associated with $\tilde{s}_k$ (6.66) yields

$$d\tilde{\lambda}_{i,k} = (\Sigma_{l,s,k} \odot \rho_i^{-2} \odot \sigma_s^2) \odot (G_{zi,k} d\tilde{z}_k + G_{pi,k} d\tilde{p}_k + r_{\lambda_i,k}) - (\rho_i^{-1} \odot \sigma_s) \odot (H_{se,k}^\top d\varepsilon_k + r_{s,k}) \quad (6.79)$$

Substitution of (6.78) into the stationarity equation associated with ε_k (6.67) gives

$$\bar{H}_{ze,k} d\tilde{z}_k + \bar{H}_{pe,k} d\tilde{p}_k + H_{\varepsilon\varepsilon,k} d\varepsilon_k = -\bar{r}_{\varepsilon,k}$$

where

$$\begin{aligned} \bar{H}_{ze,k} &\triangleq H_{ze,k} - H_{se,k} \text{diag}(\rho_i^{-1} \odot \sigma_s) G_{zi,k} \\ \bar{H}_{pe,k} &\triangleq H_{pe,k} - H_{se,k} \text{diag}(\rho_i^{-1} \odot \sigma_s) G_{pi,k} \end{aligned} \quad (6.80)$$

$$\bar{r}_{\varepsilon,k} \triangleq r_{\varepsilon,k} - H_{se,k} (\rho_i^{-1} \odot \sigma_s \odot r_{\lambda_i,k}) \quad (6.81)$$

Since $H_{\varepsilon\varepsilon,k} > 0$, the softening variable search direction is

$$d\varepsilon_k = -H_{\varepsilon\varepsilon,k}^{-1} (\bar{H}_{ze,k} d\tilde{z}_k + \bar{H}_{pe,k} d\tilde{p}_k + \bar{r}_{\varepsilon,k}) \quad (6.82)$$

Substituting (6.79) and (6.82) into the stationary equations associated with $\tilde{z}_k$ and $\tilde{p}_k$, i.e., (6.64) and (6.65) respectively gives the condensed curvature blocks

$$\begin{aligned} \bar{H}_{zz,k} &\triangleq H_{zz,k} + G_{zi,k}^\top \text{diag}(\Sigma_{l,s,k} \odot \rho_i^{-2} \odot \sigma_s^2) G_{zi,k} - \bar{H}_{ze,k}^\top H_{\varepsilon\varepsilon,k}^{-1} \bar{H}_{ze,k} + \Delta_{z,k} \\ \bar{H}_{zp,k} &\triangleq Q_{zp,k} + G_{pi,k}^\top \text{diag}(\Sigma_{l,s,k} \odot \rho_i^{-2} \odot \sigma_s^2) G_{zi,k} - \bar{H}_{pe,k}^\top H_{\varepsilon\varepsilon,k}^{-1} \bar{H}_{ze,k} \\ \bar{H}_{pp,k} &\triangleq H_{pp,k} + G_{pi,k}^\top \text{diag}(\Sigma_{l,s,k} \odot \rho_i^{-2} \odot \sigma_s^2) G_{pi,k} - \bar{H}_{pe,k}^\top H_{\varepsilon\varepsilon,k}^{-1} \bar{H}_{pe,k} + \Delta_{p,k} \end{aligned} \quad (6.83)$$

$$\begin{aligned} \bar{r}_{z,k} &\triangleq \left[r_{z,k} - \bar{H}_{ze,k}^\top H_{\varepsilon\varepsilon,k}^{-1} \bar{r}_{\varepsilon,k} - G_{zi,k}^\top (\rho_i^{-1} \odot \sigma_s \odot r_{s,k}) + G_{zi,k}^\top (\Sigma_{l,s,k} \odot \rho_i^{-2} \odot \sigma_s^2 \odot r_{\lambda_i,k}) \right] \\ \bar{r}_{p,k} &\triangleq \left[r_{p,k} - \bar{H}_{pe,k}^\top H_{\varepsilon\varepsilon,k}^{-1} \bar{r}_{\varepsilon,k} - G_{pi,k}^\top (\rho_i^{-1} \odot \sigma_s \odot r_{s,k}) + G_{pi,k}^\top (\Sigma_{l,s,k} \odot \rho_i^{-2} \odot \sigma_s^2 \odot r_{\lambda_i,k}) \right] \end{aligned} \quad (6.84)$$

The condensed stationarity equations are therefore

$$\bar{H}_{zz,k} d\tilde{z}_k + \bar{H}_{zp,k}^\top d\tilde{p}_k + G_{zz,k,k}^\top d\tilde{\lambda}_{z,k} + G_{zz,k+1,k}^\top d\tilde{\lambda}_{z,k+1} + G_{ze,k}^\top d\tilde{\lambda}_{e,k} = -\bar{r}_{z,k} \quad (6.85)$$

$$\bar{H}_{zp,k} d\tilde{z}_k + \bar{H}_{pp,k} d\tilde{p}_k + G_{pp,k,k}^\top d\tilde{\lambda}_{p,k} + G_{pp,k+1,k}^\top d\tilde{\lambda}_{p,k+1} + G_{pe,k}^\top d\tilde{\lambda}_{e,k} = -\bar{r}_{p,k} \quad (6.86)$$

and the remaining feasibility equations are

$$G_{zz,k,k-1} d\tilde{z}_{k-1} + G_{zz,k,k} d\tilde{z}_k - \Delta_{c_z,k} d\tilde{\lambda}_{z,k} = -r_{\lambda_z,k} \quad (6.87)$$

$$G_{pp,k,k-1}d\tilde{p}_{k-1} + G_{pp,k,k}d\tilde{p}_k - \Delta_{c,p,k}d\tilde{\lambda}_{p,k} = -r_{\lambda_p,k} \quad (6.88)$$

$$G_{ze,k}d\tilde{z}_k + G_{pe,k}d\tilde{p}_k - \Delta_{c,e,k}d\tilde{\lambda}_{e,k} = -r_{\lambda_e,k} \quad (6.89)$$

For $k = t$, the predecessor-stage directions in (6.87) and (6.88) are zero because the initial conditions are fixed. Define

$$d\chi_k \triangleq \text{vec}(d\tilde{z}_k, d\tilde{p}_k), \quad d\Lambda_k \triangleq \text{vec}(d\tilde{\lambda}_{z,k}, d\tilde{\lambda}_{p,k}, d\tilde{\lambda}_{e,k}) \quad (6.90)$$

$$\bar{H}_k \triangleq \begin{bmatrix} \bar{H}_{zz,k} & \bar{H}_{zp,k}^\top \\ \bar{H}_{zp,k} & \bar{H}_{pp,k} \end{bmatrix}, \quad \bar{r}_k \triangleq \begin{bmatrix} \bar{r}_{z,k} \\ \bar{r}_{p,k} \end{bmatrix} \quad (6.91)$$

$$\bar{G}_{k,k} \triangleq \begin{bmatrix} G_{zz,k,k} & 0 \\ 0 & G_{pp,k,k} \\ G_{ze,k} & G_{pe,k} \end{bmatrix}, \quad \bar{G}_{k+1,k} \triangleq \begin{bmatrix} G_{zz,k+1,k} & 0 \\ 0 & G_{pp,k+1,k} \\ 0 & 0 \end{bmatrix} \quad (6.92)$$

Stacking the stagewise quantities over the prediction horizon gives

$$\begin{bmatrix} \bar{H} & \bar{G}^\top \\ \bar{G} & -\Delta_c \end{bmatrix} \begin{bmatrix} d\chi \\ d\Lambda \end{bmatrix} = - \begin{bmatrix} \bar{r} \\ \nabla_\Lambda L \end{bmatrix} \quad (6.93)$$

Here $\bar{H} = \text{blkdiag}(\bar{H}_t, \dots, \bar{H}_{t+n_p})$, while $\bar{G}$ is assembled from the diagonal and subdiagonal blocks in (6.92) using the sacking convention in (6.73) and therefore retains a lower block-bidiagonal structure. Provided that $\bar{H}$ is nonsingular, the primal directions can be eliminated from (6.93) giving

$$(\bar{G}\bar{H}^{-1}\bar{G}^\top + \Delta_c)d\Lambda = \nabla_\Lambda L - \bar{G}\bar{H}^{-1}\bar{r} \quad (6.94)$$

$$d\chi = -\bar{H}^{-1}(\bar{r} + \bar{G}^\top d\Lambda) \quad (6.95)$$

Since $\bar{H}$ is block diagonal and $\bar{G}$ is lower block bidiagonal, the Schur-complement matrix $\bar{G}\bar{H}^{-1}\bar{G}^\top + \Delta_c$ is symmetric block tri-diagonal. Hence it can be factorized using block $LDL^\top$ factorization as

$$LDL^\top = \bar{G}\bar{H}^{-1}\bar{G}^\top + \Delta_c$$

where

$$L = \begin{bmatrix} L_{t,t} & 0 & \cdots & 0 \\ L_{t+1,t} & L_{t+1,t+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & L_{t+n_p,t+n_p} \end{bmatrix}, \quad D = \begin{bmatrix} D_t & 0 & \cdots & 0 \\ 0 & D_{t+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & D_{t+n_p} \end{bmatrix}$$

Here, each diagonal block $L_{k,k}$ is lower triangular with unit diagonal, each D_k is diagonal, and each sub-diagonal block $L_{k,k-1}$ accounts for the coupling between consecutive stages. The block-tridiagonal structure of the Schur-complement matrix is therefore

$$\bar{G}\bar{H}^{-1}\bar{G}^\top + \Delta_c = \begin{bmatrix} S_{t,t} & S_{t+1,t}^\top & \cdots & 0 & 0 \\ S_{t+1,t} & S_{t+1,t+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & S_{t+n_p-1,t+n_p-1} & S_{t+n_p,t+n_p-1}^\top \\ 0 & 0 & \cdots & S_{t+n_p,t+n_p-1} & S_{t+n_p,t+n_p} \end{bmatrix} \quad (6.96)$$

with

$$\begin{aligned} S_{t,t} &= \bar{G}_{t,t}\bar{H}_t^{-1}\bar{G}_{t,t}^\top + \Delta_{c,t} \\ S_{k,k} &= \bar{G}_{k,k-1}\bar{H}_{k-1}^{-1}\bar{G}_{k,k-1}^\top + \bar{G}_{k,k}\bar{H}_k^{-1}\bar{G}_{k,k}^\top + \Delta_{c,k}, \quad k = t+1, \dots, t+n_p \\ S_{k,k-1} &= \bar{G}_{k,k-1}\bar{H}_{k-1}^{-1}\bar{G}_{k-1,k-1}^\top, \quad k = t+1, \dots, t+n_p \end{aligned}$$

Equating the corresponding blocks in $LDL^\top$ and the Schur-complement matrix gives,

$$\begin{aligned} L_{t,t}D_tL_{t,t}^\top &= S_{t,t} \\ L_{k,k-1}D_{k-1}L_{k-1,k-1}^\top &= S_{k,k-1}, \quad k = t+1, \dots, t+n_p \\ L_{k,k}D_kL_{k,k}^\top &= S_{k,k} - L_{k,k-1}D_{k-1}L_{k,k-1}^\top, \quad k = t+1, \dots, t+n_p \end{aligned} \quad (6.97)$$

Thus, $L_{t,t}$ and D_t are obtained by factorizing the initial diagonal block $S_{t,t}$. At each subsequent stage, $L_{k,k-1}$ is obtained from the second equation in (6.97), after which $L_{k,k}$ and D_k are obtained by factorizing the corrected diagonal block in the third equation. The multiplier-direction equation in (6.94) is solved by forward

substitution through L , diagonal solution through D , and backward substitution through $L^\top$. The primal-direction vector $d\chi$ is then recovered from the second equation in (6.95).

The above computation exploits the sparse finite-horizon structure of the Newton system. Closely related structure-exploiting solution methods have been developed for linear or convex quadratic MPC: Wang and Boyd [46] solve the resulting block-tridiagonal system using a specialized Cholesky factorization, while Rao, et. al. [54] obtain an efficient structured interior-point solution using block factorization and a Riccati recursion. In the present nonlinear formulation, the same finite-horizon sparsity principle is applied to the locally linearized primal-dual Newton system, while the $LDL^\top$ factorization is retained to support the inertia computation and regularization conditions developed in the following subsection.

6.5.3 Inertia Control and Regularization

Following the inertia-correction strategy used in IPOPT [49], the regularization terms introduced in Section 6.5.1 are selected so that the regularized Newton matrix K is nonsingular and possesses positive curvature in its primal block along directions satisfying the linearized equality constraints. These properties are ensured by imposing a target inertia with one positive eigenvalue for each primal direction, one negative eigenvalue for each equality-multiplier direction, and no zero eigenvalues [36], [49]. For the present formulation, the complete regularized Newton matrix is therefore required to satisfy

$$In(K) = \left((n_\chi + n_i + n_\epsilon)(n_p + 1), (n_\Lambda + n_i)(n_p + 1), 0 \right) \quad (6.98)$$

The elimination performed in Section 6.5.2 permits this target inertia to be verified using analytical contributions from the eliminated blocks and the stagewise factors already computed for the search direction.

Lemma 6.4: Inertia Verification for a Valid Newton Direction

Let n_χ , n_Λ , n_i , and n_ε denote the numbers of elements in $d\chi_k$, $d\Lambda_k$, $d\tilde{\lambda}_{i,k}$ (equivalently $d\tilde{s}_k$), and $d\varepsilon_k$, respectively. Suppose that, for $t \leq k \leq t + n_p$,

$$G_{si} = \text{diag}(\rho_i \odot \sigma_s^{-1})$$

is nonsingular,

$$H_{ss,k} > 0, \quad H_{\varepsilon\varepsilon,k} > 0, \quad \bar{H}_k \text{ is nonsingular,}$$

and the Schur-complement factorization introduced in Section 6.5.2 exists:

$$\bar{G}\bar{H}^{-1}\bar{G}^\top + \Delta_c = LDL^\top, \quad D = \text{blkdiag}(D_t, \dots, D_{t+n_p})$$

If the following is true,

$$\sum_{k=t}^{t+n_p} [In(\bar{H}_k) + In(-D_k)] = (n_\chi(n_p + 1), n_\Lambda(n_p + 1), 0) \quad (6.99)$$

then the complete regularized Newton matrix satisfies the target inertia in (6.98).

Proof:

Apply the Haynsworth inertia additivity formula (2.4) or (2.5) according to the elimination sequence used in Section 6.5.2. For each stage k , the first eliminated pivot block is associated with $d\tilde{s}_k$ and $d\tilde{\lambda}_{i,k}$:

$$\begin{bmatrix} H_{ss,k} & G_{si}^\top \\ G_{si} & 0 \end{bmatrix}$$

Since G_{si} is nonsingular, this block is nonsingular and has inertia

$$\text{In} \begin{pmatrix} H_{ss,k} & G_{si}^\top \\ G_{si} & 0 \end{pmatrix} = \text{In}(H_{ss,k}) + \text{In}(-G_{si}H_{ss,k}^{-1}G_{si}^\top) = (n_i, n_i, 0) \quad (6.100)$$

Following this elimination, the pivot block associated with $d\varepsilon_k$ is $H_{\varepsilon\varepsilon,k} > 0$. Hence,

$$\text{In}(H_{\varepsilon\varepsilon,k}) = (n_\varepsilon, 0, 0) \quad (6.101)$$

The matrix remaining is the condensed KKT matrix derived in Section 6.5.2:

$$\begin{bmatrix} \bar{H} & \bar{G}^\top \\ \bar{G} & -\Delta_c \end{bmatrix}.$$

Applying the Haynsworth inertia additivity formula to this matrix gives

$$\text{In} \begin{pmatrix} \bar{H} & \bar{G}^\top \\ \bar{G} & -\Delta_c \end{pmatrix} = \text{In}(\bar{H}) + \text{In}(-(\bar{G}\bar{H}^{-1}\bar{G}^\top + \Delta_c)) \quad (6.102)$$

Since $\bar{H}$ is block diagonal,

$$\text{In}(\bar{H}) = \sum_{k=t}^{t+n_p} \text{In}(\bar{H}_k) \quad (6.103)$$

Moreover, from the factorization in (6.97), Sylvester's law of inertia gives

$$\text{In}(-(\bar{G}\bar{H}^{-1}\bar{G}^\top + \Delta_c)) = \sum_{k=t}^{t+n_p} \text{In}(-D_k) \quad (6.104)$$

Therefore,

$$\text{In}(K) = \sum_{k=t}^{t+n_p} [\text{In}(\bar{H}_k) + \text{In}(-D_k)] + ((n_i + n_\varepsilon)(n_p + 1), n_i(n_p + 1), 0) \quad (6.105)$$

Substituting condition (6.99) into (6.105) yields the target inertia in (6.98).

The condition in Lemma 6.4 is evaluated using quantities already generated during the structure-exploiting factorization. If (6.99) is not satisfied, the regularization terms in Δ_h and Δ_c are increased and the factorization is repeated

until the target inertia in (6.99) is obtained. The resulting factors are then used to compute the regularized primal-dual Newton direction. Thus, the IPOPT inertia-correction principle [49] is applied through the stagewise structure-exploiting Newton formulation developed for the nonlinear MPC problem considered in this work.

For implementation of the regularization update, the nonzero stagewise primal regularization blocks are parameterized as $\Delta_{z,k} = \delta_h I$, $\Delta_{p,k} = \delta_h I$, $\delta_h \geq 0$, where the identity matrices have dimensions compatible. Similarly, when constraint regularization is required because of singularity in the Schur-complement factorization, the eligible constraint blocks in Δ_c are assigned a prescribed nominal diagonal perturbation. The following algorithm is written using this implementation form.

Algorithm 6.1: Fast Computation of Primal-Dual Search Direction

Inputs

- Current primal-dual iterate and barrier parameter μ
- Stagewise modified curvature blocks

$$H_{zz,k}, Q_{zp,k}, H_{pp,k}, H_{ss,k}, H_{ze,k}, H_{pe,k}, H_{se,k}, H_{\varepsilon\varepsilon,k}$$

- Stagewise residual vectors

$$r_{z,k}, r_{p,k}, r_{s,k}, r_{\varepsilon,k}, r_{\lambda_z,k}, r_{\lambda_p,k}, r_{\lambda_e,k}, r_{\lambda_i,k}$$

- Stagewise equality-constraint Jacobian blocks

$$G_{zi,k}, G_{pi,k}, G_{si}, G_{zz,k,k}, G_{zz,k+1,k}, G_{pp,k,k}, G_{pp,k+1,k}, G_{ze,k}, G_{pe,k}$$

- Previously accepted primal regularization magnitude $\delta_{h,\text{old}}$, regularization

parameters τ_{dec} , τ_{inc} , and $\delta_{h,\text{min}}$, and prescribed nominal constraint regularization Δ_c .

Outputs

- Accepted regularization matrices Δ_h and Δ_c
- Complete regularized primal-dual Newton direction for all $t \leq k \leq t + n_p$:

$$d\tilde{z}_k, d\tilde{p}_k, d\tilde{s}_k, d\varepsilon_k, d\tilde{\lambda}_{z,k}, d\tilde{\lambda}_{p,k}, d\tilde{\lambda}_{e,k}, d\tilde{\lambda}_{i,k}, d\tilde{\gamma}_{z,k}, d\tilde{\beta}_{z,k}, d\tilde{\gamma}_{p,k}, d\tilde{\beta}_{p,k}, d\tilde{\gamma}_{s,k}$$

Procedure

Step 1. Initialize the trial regularization terms

Initialize the scalar primal regularization from its most recently accepted value:

$$\delta_h \leftarrow \tau_{\text{dec}} \delta_{h,\text{old}}, \quad 0 \leq \tau_{\text{dec}} < 1$$

For each stage $t \leq k \leq t + n_p$, construct

$$\Delta_{z,k} = \delta_h I, \quad \Delta_{p,k} = \delta_h I$$

and assemble Δ_h . Set the constraint regularization to zero initially. The prescribed nominal value of Δ_c is activated only if singularity is encountered during the Schur-complement factorization.

Step 2. Assemble the condensed stagewise Newton quantities

For each stage $t \leq k \leq t + n_p$:

- Compute $\bar{H}_{z\varepsilon,k}$ and $\bar{H}_{p\varepsilon,k}$ using (6.80).
- Compute $\bar{r}_{\varepsilon,k}$ using (6.81).
- Compute the condensed curvature blocks $\bar{H}_{zz,k}, \bar{H}_{zp,k}, \bar{H}_{pp,k}$ using (6.83)
- Compute the condensed residual vectors $\bar{r}_{z,k}, \bar{r}_{p,k}$ using (6.84)

- Assemble $\bar{H}_k$ and $\bar{r}_k$ using (6.91).

Step 3. Factorize the stagewise condensed curvature blocks

For each stage $t \leq k \leq t + n_p$:

- Factorize $\bar{H}_k$ and determine $\text{In}(\bar{H}_k)$ using algorithm 2.1 at current Δ_h .
- If $\bar{H}_k$ is nonsingular, retain its factorization and $\text{In}(\bar{H}_k)$, increment k , and continue
- If $\bar{H}_k$ is singular, update the primal regularization magnitude as

$$\delta_h \leftarrow \begin{cases} \delta_{h,\min}, & \delta_h = 0, & \delta_{h,\text{old}} = 0, \\ \frac{\delta_{h,\text{old}}}{2}, & \delta_h = 0, & \delta_{h,\text{old}} > 0, & \tau_{\text{inc}} > 1. \\ \tau_{\text{inc}} \delta_h, & \delta_h > 0, \end{cases}$$

- After computing δ_h , set

$$\Delta_{z,k} = \delta_h I, \quad \Delta_{p,k} = \delta_h I, \quad t \leq k \leq t + n_p$$

assemble Δ_h , reset the stage counter $k = t$, and return to step 2

Step 4. Factorize the Schur-complement matrix and evaluate inertia

- Using $\bar{H}_k$, $\bar{G}_{k,k}$, and $\bar{G}_{k+1,k}$, compute the block $LDL^\top$ factorization and obtain the block-tridiagonal recursion in (6.96)– (6.97). Evaluate the target-inertia condition (6.99) in Lemma 6.4:
- If the factorization fails because of singularity, activate the prescribed nominal value of Δ_c and repeat the Schur-complement factorization. If condition (6.99) is not satisfied, update δ_h according to step 3, reconstruct Δ_h , reset the stage counter to $k = t$, and return to step 2.

Step 5. Form the condensed multiplier-equation right-hand side

Once the target inertia is satisfied, form the stagewise right-hand side of the multiplier-direction equation in the first relation of (6.94):

$$\{\nabla_{\Lambda} L - \bar{G} \bar{H}^{-1} \bar{r}\}_k = \begin{cases} \nabla_{\Lambda_t} L - \bar{G}_{t,t} \bar{H}_t^{-1} \bar{r}_t & , \quad k = t \\ \nabla_{\Lambda_k} L - \bar{G}_{k,k-1} \bar{H}_{k-1}^{-1} \bar{r}_{k-1} - \bar{G}_{k,k} \bar{H}_k^{-1} \bar{r}_k & , \quad t < k \leq t + n_p \end{cases}$$

Step 6. Compute the retained equality-multiplier directions

Solve the first relation in (6.94) using the accepted LDL^T factors.

- First determine the intermediate directions $d\bar{\Lambda}_k$ by forward substitution and diagonal solution,

$$\begin{aligned} & L_{k,k} D_k d\bar{\Lambda}_k \\ &= \begin{cases} \nabla_{\Lambda_t} L - \bar{G}_{t,t} \bar{H}_t^{-1} \bar{r}_t & , \quad k = t \\ \nabla_{\Lambda_k} L - \bar{G}_{k,k-1} \bar{H}_{k-1}^{-1} \bar{r}_{k-1} - \bar{G}_{k,k} \bar{H}_k^{-1} \bar{r}_k - L_{k,k-1} D_{k-1} d\bar{\Lambda}_{k-1} & , \quad t < k \leq t + n_p \end{cases} \end{aligned}$$

- Next determine $d\Lambda_k$ by backward substitution:

$$\begin{cases} L_{t+n_p, t+n_p}^T d\Lambda_{t+n_p} &= d\bar{\Lambda}_{t+n_p} & , \quad k = t + n_p \\ L_{k,k}^T d\Lambda_k &= d\bar{\Lambda}_k - L_{k+1,k}^T d\Lambda_{k+1} & , \quad k = t + n_p - 1, \dots, t \end{cases}$$

- Extract $d\tilde{\lambda}_{z,k}, d\tilde{\lambda}_{p,k}, d\tilde{\lambda}_{e,k}$ from $d\Lambda_k$ according to (6.90)

Step 7. Compute the retained primal directions

- Using the computed $d\Lambda_k$, recover $d\chi_k$ from the second relation in (6.95):

$$d\chi_k = \begin{cases} -\bar{H}_k^{-1} (\bar{r}_k + \bar{G}_{k,k}^T d\Lambda_k + \bar{G}_{k+1,k}^T d\Lambda_{k+1}) & , \quad t < k \leq t + n_p \\ -\bar{H}_{t+n_p}^{-1} (\bar{r}_{t+n_p} + \bar{G}_{t+n_p, t+n_p}^T d\Lambda_{t+n_p}) & , \quad k = t + n_p \end{cases}$$

- Extract $d\tilde{z}_k, d\tilde{p}_k$ from $d\chi_k$ according to (6.90)

Step 8. Recover the eliminated primal and equality-multiplier directions

For each stage $t \leq k \leq t + n_p$:

- Compute $d\varepsilon_k$ using (6.82).

- Compute $d\tilde{s}_k$ using (6.78).
- Compute $d\tilde{\lambda}_{i,k}$ using (6.79).

Step 9. Recover the bound-dual directions

For each stage $t \leq k \leq t + n_p$, compute $d\tilde{\gamma}_{z,k}$, $d\tilde{\beta}_{z,k}$, $d\tilde{\gamma}_{p,k}$, $d\tilde{\beta}_{p,k}$, $d\tilde{\gamma}_{s,k}$ using the eliminated perturbed-complementarity equations in (6.56).

The Hessian regularization strategy in Algorithm 6.1 is based on the inertia-correction principles described in Nocedal and Wright [36] and implemented in the IPOPT interior-point framework [49].

6.6 Globalization and Search-Direction Enhancement

The regularized primal–dual Newton direction computed in Section 6.5 is obtained from a local linearization of the perturbed optimality conditions and therefore requires globalization when a full step does not provide acceptable progress in the nonlinear barrier problem. The principal globalization mechanism used here is a filter line search, which treats reduction of the barrier objective and reduction of constraint violation as separate acceptance measures rather than combining them through a penalty parameter [51], [89], [90]. The present implementation follows the IPOPT filter line-search framework of Wächter and Biegler [49].

When an otherwise useful Newton step is rejected because of nonlinear equality-constraint curvature, a second-order correction (SOC) is attempted. This mechanism addresses the Maratos effect [91], where a step with good local optimality progress is rejected because constraint curvature causes an unexpected

increase in nonlinear constraint violation. The SOC follows the correction strategy developed for filter methods in [51], [92] and implemented in IPOPT [49]. If backtracking fails to identify an acceptable trial point, an additional KKT error-reduction check is applied before increasing the primal regularization.

Additionally, barrier parameter is reduced only after approximate convergence of the current fixed- μ_j subproblem; following each update, the filter is reinitialized because the barrier objective has changed [49].

6.6.1 Filter Line Search and Fraction-to-the-Boundary Steps

The filter acceptance test is based on the barrier objective and the violation of the equality constraints of the unscaled softened MPC problem. Although the Newton equations are solved in scaled variables, feasibility is evaluated in unscaled coordinates. Let $\tilde{v}$ denote the complete scaled primal vector and define

$$\mathcal{C}(\tilde{v}) \triangleq \text{vec} \left(c_{z,t}, \dots, c_{z,t+n_p}, c_{p,t}, \dots, c_{p,t+n_p}, c_{e,t}, \dots, c_{e,t+n_p}, c_{i,t} + s_t, \dots, c_{i,t+n_p} + s_{t+n_p} \right) \quad (6.106)$$

The scalar constraint-violation measure used by the filter is

$$\theta(\tilde{v}) \triangleq \|\mathcal{C}(\tilde{v})\| \quad (6.107)$$

For a fixed barrier parameter $\mu_j > 0$, the barrier objective associated with the softened-bound formulation is

$$\begin{aligned}
\phi_{\mu_j}(\tilde{v}) \triangleq & \sum_{k=t}^{t+n_p} \left[J_k + \frac{1}{2} \|\varepsilon_k\|_{Q_{\varepsilon,k}}^2 - \mu_j \sum_{m=1}^{n_x+n_u} (\log(\sigma_{z,m}^{-1} \tilde{z}_{l,k,m}) + \log(\sigma_{z,m}^{-1} \tilde{z}_{u,k,m})) \right. \\
& - \mu_j \sum_{m=1}^{n_\theta+1} (\log(\sigma_{p,m}^{-1} \tilde{p}_{l,k,m}) + \log(\sigma_{p,m}^{-1} \tilde{p}_{u,k,m})) \\
& \left. - \mu_j \sum_{m=1}^{n_s} \log(\sigma_{s,m}^{-1} \tilde{s}_{s,k,m}) \right] \quad (6.108)
\end{aligned}$$

Each filter entry is the pair $(\theta(\tilde{v}), \phi_{\mu_j}(\tilde{v}))$ representing equality-constraint violation and the objective of the current barrier subproblem, respectively. The acceptance conditions below follow the IPOPT filter procedure [49]. Before a trial point is tested against the filter, the step length must preserve strict interiority of the softened-bound residuals and their associated bound-dual variables. Let

$$dY_l = \text{vec}(d\tilde{v}_l, d\tilde{\lambda}_l, d\tilde{y}_l, d\tilde{\beta}_l)$$

denote the regularized primal-dual Newton direction at inner iteration l for the current barrier parameter μ_j . Let $\tilde{r}_l$ and $\tilde{\pi}_l$ denote the horizon-level softened-bound residual and bound-dual vectors defined in Section 6.4.2, with corresponding directions $d\tilde{r}_l$ and $d\tilde{\pi}_l$. Define fraction to boundary parameter as:

$$\tau_j := \max\{\tau_{\min}, 1 - \mu_j\}, \quad \tau_{\min} \in (0,1)$$

The primal fraction-to-the-boundary step length is

$$\alpha_l^{\max} = \max\{\alpha \in (0,1]: \tilde{r}_l + \alpha d\tilde{r}_l \geq (1 - \tau_j)\tilde{r}_l\} \quad (6.109)$$

The line search examines the decreasing sequence

$$\alpha_{l,r} = 2^{-r} \alpha_l^{\max}, r = 0, 1, 2, \dots, \quad (6.110)$$

with primal trial point

$$\tilde{v}_{l,r} = \tilde{v}_l + \alpha_{l,r} d\tilde{v}_l \quad (6.111)$$

The bound-dual variables are updated using the independent dual fraction-to-the-boundary step length

$$\alpha_l^{\text{dual}} = \max\{\alpha \in (0,1]: \tilde{\pi}_l + \alpha d\tilde{\pi}_l \geq (1 - \tau_j)\tilde{\pi}_l\} \quad (6.112)$$

As in [49], α_l^{dual} is used directly in the bound-dual update and is not reduced during primal filter backtracking.

Let $\mathcal{F}_\ell$ denote the current filter at inner iteration ℓ , whose entries are previously accepted pairs $(\theta_{\mathcal{F}}, \phi_{\mathcal{F}})$. The filter is initialized at the beginning of each fixed- μ_j barrier subproblem and reinitialized whenever μ_j is reduced. A primal trial point $\tilde{v}_{\ell,r}$ is acceptable to the filter if, for every $(\theta_{\mathcal{F}}, \phi_{\mathcal{F}}) \in \mathcal{F}_\ell$, at least one of the following conditions holds:

$$\theta(\tilde{v}_{\ell,r}) \leq (1 - \gamma_\theta)\theta_q \quad (6.113a)$$

$$\phi_{\mu_j}(\tilde{v}_{\ell,r}) \leq \phi_q - \gamma_\phi \theta_q \quad (6.113b)$$

where $\gamma_\theta, \gamma_\phi \in (0,1)$ are filter parameters. Conditions (6.113a)–(6.113b) determine whether the trial point is acceptable with respect to all existing filter entries.

In addition to filter acceptability, the trial point must satisfy a progress condition relative to the current iterate. Define the directional derivative of the barrier objective along the nominal primal Newton direction as

$$d\phi_l := \nabla_{\tilde{v}} \phi_{\mu_j}(\tilde{v}_l)^\top d\tilde{v}_l \quad (6.113c)$$

If the current iterate has sufficiently small constraint violation $\theta(\tilde{v}_l) \leq \theta_{\min}$ and the

switching condition

$$d\phi_l < 0 \quad \text{and} \quad \alpha_{l,r}(-d\phi_l)^{s_\phi} > \delta[\theta(\tilde{v}_l)]^{s_\theta} \quad (6.113d)$$

holds, where $\delta > 0$, $s_\theta > 1$, and $s_\phi \geq 1$, then the trial point is accepted only if it is acceptable to the filter and satisfies the Armijo condition

$$\phi_{\mu_j}(\tilde{v}_{l,r}) \leq \phi_{\mu_j}(\tilde{v}_l) + \eta_\phi \alpha_{l,r} d\phi_l, \quad \eta_\phi \in (0,0.5) \quad (6.113e)$$

Otherwise, a trial point that is acceptable to the filter is accepted only if

$$\theta(\tilde{v}_{l,r}) \leq (1 - \gamma_\theta)\theta(\tilde{v}_l) \quad \text{or} \quad \phi_{\mu_j}(\tilde{v}_{l,r}) \leq \phi_{\mu_j}(\tilde{v}_l) - \gamma_\phi \theta(\tilde{v}_l) \quad (6.113f)$$

Following the IPOPT filter line-search procedure [49], the filter is augmented after an accepted step unless both the switching condition in (6.113d) and the Armijo condition in (6.113e) hold. In the latter case, the accepted step is objective-driven and the filter is left unchanged. Once a primal trial point is accepted, the primal variables and equality multipliers are updated using the accepted backtracking step length $\alpha_{\ell,r}$, and the bound-dual variables are updated using $\alpha_\ell^{\text{dual}}$:

$$\begin{aligned} \tilde{v}_{\ell+1} &= \tilde{v}_\ell + \alpha_{\ell,r} d\tilde{v}_\ell \\ \tilde{\lambda}_{\ell+1} &= \tilde{\lambda}_\ell + \alpha_{\ell,r} d\tilde{\lambda}_\ell \\ \tilde{\gamma}_{\ell+1} &= \tilde{\gamma}_\ell + \alpha_\ell^{\text{dual}} d\tilde{\gamma}_\ell \\ \tilde{\beta}_{\ell+1} &= \tilde{\beta}_\ell + \alpha_\ell^{\text{dual}} d\tilde{\beta}_\ell \end{aligned} \quad (6.114)$$

If the initial primal trial point is rejected because it does not achieve the required reduction in the nonlinear equality-constraint residuals, the second-order correction described in the following subsection is attempted before continuing the primal backtracking line search. Thus, the filter controls acceptance of the primal step, while the bound-dual step is taken using its independently computed maximum admissible fraction-to-the-boundary value, consistent with [49].

6.6.2 Second-Order Correction

When the initial trial point generated by the selected Newton direction is rejected because of insufficient reduction in the nonlinear equality-constraint violation measure, a second-order correction is attempted. The SOC modifies the equality-constraint residual terms using their values at the rejected trial point and computes a corrected direction using the regularized factorization already obtained for the nominal Newton system. The correction is specialized to the equality-constraint structure and condensed Newton system of the present MPC formulation.

Let $d\tilde{v}_\ell$ denote the primal component of the regularized primal-dual Newton direction at inner iteration ℓ , and let $\alpha_{\ell,0}$ denote the initial admissible primal step length obtained from the fraction-to-the-boundary condition in (6.109). The first primal trial point is

$$\tilde{v}_{\ell,0} = \tilde{v}_\ell + \alpha_{\ell,0} d\tilde{v}_\ell \quad (6.115)$$

If this trial point is rejected by the filter because it does not provide sufficient reduction in the unscaled constraint-violation measure θ , a second-order correction is attempted. Define the unscaled SOC residual vector as

$$\mathcal{C}_\ell^{\text{soc}} = \alpha_{\ell,0} \mathcal{C}(\tilde{v}_\ell) + \mathcal{C}(\tilde{v}_{\ell,0}) \quad (6.116)$$

where $\mathcal{C}$ is the unscaled constraint-residual vector defined in (6.106). Partition $\mathcal{C}_\ell^{\text{soc}}$ into the stagewise corrected residual groups $c_{z,k}^{\text{soc}}$, $c_{p,k}^{\text{soc}}$, $c_{e,k}^{\text{soc}}$, and $(c_i + s)_k^{\text{soc}}$, corresponding to the four constraint groups in $\mathcal{C}$. The corrected residuals are transformed into the scaled Newton-system coordinates according to

$$\tilde{c}_{z,k}^{\text{soc}} := \rho_z \odot c_{z,k}^{\text{soc}}, \quad \tilde{c}_{p,k}^{\text{soc}} := \rho_p \odot c_{p,k}^{\text{soc}},$$

$$\tilde{c}_{e,k}^{\text{soc}} := \rho_e \odot c_{e,k}^{\text{soc}}, \quad \tilde{r}_{i,k}^{\text{soc}} := \rho_i \odot (c_i + s)_k^{\text{soc}}.$$

The equality-residual components of the right-hand side of (6.68)- (6.71) are then replaced, for all $t \leq k \leq t + n_p$, by

$$\begin{aligned} r_{\lambda_z,k} &\leftarrow \tilde{c}_{z,k}^{\text{soc}}, & r_{\lambda_p,k} &\leftarrow \tilde{c}_{p,k}^{\text{soc}} \\ r_{\lambda_e,k} &\leftarrow \tilde{c}_{e,k}^{\text{soc}}, & r_{\lambda_i,k} &\leftarrow \tilde{r}_{i,k}^{\text{soc}} \end{aligned} \tag{6.119}$$

Only the equality-constraint residuals are modified in the SOC computation. The stationarity and perturbed-complementarity residuals, as well as the regularized Newton matrix, remain unchanged. Consequently, the SOC direction is obtained by reusing the regularized factorization computed for the selected search direction, requiring only an additional right-hand-side solve [49], [51].

Let $d\tilde{v}_\ell^{\text{soc}}$ denote the primal component of the resulting SOC direction.

Reapplying the primal fraction-to-the-boundary rule in (6.109) gives the admissible corrected primal step length α_ℓ^{soc} , and the corresponding trial point is

$$\tilde{v}_\ell^{\text{soc}} = \tilde{v}_\ell + \alpha_\ell^{\text{soc}} d\tilde{v}_\ell^{\text{soc}} \tag{6.120}$$

The SOC trial point is evaluated using the acceptance conditions in (6.113), with the switching condition and Armijo right-hand side evaluated using the nominal direction $d\tilde{v}_\ell$ and the initial step length $\alpha_{l,0}$, consistent with [49]. If it is accepted, the primal variables and equality multipliers are updated using α_l^{soc} , while the bound-dual variables are updated using the independently computed maximum dual fraction-to-the-boundary step length for the SOC direction. If it is rejected, the nominal Newton direction is restored and the primal backtracking line search continues with a reduced step length.

Thus, the SOC procedure corrects the effect of nonlinear equality-constraint curvature without requiring an additional matrix factorization. For the present formulation, this requires correcting all equality-constraint groups, including the slack-augmented residual $c_i + s$, while retaining the scaling and condensation structure of the Newton solve.

6.6.3 Optimality Error and Barrier-Parameter Update

The barrier parameter is held fixed during the inner primal-dual iterations and is reduced only after the current barrier subproblem has been solved to a prescribed accuracy. This sequence-of-barrier-subproblems interpretation is standard in nonlinear interior-point methods [93]- [95]. In the present implementation, the specific optimality-error measures and barrier-parameter decrement strategy are adopted from IPOPT [49]. Following [49], convergence and barrier-update decisions are evaluated using the unscaled residuals of the softened MPC problem, rather than the scaled quantities. Let r_d denote the stacked unscaled stationarity residual of the Lagrangian and r_c^μ denote the stacked perturbed-complementarity residual corresponding to (6.22)– (6.26). The associated complementarity products are invariant under the primal–dual scaling and can therefore be used directly in the unscaled optimality measure. The optimality error for the barrier problem is defined as

$$E_\mu \triangleq \max \left\{ \frac{\| r_d \|_\infty}{s_d}, \quad \| \mathcal{C}(\tilde{v}) \|_\infty, \quad \frac{\| r_c^\mu \|_\infty}{s_c} \right\} \quad (6.121)$$

where $\mathcal{C}(\tilde{v})$ is the unscaled equality-constraint residual vector evaluated using the unscaled variables corresponding to $\tilde{v}$. Using the previously defined collections of

equality-constraint and softened-bound multipliers, the dual-scaling factors are

$$\begin{aligned} s_d &= \frac{\max\left\{s_{\max}, \frac{\|\lambda\|_1 + \|\gamma\|_1 + \|\beta\|_1}{n_\lambda + n_\gamma + n_\beta}\right\}}{s_{\max}} \\ s_c &= \frac{\max\left\{s_{\max}, \frac{\|\gamma\|_1 + \|\beta\|_1}{n_\gamma + n_\beta}\right\}}{s_{\max}} \end{aligned} \quad (6.122)$$

Here, $s_{\max} \geq 1$ is the prescribed dual-scaling threshold, and n_λ , n_γ , and n_β are the numbers of elements in the corresponding horizon-level multiplier vectors. The factors s_d and s_c prevent unusually large multiplier magnitudes from dominating the optimality test, consistent with the IPOPT stopping criterion [49].

The optimality error for the zero-barrier softened MPC problem is obtained by setting the barrier parameter to zero:

$$E_0 \triangleq E_\mu|_{\mu=0} \quad (6.123)$$

The overall optimization is terminated when E_0 satisfies the prescribed terminal tolerance. For a fixed barrier parameter $\mu_j > 0$, the inner primal-dual iterations are continued until E_{μ_j} satisfies the prescribed accuracy requirement for the current barrier subproblem. These termination and barrier-subproblem convergence tests are stated explicitly in the overall solution algorithm. Once the barrier-subproblem convergence test stated in the overall solution algorithm is satisfied, the barrier parameter is decreased using the update rule adopted from IPOPT [49]:

$$\mu_{j+1} = \max\left\{\frac{\epsilon_{tol}}{10}, \min\left(\kappa_\mu \mu_j, \mu_j^{\theta_\mu}\right)\right\}, \quad 0 < \kappa_\mu < 1, \quad 1 < \theta_\mu < 2 \quad (6.124)$$

Thus, each barrier subproblem is solved to an accuracy proportional to the current value of μ_j , while the barrier parameter is progressively reduced until the

stopping criterion for the zero-barrier softened MPC problem is satisfied. Since the barrier objective changes when μ_j is updated, the filter is reinitialized after each barrier-parameter decreases, consistent with [49].

6.6.4 Globalized Interior-Point Optimization Cycle

Algorithm 6.2 summarizes the complete optimization cycle after warm-start initialization and determination of the scaling factors. It incorporates the structure-exploiting Newton computation in Algorithm 6.1, the filter and SOC procedures in Sections 6.6.1–6.6.2, the barrier update in Section 6.6.3, and the KKT error-reduction check included in Algorithm 6.2.

Algorithm 6.2: Interior-Point Optimization Cycle for the Unified MPPFC

Problem

Inputs

- Initialized scaled strictly interior primal–dual point $Y_0 = \text{vec}(\tilde{v}_0, \tilde{\lambda}_0, \tilde{\gamma}_0, \tilde{\beta}_0)$
- Stage objective, constraint functions, constraint-softening weights, and required first- and second-order derivative functions for the softened MPPFC problem in (6.9)
- Fixed primal and constraint scaling vectors defined in (6.15)–(6.19)
- Initial barrier parameter $\mu_0 > 0$
- Initial or previously accepted primal regularization magnitude $\delta_{h,\text{old}}$
- Terminal tolerance ε_{tol} , barrier convergence tolerance ε_{μ_j} , and maximum allowable primal regularization threshold $\delta_{h,\text{max}}$
- Maximum number of optimization iterations l_{max}

- Algorithm parameters and tolerances used in Algorithm 6.1 and sections 6.6.1–6.6.3, together with the KKT error-reduction factor κ_{er} and maximum backtracking count r_{max}

Outputs

- Final scaled primal–dual point Y^* and corresponding unscaled softened MPPFC iterate
- Termination status: Converged, MaximumIterations, RegularizationLimit

Procedure

Step 1. Initialize the optimization cycle

- Set: $j \leftarrow 0$, $l \leftarrow 0$, $Y_l \leftarrow Y_0$, $\mu_j \leftarrow \mu_0$.
- Initialize the filter F_l for the barrier objective ϕ_{μ_j} as

$$F_l \leftarrow \{(\theta, \phi): \theta \geq \theta_{\text{max}}\}, \quad \theta_{\text{max}} > 0$$

where θ_{max} is the maximum admissible constraint violation for a trial point.

Step 2. Evaluate the current iterate and apply termination tests

- Evaluate the quantities required for the perturbed primal–dual conditions in (6.22)– (6.26) and (6.33)– (6.40), the Newton-system quantities in (6.64)– (6.71), and the optimality errors $E_{\mu_j}(Y_l)$ and $E_0(Y_l)$ using (6.121)– (6.123).

- Apply the following decision rule

$$\begin{array}{ll} \text{if} & E_0(Y_l) \leq \varepsilon_{\text{tol}} \quad \text{status} \leftarrow \text{Converged} \\ \text{elseif} & l \geq l_{\text{max}} \quad \text{status} \leftarrow \text{MaximumIterations} \\ \text{else} & \text{continue to step3} \end{array}$$

- If either termination condition is satisfied, set $Y^* \leftarrow Y_l$, recover the corresponding unscaled softened MPPFC iterate and terminate.

Step 3. Test convergence of the current barrier subproblem

- If $E_{\mu_j}(Y_l) \leq \varepsilon_{\mu_j}$, update the barrier parameter using (6.124), reinitialize the filter for the updated barrier objective $\phi_{\mu_{j+1}}$ as in step 1, set $j \leftarrow j + 1$, $\mu_j \leftarrow \mu_{j+1}$, and return to step 2 with Y_l unchanged.
- Otherwise, continue to step 4.

Step 4. Compute the nominal regularized Newton direction

- Apply Algorithm 6.1 at the current iterate Y_l and barrier parameter μ_j to compute the nominal regularized primal–dual Newton direction dY_l and the retained regularized factorization.
- If Algorithm 6.1 reaches the regularization threshold $\delta_h \geq \delta_{h,\max}$, set

$$Y^* \leftarrow Y_l, \quad \text{status} \leftarrow \text{RegularizationLimit}$$

recover the corresponding unscaled softened MPPFC iterate and terminate.

- Otherwise continue to step 5.

Step 5. Initialize the fraction-to-the-boundary step lengths and primal trial point

- Using the direction dY_l , compute the primal and bound-dual fraction-to-the-boundary step lengths $\alpha_l^{\max}$, α_l^{dual} from (6.109) and (6.112)
- Set $r \leftarrow 0$, $\alpha_{l,r} = 2^{-r} \alpha_l^{\max}$, and form the initial primal trial $\tilde{v}_{l,r} = \tilde{v}_l + \alpha_{l,r} d\tilde{v}_l$ point using (6.110)–(6.111).
- Continue to step 6.

Step 6. Apply the filter line-search acceptance test

- Evaluate the primal trial point $\tilde{v}_{l,r}$ using the filter acceptance conditions in (6.113a)- (6.113f).
- If the trial point satisfies the acceptance conditions, continue to step 10.
- If $r = 0$ and the initial trial point is rejected due to insufficient reduction in the equality-constraint violation measure θ , continue to step 7 to attempt a second-order correction.
- Otherwise, continue to step 8

Step 7. Attempt second-order correction

- Construct the SOC residual using (6.116), modify the right-hand side according to (6.119), and compute the SOC direction using the factorization retained from step 4.
- Form and evaluate the SOC trial point using (6.109), (6.112), (6.120), and the acceptance conditions in (6.113).
- If the SOC trial point satisfies the filter line-search acceptance conditions in (6.113), select the SOC direction and step lengths and continue to step 10.
- Else, retain the direction and step lengths from step 5 and continue to step 8.

Step 8. Backtrack the primal step

- If $r < r_{max}$, set, $r \leftarrow r + 1$ and update the primal trial point using (6.110)–(6.111), and return to step 6. Since $r > 0$, the SOC procedure in step 7 is not attempted again.
- If $r \geq r_{max}$, continue to step 9.

Step 9. Apply the KKT error-reduction phase

- After line-search failure, form a full primal–dual trial point Y_l^{trial} using the selected direction dY_l and its primal and bound-dual step lengths.
- Evaluate whether $E_{\mu_j}(Y_l^{\text{trial}}) \leq \kappa_{\text{er}} E_{\mu_j}(Y_l)$, $0 < \kappa_{\text{er}} < 1$, and whether its primal component is acceptable to the current filter according to (6.113a)–(6.113b).
- If both conditions are satisfied, select Y_l^{trial} and continue to Step 10.
- Otherwise, increase the primal regularization magnitude. If $\delta_h \geq \delta_{h,\text{max}}$, Set $Y^* \leftarrow Y_l$, status $\leftarrow$ RegularizationLimit, recover the corresponding unscaled softened MPPFC iterate and terminate.
- Otherwise, return to step 4 with the increased primal regularization.

Step 10. Accept a trial point and update the filter

- Update the primal–dual iterate using (6.114) with the accepted direction and step lengths.
- If the accepted step does not satisfy both the switching condition in (6.113d) and the Armijo condition in (6.113e), augment the filter

$$F_{l+1} \leftarrow F_l \cup \left\{ \left((1 - \gamma_\theta) \theta(\tilde{v}_l), \phi_{\mu_j}(\tilde{v}_l) - \gamma_\phi \theta(\tilde{v}_l) \right) \right\}$$

Otherwise, retain the current filter: $F_{l+1} \leftarrow F_l$.

- Set $l \leftarrow l + 1$, and return to step 2.

The filter line search, SOC procedure, and monotone barrier-parameter update in Algorithm 6.2 follow the applicable components of the IPOPT framework [49], with their theoretical foundations in [51], [89]–[92]. The KKT error-reduction

phase provides an additional safeguard when no acceptable filter line-search step is obtained. Warm-start generation and determination of the scaling factors are performed before Algorithm 6.2 and are therefore excluded from it.

6.7 Warm-Starting the Unified MPPFC Optimization Cycle

The Unified MPPFC problem is solved repeatedly over a receding prediction horizon. Consequently, two successive optimization problems normally differ only through the shifted horizon, updated measured state, and changes in the local constraint environment. Reusing the preceding primal–dual solution is therefore a natural initialization strategy for interior-point MPC [54], [96]- [98]. However, as noted in [54], a shifted solution may lie on or near the boundary of the feasible region and therefore must be modified before it is used as an interior-point initial iterate. More generally, warm-start studies for primal–dual interior-point methods show that a solution of a nearby perturbed problem can reduce computational effort when adequate interiority and centrality are restored before the new solve begins [99]- [102].

Let Y_t^* denote the accepted primal–dual solution at sampling instant t . For the subsequent optimization cycle, the shifted candidate is defined by

$$Y_{t+1}^{\text{sh}} = \mathcal{S}(Y_t^*) \quad (6.132)$$

where $\mathcal{S}(\cdot)$ shifts the stagewise primal and dual variables forward by one stage and appends terminal values for the newly introduced stage. No additional terminal-state rollout is performed during this append operation.

A shifted solution cannot generally be used directly in the barrier

formulation because previously active or nearly active bounds may place the new iterate on, or too close to, the logarithmic-barrier boundary. For the softened-bound residuals defined in (6.10)– (6.12), the warm start must satisfy

$$z_{l,k}^{ws}, z_{u,k}^{ws}, p_{l,k}^{ws}, p_{u,k}^{ws}, s_{l,k}^{ws} \geq \delta_{ws} > 0, \quad k = t, \dots, t + n_p \quad (6.133)$$

Accordingly, shifted components of z_k and p_k are pushed into the strict interior of their finite bounds. For a scalar bounded component $q_{k,j}$,

$$q_{k,j} + \delta_{q,j} \leq q_{k,j}^{ws} \leq \bar{q}_{k,j} - \delta_{q,j}, \quad \delta_{q,j} = \max \left\{ \delta_{abs}, \rho_q \min \left(d_{q,j}, \frac{\bar{q}_{k,j} - q_{k,j}}{2} \right) \right\} \quad (6.134)$$

For one-sided bounds, $\delta_{q,j}$ is computed from the prescribed physical scale $d_{q,j}$, avoiding excessive corrections associated with numerical surrogate bounds. The nonlinear inequality slack is initialized from the shifted primal candidate,

$$s_k^{ws} = -c_i(z_k^{ws}, p_k^{ws}) \quad (6.135)$$

and the relaxation variable ε_k^{ws} is adjusted, subject to $\varepsilon_k^{ws} \leq \varepsilon_{\max}$, to enforce the barrier-room condition in (6.133). If this limit is insufficient, the remaining correction is applied directly to the affected primal or slack component. This may introduce a nonzero initial residual in $c_i(z_k, p_k) + s_k = 0$, which is permissible because the warm start is an initial interior iterate rather than a feasible solution. This strict-interiority restoration is consistent with interior-point warm-start practice in MPC and related perturbed optimization sequences [54], [96]- [102].

The preceding barrier parameter and shifted dual variables are retained under normal shift-and-append operation because they carry useful continuation information from the previous optimization cycle [96]- [102]. This retention is

beneficial when the new problem is only moderately perturbed, but it may become ineffective when the active-constraint pattern changes substantially, such as when a new obstacle is encountered or a much larger inequality violation appears. In that case, the shifted point may lie too close to the barrier boundary, causing the fraction-to-the-boundary rule to permit only very small Newton steps [102].

Therefore, the present implementation requests a non-cold-start barrier reset only when the shifted candidate indicates a significantly more severe inequality episode than the recent warm-start history. To make this decision, the shifted candidate is evaluated using a scalar severity measure that combines nonlinear inequality violation and softened-bound barrier violation. The current severity measure is:

$$\begin{aligned} v_t &= \max\{v_t^c, v_t^b\}, \quad v_t^c = \max\left\{0, \max_k c_i(z_k^{sh}, p_k^{sh})\right\}, \\ v_t^b &= \max\left\{0, \max_{k,j} \{-z_{l,k,j}^{sh}, -z_{u,k,j}^{sh}, -p_{l,k,j}^{sh}, -p_{u,k,j}^{sh}\}\right\} \end{aligned} \quad (6.136)$$

Here, v_t^c measures violation of the nonlinear inequality constraints, while v_t^b measures violation of the shifted softened-bound barrier arguments. A reset is triggered only if

$$v_t > v_{t-1}^{mem} \quad \text{and} \quad \frac{v_t}{\max(v_{t-1}^{mem}, v_{min})} \geq r_v, \quad r_v > 1 \quad (6.137)$$

where the remembered severity level is updated using a decaying maximum. When (6.137) holds,

$$\mu^{ws} = \max\{\mu, \mu_{init}\} \quad (6.138)$$

the equality multipliers are reinitialized, and the positive bound multipliers are

recomputed from the restored barrier parameter and current barrier rooms.

Otherwise, μ and the shifted multipliers are retained, with positivity safeguards applied to the bound multipliers. Cold starting is used only for the first optimization cycle or after repeated hard solver failures, or an external initialization request.

Algorithm 6.3: Warm-Start Initialization for the Unified MPPFC Optimization Cycle

Inputs: Previous accepted solution Y_t^* ; current measured state and constraint data; scaling vectors; μ ; interiority, relaxation, severity-memory, and reset parameters.

Outputs: Initialized scaled strictly interior primal–dual point Y_0 ; initialized barrier parameter μ_0 .

Procedure

1. Shift Y_t^* forward by one stage and append terminal values as per (6.132).
2. Replace the initial-state components by the current measurement and apply the bound-push rule (6.134) to shifted z_k and p_k .
3. Initialize s_k^{ws} using (6.135), and adjust $\varepsilon_k^{\text{ws}}$, subject to $\varepsilon_{\max}$, until (6.133) is satisfied; apply any remaining room correction directly to the affected primal or slack component.
4. Evaluate v_t from (6.136) and test the new-episode condition (6.137).
5. If a reset is triggered, set μ_0 using (6.138), reinitialize equality multipliers, and recompute positive bound multipliers. Otherwise, retain $\mu_0 = \mu$ and the shifted multipliers with positivity safeguards.
6. Apply the fixed scaling transformations and return the initialized interior

point Y_0 to Algorithm 6.2.

The shift-and-interiority-restoration components of Algorithm 6.3 are grounded in established interior-point warm-start and MPC literature [54], [95]-[102]. The inequality-episode-based barrier-reset condition is an implementation-specific safeguard introduced in this work to avoid repeated resets during persistent constraint interactions while still responding to newly severe inequality episodes.

6.8 Scaling-Factor Selection

The primal and constraint scaling factors used in Algorithm 6.2 are computed before the interior-point optimization cycle begins. All quantities used for scaling are evaluated in the original physical coordinates. The current implementation uses a structural expected-box scaling for the primal variables and a Jacobian-row scaling for the equality and softened-inequality constraints. The distance-to-bound and gradient-based corrections to the primal-variable scaling are disabled in the present results. For each primal decision-variable component q_j , representing an element of either z_k or p_k , the effective expected range is chosen as the smaller of the user-defined expected range and the available box-bound range:

$$\sigma_{q,j} = \text{sat}_{[\sigma_{min}, \sigma_{max}]} \left(\frac{1}{\min \{ \Delta q_j^{user}, \bar{q}_j - \underline{q}_j \}} \right) \quad (6.139)$$

where $\text{sat}_{[\sigma_{min}, \sigma_{max}]}(\cdot)$ denotes clamping to the prescribed numerical range. Thus, variables with larger expected physical variation receive smaller scaling factors,

while tightly bounded or small-magnitude variables receive larger scaling factors.

After the primal scaling factors are fixed, the constraint-row scaling factors are computed from the scaled Jacobian row magnitudes. For a generic constraint row c_r , the row scaling is chosen in the form

$$\rho_r = \text{sat}_{[\rho_{min}, \rho_{max}]} \left(\frac{1}{\bar{m}_r} \right) \quad (6.140)$$

where $\bar{m}_r$ is row magnitude of the corresponding constraint Jacobian after applying the primal scaling factors. Rows whose magnitudes are below the prescribed absolute or relative activity thresholds are assigned neutral scaling, $\rho_r = 1$. This procedure is applied separately to the system-dynamics constraints, path-dynamics constraints, softened nonlinear inequality constraint rows, and user-defined equality constraints. The slack-variable scaling is set equal to the softened-inequality row scaling,

$$\sigma_s = \rho_i \quad (6.141)$$

so that the scaled slack contribution is consistent with the scaled softened-inequality residual. The resulting scaling factors are kept fixed during the subsequent interior-point optimization cycle.

6.9 Chapter Summary

This chapter presented the structure-exploiting primal–dual interior-point optimization framework used to solve the softened Unified MPPFC problem. The constrained MPPFC formulation was transformed into a softened barrier problem using slack and relaxation variables, and scaling factors were introduced to improve numerical conditioning while preserving the physical interpretation of the original

problem variables and constraints.

The perturbed primal–dual optimality conditions were derived in scaled form, including the stationarity, equality-feasibility, and complementarity equations required by the interior-point method. The resulting Newton system was then organized to exploit the stagewise structure of the prediction horizon. Bound-dual directions were eliminated, selected slack and relaxation-variable directions were condensed, and the remaining system was reduced to a structured form suitable for efficient horizon-wise computation. Regularization and inertia checks were included to obtain reliable search directions while preserving the sparse structure of the MPPFC problem.

Globalization was handled using a filter line search with fraction-to-the-boundary step selection, second-order correction, and monotone barrier-parameter updates. These mechanisms combine established interior-point globalization principles with formulation-specific safeguards for the softened Unified MPPFC problem. The chapter also specified the warm-start procedure, scaling-factor determination, and complete globalized optimization cycle used for repeated receding-horizon solution.

Overall, this chapter provides the computational foundation for the closed-loop Unified MPPFC implementation. It establishes how the softened collision-aware MPPFC problem is scaled, linearized, regularized, globalized, warm-started, and solved within a single structure-exploiting interior-point optimization cycle.

CHAPTER 7 - APPLICATION 1: TRACTOR-SEMITRAILER UNIFIED PATH FOLLOWING AND COLLISION AVOIDANCE

7.1 Introduction

This chapter presents a nonlinear dynamic model of an articulated tractor–semitrailer combination for unified path following and collision avoidance. The model captures the coupled longitudinal, lateral, yaw, and articulation dynamics required for full-vehicle simulation and nonlinear MPC. The formulation follows established approaches in articulated rigid-body vehicle modeling, including differential kinematics for nonholonomic motion and Newton–Euler equations for coupled multibody dynamics, consistent with Chen and Tomizuka [103], De Bernardis et al. [104], and Rajamani [105]. The resulting model is expressed in continuous-time nonlinear state-space form:

$$\dot{x} = f(x, u), \quad y = Cx \quad (7.1)$$

and is also written in the structured robotics form used throughout this dissertation:

$\begin{aligned} \dot{\rho} &= \Gamma(\rho)\xi \\ \dot{\xi} &= M^{-1}(\rho)(-(H(\rho, \xi) + E(\rho, \xi))\xi - K(\rho) + D + G(\rho, \xi)u) \end{aligned} \quad (7.2)$

Here, ρ denotes configuration variables, ξ denotes body-fixed velocity variables, $M(\rho)$ is the configuration-dependent positive-definite mass matrix, $H(\rho, \xi)$ collects inertial and coordinate-coupling terms, $E(\rho, \xi)\xi$ represents nonconservative velocity-dependent generalized forces, $K(\rho)$ represents conservative generalized forces, D represents external forcing terms not modeled as purely velocity proportional, and $G(\rho, \xi)u$ maps the control inputs into generalized forces. This

representation preserves the physical structure of the articulated vehicle model while matching the formulation required by the nonlinear MPC solver developed in Chapter 6.

7.2 States, Inputs, Frames, and Notation

The full system state is defined as

$$x = [X \ Y \ \varphi_t \ \varphi_r \ v_x \ v_y \ \psi_t \ \psi_r]^\top \in \mathbb{R}^8 \quad (7.3)$$

where X, Y are the global Cartesian coordinates of the tractor center of gravity in the road-fixed inertial frame, φ_t is the tractor yaw angle, φ_r is the articulation angle between the tractor and trailer, v_x and v_y are the longitudinal and lateral velocities of the tractor CG expressed in the tractor body-fixed frame, $\psi_t = \dot{\varphi}_t$ is the tractor yaw rate, and $\psi_r = \dot{\varphi}_r$ is the articulation rate. The system configuration and body-fixed velocity vectors are therefore

$$\rho = [X \ Y \ \varphi_t \ \varphi_r]^\top \in \mathbb{R}^4 \quad (7.4)$$

$$\xi = [v_x \ v_y \ \psi_t \ \psi_r]^\top \in \mathbb{R}^4 \quad (7.5)$$

With the sign convention used in this chapter, the trailer yaw angle is

$$\varphi_{trl} = \varphi_t + \varphi_r$$

and therefore, the trailer yaw rate is

$$\psi_{trl} = \dot{\varphi}_{trl} = \dot{\varphi}_t + \dot{\varphi}_r = \psi_t + \psi_r \quad (7.6)$$

The control vector is

$$u = [T_w \ \delta]^\top \in \mathbb{R}^2 \quad (7.7)$$

where T_w is the equivalent drive-torque applied on the powered axle(s) and δ is front-axle steering angle. These inputs represent the standard actuation channels for

articulated trucks in both control research and production systems [105].

The MPC controller for path following and collision avoidance requires full knowledge of pose and body-frame velocities; therefore,

$$y = Cx, \quad C = I_8 \quad (7.8)$$

This corresponds to measurements typically available from GPS/INS fusion, wheel-speed sensors, articulation-angle encoders, and IMU gyroscopes.

Rotation and 90° operator

The planar rotation matrix and the associated 90° rotation operation are defined as

$$R(\varphi) = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix}, \quad \Omega = R(\pi/2) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad (7.9)$$

Useful identities include

$$\frac{d}{d\varphi} R(\varphi) = R(\varphi)\Omega = \Omega R(\varphi) \quad (7.10)$$

which follows from the skew-symmetric generator for planar rotations.

Notation and Frame Conventions

The following notation is used consistently throughout this section:

- Frame 0: inertial (world-fixed) frame
- Frame 1: tractor body-fixed frame (origin at tractor CG)

For any vector quantity:

- subscripts identify the physical point or body
- superscripts identify the frame in which the vector is expressed:

The symbol p is used exclusively to denote position vectors and v to denote the

velocity vectors throughout this chapter. For example:

- $p_1^{(0)}$: position of the tractor CG expressed in inertial frame
- $p_2^{(1)}$: position of the trailer CG expressed in tractor frame
- $p_{2/1}^{(1)}$: position of trailer CG relative to tractor CG, expressed in tractor frame

Model Parameters

Symbol	Description	Units
l_1	Distance from tractor CG to front axle	m
l_2	Distance from tractor CG to equivalent rear-axle center	m
d_1	Distance from tractor CG to articulation joint	m
l_3	Distance from trailer CG to trailer axle	m
d_3	Distance from trailer CG to articulation joint	m
$d_{x,tr}$	Longitudinal offset (CG $\rightarrow$ tractor rectangle centroid)	m
$d_{y,tr}$	Lateral offset (CG $\rightarrow$ tractor rectangle centroid)	m
$d_{x,trl}$	Longitudinal offset (joint $\rightarrow$ trailer rectangle centroid)	m
$d_{y,trl}$	Lateral offset (joint $\rightarrow$ trailer rectangle centroid)	m
m_1	Tractor mass	kg
m_2	Trailer mass	kg
I_{z_1}	Tractor yaw moment of inertia	kg·m ²
I_{z_2}	Trailer yaw moment of inertia	kg·m ²
C_{af}	Front-axle tire cornering stiffness	N/rad
C_{ar}	Rear tractor axle cornering stiffness	N/rad
C_{at}	Trailer axle cornering stiffness	N/rad
C_a	Aerodynamic drag coefficient (linearized)	N·s/m
μ_0	Rolling resistance coefficient	—
g	Gravitational acceleration	m/s ²
α	Road grade angle	rad
β	Road bank angle	rad
k_{art}	Articulation spring constant	N·m/rad
C_{art}	Articulation damping constant (dissipative)	N·m·s/rad

Table 7.1: Geometric and Dynamic Parameters of Tractor-Trailer Model

7.3 Vehicle Geometry and Vehicle Kinematics

Vehicle Kinematics

The tractor CG velocity in the inertial frame is obtained by rotating the body-fixed velocity vector:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = R(\varphi_t) \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

By definition,

$$\dot{\varphi}_t = \psi_t, \quad \dot{\varphi}_r = \psi_r$$

Therefore, the full configuration kinematics can be written compactly as

$$\dot{\rho} = \Gamma(\rho)\xi, \quad \Gamma(\rho) = \begin{bmatrix} \cos \varphi_t & -\sin \varphi_t & 0 & 0 \\ \sin \varphi_t & \cos \varphi_t & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.11)$$

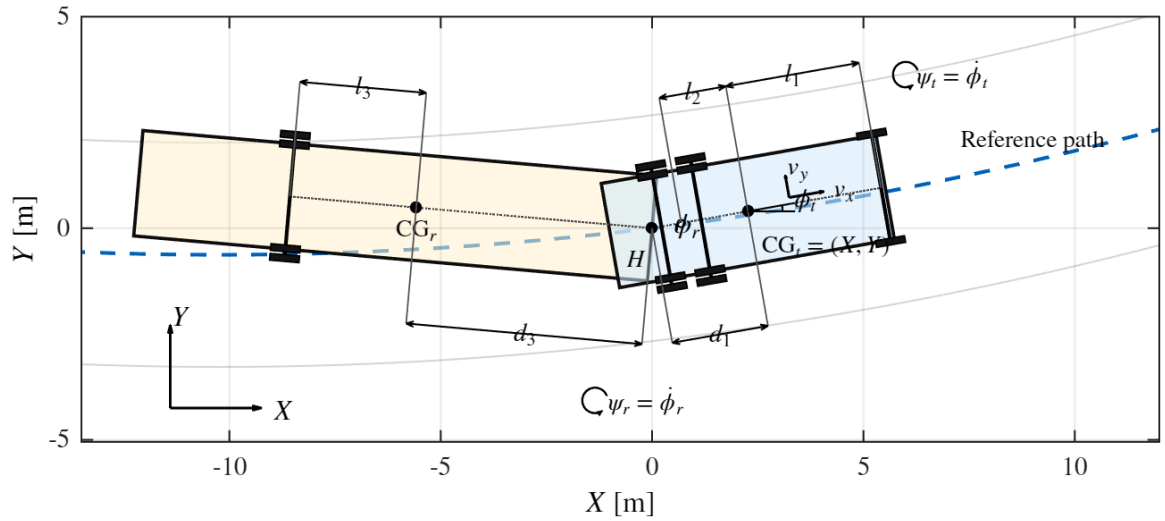


Figure 7.1: Tractor-Semitrailer Model and Geometric Parameters.

Trailer Position in the Tractor Frame

Let the hitch point be located at a distance $d_1 > 0$ behind the tractor CG along the tractor longitudinal axis. In the tractor frame:

$$p_{H/1}^{(1)} = \begin{bmatrix} -d_1 \\ 0 \end{bmatrix}$$

Let the trailer CG be located $d_3 > 0$ behind the hitch along the trailer longitudinal axis. Its position relative to the hitch, expressed in the trailer frame, is

$$p_{2/H}^{(2)} = \begin{bmatrix} -d_3 \\ 0 \end{bmatrix}$$

Expressing this vector in the tractor frame using the articulation angle,

$$p_{2/H}^{(1)} = R(\varphi_r) p_{2/H}^{(2)}$$

Hence, the position of the trailer CG relative to the tractor CG, expressed in tractor frame is

$$p_{2/1}^{(1)} = p_{H/1}^{(1)} + p_{2/H}^{(1)} \quad (7.12)$$

Hence,

$$p_{2/1}^{(1)} = \begin{bmatrix} -(d_1 + d_3 \cos \varphi_r) \\ -d_3 \sin \varphi_r \end{bmatrix} \quad (7.13)$$

Kinematic Relation in the Tractor Frame

The tractor frame is a rotating frame with angular velocity ψ_t about the vertical axis. For any point whose position relative to the tractor CG is given by $p^{(1)}$, the velocity of that point expressed in tractor frame is given by the standard rigid-body kinematic identity

$$v^{(1)} = v_{origin}^{(1)} + \psi_t \Omega p^{(1)} + \dot{p}^{(1)} \quad (7.14)$$

where:

- $v_{origin}^{(1)}$ is the velocity of the tractor CG expressed in the tractor frame
- $\dot{p}^{(1)}$ denotes the time derivative of the relative position taken in tractor frame

This identity accounts for translation of the tractor CG, rotation of the tractor frame,

and relative motion of the point within the rotating frame

Trailer Center-of-Gravity Velocity

The tractor CG velocity expressed in the tractor frame is

$$v_1^{(1)} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

Applying the kinematic identity (7.14) to the trailer CG with $p^{(1)} = p_{2/1}^{(1)}$ yields

$$v_2^{(1)} = v_1^{(1)} + \psi_t \Omega p_{2/1}^{(1)} + \dot{p}_{2/1}^{(1)} \quad (7.15)$$

The relative position $p_{2/1}^{(1)}$ depends on time only through the articulation angle.

Therefore

$$\dot{p}_{2/1}^{(1)} = \frac{\partial p_{2/1}^{(1)}}{\partial \varphi_r} \psi_r$$

Evaluating the required terms,

$$\Omega p_{2/1}^{(1)} = \begin{bmatrix} d_3 \sin \varphi_r \\ -(d_1 + d_3 \cos \varphi_r) \end{bmatrix}, \quad \frac{\partial p_{2/1}^{(1)}}{\partial \varphi_r} = \begin{bmatrix} d_3 \sin \varphi_r \\ -d_3 \cos \varphi_r \end{bmatrix}$$

Substituting into the velocity expression thus gives the trailer CG velocity components expressed in tractor frame as

$$\begin{aligned} v_{2x} &= v_x + d_3 \sin \varphi_r (\psi_t + \psi_r) \\ v_{2y} &= v_y - (d_1 + d_3 \cos \varphi_r) \psi_t - d_3 \cos \varphi_r \psi_r \end{aligned} \quad (7.16)$$

This resulting closed-form velocity expressions are directly compatible with the body-fixed state variables used in the dynamic model and form the basis for the subsequent energy-based derivation.

7.4 Energy-Based Vehicle Dynamics

This section derives the kinetic energy of the articulated tractor-semitrailer

system and constructs the configuration-dependent mass matrix appearing in the dynamics. All velocity variables are expressed in the tractor body-fixed frame, consistent with the state vector definition and with standard articulated vehicle formulations in [103]- [105].

Kinetic Energy of the Tractor-Semitrailer System

The total kinetic energy is the sum of translational and rotational kinetic energies of the tractor and the trailer:

$$T = T_1 + T_2$$

The tractor kinetic energy is

$$T_1 = \frac{1}{2} m_1 (v_x^2 + v_y^2) + \frac{1}{2} I_{z_1} \psi_t^2 \quad (7.17)$$

The trailer kinetic energy consists of translational kinetic energy and rotational kinetic energy about its yaw axis. Using the trailer CG velocity derived in previous section

$$T_2 = \frac{1}{2} m_2 \left(\left(v_{2x}^{(2)} \right)^2 + \left(v_{2y}^{(2)} \right)^2 \right) + \frac{1}{2} I_{z_2} (\psi_t + \psi_r)^2$$

where $v_{2x}^{(2)}$ and $v_{2y}^{(2)}$ are the trailer CG velocity components derived in the section (7.3). Since planar rotation preserves the Euclidean norm, the translational kinetic energy may be evaluated in either the tractor or trailer frame

$$T_2 = \frac{1}{2} m_2 (v_{2x}^2 + v_{2y}^2) + \frac{1}{2} I_{z_2} (\psi_t + \psi_r)^2 \quad (7.18)$$

Hence,

$$T = \frac{1}{2} m_1 (v_x^2 + v_y^2) + \frac{1}{2} I_{z_1} \psi_t^2 + \frac{1}{2} m_2 (v_{2x}^2 + v_{2y}^2) + \frac{1}{2} I_{z_2} (\psi_t + \psi_r)^2 \quad (7.19)$$

Potential Energy and Conservative Generalized Forces

The configuration vector ρ contains only planar position and orientation variables. Effects of road grade and bank angle are treated as exogenous road inputs and are incorporated directly as generalized forces in $E(\rho, \xi)$, rather than through an explicit configuration-dependent gravitational potential in ρ . In contrast, articulation compliance at the tractor-trailer hitch represents a conservative mechanical effect that can be naturally modeled through a potential energy function. We introduce the articulation potential

$$V_{artic}(\rho) = \frac{1}{2} k_{art} \varphi_r^2, \quad k_{art} > 0$$

which represents an equivalent torsional spring opposing articulation. Accordingly, the total potential energy of the system is

$$V(\rho) = V_{art}(\rho) = \frac{1}{2} k_{art} \varphi_r^2 \quad (7.20)$$

Lagrangian and Euler Lagrange Equation

Let $(p, \dot{p})$ denote generalized position and velocity variables, expressed using inertial (world) frame. The Lagrangian is defined as the difference between kinetic and potential energy [76]:

$$L(p, \dot{p}) \triangleq T(p, \dot{p}) - V(p) \quad (7.21)$$

where $T(p, \dot{p})$ denotes the total kinetic energy and $V(p)$ denotes the conservative potential energy. In the classical Euler–Lagrange formulation, the equations of motion are written in terms of the generalized coordinates p and their time derivatives $\dot{p}$ as follows [76]

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{p}} \right) - \frac{\partial L}{\partial p} = F_p \quad (7.22)$$

where F_p denotes the vector of generalized forces expressed in the inertial coordinate representation. As defined in chapter 2, for a scalar function $f(x)$, the gradient operator is defined by column stacking,

$$\nabla_x f = \left[\frac{\partial f}{\partial x_1} \quad \dots \quad \frac{\partial f}{\partial x_n} \right]^\top$$

Using this notation, (7.22) becomes

$$\frac{d}{dt} \nabla_{\dot{p}} L - \nabla_p L = F_p \quad (7.23)$$

For the tractor–semitrailer system, the total kinetic and potential energies are naturally expressed using the mixed coordinates (ρ, ξ) , where ρ is an inertial-frame configuration variable and ξ is a body-fixed velocity vector. To apply the Euler–Lagrange formulation, it is convenient to introduce an equivalent generalized-coordinate representation in inertial coordinates. To connect the energy expression written in the mixed variables (ρ, ξ) with the classical Euler–Lagrange form, an auxiliary inertial-coordinate representation is introduced. Let p denote the generalized coordinate vector in which the tractor and trailer yaw angles are represented as absolute angles. The coordinate transformation is defined as

$$p = A\rho, \quad \dot{p} = A\Gamma(\rho)\xi \quad (7.24)$$

where

$$A = \begin{bmatrix} I_2 & 0 \\ 0 & L_2 \end{bmatrix}, \quad L_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad \Gamma(\rho) = \begin{bmatrix} R(\varphi_t) & 0 \\ 0 & I_2 \end{bmatrix} \quad (7.25)$$

Here, A maps the angular coordinates (φ_t, φ_r) to the absolute yaw coordinates

$(\varphi_t, \varphi_t + \varphi_r)$, while $\Gamma(\rho)$ maps the body-fixed translational velocity components into inertial translational velocity components. Since A is nonsingular and $R(\phi_t)$ is orthogonal, the body-fixed velocity vector can be recovered from

$$\xi = \Gamma^{-1}(\rho)A^{-1}\dot{p} \quad (7.26)$$

Substituting this relation into the quadratic kinetic-energy form

$$T = \frac{1}{2}\xi^\top M(\rho)\xi$$

gives the equivalent inertial-coordinate kinetic energy

$$T(p, \dot{p}) = \frac{1}{2}\dot{p}^\top M_p(p)\dot{p} \quad (7.27)$$

where

$$M_p(p) = A^{-\top}\Gamma^{-\top}(\rho)M(\rho)\Gamma^{-1}(\rho)A^{-1}, \quad \rho = A^{-1}p \quad (7.28)$$

Applying the Euler–Lagrange equation in the auxiliary coordinates p , and transforming the result back to the body-fixed variables (ρ, ξ) , gives the implementation form used in this chapter,

$$\dot{\xi} = M^{-1}(\rho)\left(-\left(H(\rho, \xi) + E(\rho, \xi)\right)\xi - K(\rho) + D + G(\rho, \xi)u\right) \quad (7.29)$$

The term $H(\rho, \xi)\xi$ collects the velocity-dependent generalized force that arises from the Lagrangian transformation and is defined as,

$$H(\rho, \xi)\xi \triangleq (\dot{M} + \Gamma^\top \dot{\Gamma} M)\xi - (A\Gamma)^\top \nabla_p T \quad (7.30)$$

Mass Matrix and its Time Derivative

Because the kinetic energy is quadratic in the body-fixed velocity vector ξ ,

$T = \frac{1}{2}\xi^\top M(\varphi_r)\xi$. Let

$$s = \sin \varphi_r, \quad c = \cos \varphi_r \quad (7.31)$$

Then the mass matrix is

$$M(\varphi_r) = \begin{bmatrix} m_1 + m_2 & 0 & m_2 d_3 s & m_2 d_3 s \\ 0 & m_1 + m_2 & -m_2(d_1 + d_3 c) & -m_2 d_3 c \\ m_2 d_3 s & -m_2(d_1 + d_3 c) & \left(\begin{matrix} I_{z_1} + I_{z_2} + \\ m_2(d_1^2 + 2d_1 d_3 c + d_3^2) \end{matrix} \right) & I_{z_2} + m_2(d_3^2 + d_1 d_3 c) \\ m_2 d_3 s & -m_2 d_3 c & I_{z_2} + m_2(d_3^2 + d_1 d_3 c) & I_{z_2} + m_2 d_3^2 \end{bmatrix}$$

By construction, $M(\varphi_r)$ is symmetric and positive definite for physically meaningful parameter values. Since mass matrix depends on the configuration only through the articulation angle φ_r ,

$$\dot{M}(\rho) = \frac{\partial M}{\partial \varphi_r} \dot{\varphi}_r = \frac{\partial M}{\partial \varphi_r} \psi_r \quad (7.32)$$

Substituting the derivative entries of M gives

$$\dot{M}(\varphi_r, \psi_r) = m_2 d_3 \psi_r \begin{bmatrix} 0 & 0 & c & c \\ 0 & 0 & s & s \\ c & s & -2d_1 s & -d_1 s \\ c & s & -d_1 s & 0 \end{bmatrix} \quad (7.33)$$

Simplification of $\Gamma^\top \dot{M}$

For the kinematic mapping used in this work,

$$\Gamma(\varphi_t) = \text{diag}(R(\varphi_t), I_2) \quad (7.34)$$

Differentiating the above with respect to time

$$\dot{\Gamma} = \text{diag}\left(\psi_t \frac{\partial R}{\partial \varphi_t}, 0\right) \quad (7.35)$$

Since $\frac{\partial R}{\partial \varphi_t} = R(\varphi_t)\Omega$,

$$\dot{\Gamma} = \psi_t \text{diag}(R\Omega, 0), \quad \Gamma^\top \dot{\Gamma} = \psi_t \text{diag}(\Omega, 0) \quad (7.36)$$

Using this, the product $\Gamma^\top \dot{M}$ becomes

$$\Gamma^\top \dot{\Gamma} M = \psi_t \begin{bmatrix} 0 & -(m_1 + m_2) & m_2(d_1 + d_3c) & m_2d_3c \\ (m_1 + m_2) & 0 & m_2d_3s & m_2d_3s \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (7.37)$$

Evaluation of $(A\Gamma)^\top \nabla_p T$

The kinetic-energy gradient contributes additional velocity-dependent generalized forces because the mass matrix depends on the articulation angle and because the velocity variables are expressed in the rotating tractor frame. Applying the transformation $(A\Gamma)^\top$ to the generalized-coordinate gradient gives

$$(A\Gamma)^\top \nabla_p T = m_2 \begin{bmatrix} 0 \\ 0 \\ (d_1v_x + d_3(cv_x + sv_y))\psi_t + d_3(cv_x + sv_y)\psi_r \\ d_3((cv_x + sv_y) - d_1s\psi_t)(\psi_t + \psi_r) \end{bmatrix}$$

Equivalently, this contribution can be written in matrix form as

$$(A\Gamma)^\top \nabla_p T = m_2 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d_1v_x + d_3(cv_x + sv_y) & d_3(cv_x + sv_y) \\ 0 & 0 & d_3((cv_x + sv_y) - d_1s\psi_t) & d_3((cv_x + sv_y) - d_1s\psi_t) \end{bmatrix} \xi \quad (7.38)$$

Using (7.37), (7.38) and (7.30), $H(\rho, \xi)$ can be expressed as

$$H(\rho, \xi) = \begin{bmatrix} 0 & -(m_1 + m_2)\psi_t & m_2 \begin{pmatrix} (d_1 + d_3c)\psi_t \\ +d_3c\psi_r \end{pmatrix} & m_2d_3c(\psi_t + \psi_r) \\ (m_1 + m_2)\psi_t & 0 & m_2d_3s(\psi_t + \psi_r) & m_2d_3s(\psi_t + \psi_r) \\ m_2d_3c\psi_r & m_2d_3s\psi_r & -m_2 \begin{pmatrix} d_1(v_x + 2d_3s\psi_r) \\ +d_3(cv_x + sv_y) \end{pmatrix} & -m_2d_3 \begin{pmatrix} cv_x + sv_y \\ +d_1s\psi_r \end{pmatrix} \\ m_2d_3c\psi_r & m_2d_3s\psi_r & -m_2d_3 \begin{pmatrix} cv_x + sv_y \\ -d_1s(\psi_t - \psi_r) \end{pmatrix} & -m_2d_3 \begin{pmatrix} cv_x + sv_y \\ -d_1s\psi_t \end{pmatrix} \end{bmatrix} \quad (7.39)$$

This representation is not unique, but it is consistent with (7.30) and is convenient for implementation in the robotics-style dynamic model.

Explicit evaluation of $(A\Gamma)^\top \nabla_p V$

The potential energy of the system, from (7.20) is

$$V(\rho) = \frac{1}{2} k_{art} \varphi_r^2$$

Converting this into the inertial frame coordinates p ,

$$V(p) = \frac{1}{2} k_{art} (\varphi_{trl} - \varphi_t)^2, \quad \nabla_p V = k_{art} \varphi_r [0 \quad 0 \quad -1 \quad 1]^\top$$

Therefore,

$$(A\Gamma)^\top \nabla_p V = K(\rho) = [0 \quad 0 \quad 0 \quad k_{art} \varphi_r]^\top \quad (7.40)$$

7.5 Generalized Forces and Input mapping

All non-inertial and non-conservative effects are collected as generalized forces and mapped into either the input term $B(\rho, \xi)u$ or the non-input term $E(\rho, \xi)\xi + D$. In this model, the effects of the following are included: drive traction, steering-generated lateral tire forces, tire restoring and damping forces, aerodynamic drag, rolling resistance, grade and bank components, and articulation damping forces. Accordingly, the generalized force vector is decomposed as

$$F_\rho(\rho, \xi, u) = F_{tire}(\rho, \xi) + F_{aero}(\xi) + F_{artic}(\rho, \xi) + F_{roll}(\rho, \xi) + F_{grade/bank} \quad (7.41)$$

Linear Tire Model

To derive the lateral tire forces, a linear tire model is used [105]:

$$F_y = -C_\alpha \alpha \quad (7.42)$$

where C_α is the lateral cornering stiffness of the tire and α is the slip angle. Under

the small-angle assumption, the slip angle is approximated

$$\alpha \approx \frac{v_{lat}}{v_{long}} \quad (7.43)$$

Lateral Forces on Tractor Axles

For the front axle, the lateral velocity in tractor frame is $v_y + l_1\psi_t$, and the longitudinal velocity is v_x . Including the steering angle δ , the front slip angle and the lateral tire force is

$$\alpha_f \approx \frac{v_y + l_1\psi_t}{v_x} - \delta, \quad F_{yf} = C_{\alpha f} \left(\delta - \frac{v_y + l_1\psi_t}{v_x} \right) \quad (7.44)$$

The rear axle is not steered. Its lateral velocity in the tractor frame is $v_y - l_2\psi_t$,

hence

$$\alpha_r \approx \frac{v_y - l_2\psi_t}{v_x}, \quad F_{yr} = -C_{\alpha r} \left(\frac{v_y - l_2\psi_t}{v_x} \right) \quad (7.45)$$

Lateral Forces on Trailer Axles

From Section 7.3, the trailer CG velocity expressed in the tractor frame is

$$\begin{aligned} v_{2x}^{(1)} &= v_{2x} = v_x + d_3 \sin \varphi_r (\psi_t + \psi_r) \\ v_{2y}^{(1)} &= v_{2y} = v_y - (d_1 + d_3 \cos \varphi_r) \psi_t - d_3 \cos \varphi_r \psi_r \end{aligned}$$

To form the trailer slip angle, this velocity must be expressed in the trailer body-fixed frame. Using

$$v_2^{(2)} = R(-\varphi_r) v_2^{(1)}$$

one obtains

$$\begin{aligned} v_{2x}^{(2)} &= \cos \varphi_r v_x + \sin \varphi_r (v_y - d_1\psi_t) \\ v_{2y}^{(2)} &= -\sin \varphi_r v_x + \cos \varphi_r (v_y - d_1\psi_t) - d_3(\psi_t + \psi_r) \end{aligned} \quad (7.46)$$

The trailer axle slip angle, formed at the axle contact point in the trailer frame, is

$$\alpha_t \approx \frac{v_{2y}^{(2)} - l_3(\psi_t + \psi_r)}{v_{2x}^{(2)}}$$

And therefore,

$$\alpha_t \approx \frac{-\sin \varphi_r v_x + \cos \varphi_r (v_y - d_1 \psi_t) - (d_3 + l_3)(\psi_t + \psi_r)}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} \quad (7.47)$$

$$F_{yt} = -C_{at} \left(\frac{-\sin \varphi_r v_x + \cos \varphi_r (v_y - d_1 \psi_t) - (d_3 + l_3)(\psi_t + \psi_r)}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} \right) \quad (7.48)$$

Longitudinal generalized forces

The tractor-frame longitudinal component induced by the trailer lateral force is

$$F_{x,hitch} = -\sin \varphi_r F_{yt}$$

Therefore, the total generalized longitudinal force can be written as

$$F_x = E_{x,tire}(\rho, \xi)\xi + \frac{T_w}{r_w} \quad (7.49)$$

where

$$\begin{aligned} E_{x,tire}(\rho, \xi)\xi = & - \left(\frac{\sin^2 \varphi_r}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) v_x \\ & + \frac{1}{v_x} \left(\frac{\cos \varphi_r \sin \varphi_r v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) v_y \\ & - \frac{1}{v_x} \left(\frac{(d_1 \cos \varphi_r + l_3 + d_3) \sin \varphi_r v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) \psi_t \\ & - \frac{1}{v_x} \left(\frac{(l_3 + d_3) \sin \varphi_r v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) \psi_r \end{aligned} \quad (7.50)$$

Lateral generalized forces

The total lateral generalized force acting on the tractor is

$$F_y = F_{yf} + F_{yr} + \cos \varphi_r F_{yt}$$

Hence

$$F_y = E_{y,tire}(\rho, \xi)\xi + C_{af}\delta \quad (7.51)$$

where

$$\begin{aligned} E_{y,tire}(\rho, \xi)\xi = & \left(\frac{\cos \varphi_r \sin \varphi_r}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) v_x \\ & - \frac{1}{v_x} \left(C_{af} + C_{ar} + \frac{\cos^2 \varphi_r v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) v_y \\ & - \frac{1}{v_x} \left(l_1 C_{af} - l_2 C_{ar} - \frac{(d_1 \cos \varphi_r + l_3 + d_3) \cos \varphi_r v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) \psi_t \\ & + \frac{1}{v_x} \left(\frac{(l_3 + d_3) \cos \varphi_r v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) \psi_r \end{aligned} \quad (7.52)$$

Tractor Yaw Moment about the Tractor CG

Since $F_{2y}^{(2)} = F_{yt}$ is lateral tire force of trailer in trailer frame, the trailer axle

force expressed in the tractor frame is

$$F_2^{(1)} = \begin{bmatrix} -\sin \varphi_r F_{yt} \\ \cos \varphi_r F_{yt} \end{bmatrix}$$

In vector form, the tractor CG to the trailer axle, expressed in the tractor frame is

$$p_{t/1}^{(1)} = \begin{bmatrix} p_{x,t/1}^{(1)} \\ p_{y,t/1}^{(1)} \end{bmatrix} = \begin{bmatrix} -(d_1 + (d_3 + l_3) \cos \varphi_r) \\ -(d_3 + l_3) \sin \varphi_r \end{bmatrix}$$

Therefore, the moment contribution or the trailer axle force about the tractor CG is:

$$M_{2z}^{(1)} = p_{x,t/1}^{(1)} F_{2y}^{(1)} - p_{y,t/1}^{(1)} F_{2x}^{(1)}$$

Substituting and simplifying the above

$$M_{2z}^{(1)} = -(d_1 \cos \varphi_r + d_3 + l_3) F_{yt} \quad (7.53)$$

Therefore, the full tractor yaw moment:

$$M_z = l_1 F_{yf} - l_2 F_{yr} - (d_1 \cos \varphi_r + d_3 + l_3) F_{yt}$$

Hence,

$$M_z = E_{z,tire}(\rho, \xi) \xi + l_1 C_{\alpha f} \delta \quad (7.54)$$

where

$$\begin{aligned} E_{z,tire}(\rho, \xi) \xi = & - \left(\frac{(d_1 \cos \varphi_r + d_3 + l_3) \sin \varphi_r C_{\alpha t}}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} \right) v_x \\ & - \frac{1}{v_x} \left(l_1 C_{\alpha f} - l_2 C_{\alpha r} - \frac{(d_1 \cos \varphi_r + d_3 + l_3) \cos \varphi_r v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{\alpha t} \right) v_y \\ & - \frac{1}{v_x} \left(l_1^2 C_{\alpha f} + l_2^2 C_{\alpha r} + \frac{(d_1 \cos \varphi_r + d_3 + l_3)^2 v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{\alpha t} \right) \psi_t \\ & - \frac{1}{v_x} \left(\frac{(d_3 + l_3)(d_1 \cos \varphi_r + d_3 + l_3) v_x}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{\alpha t} \right) \psi_r \end{aligned} \quad (7.55)$$

Articulation Generalized Moment

The vector from the hitch to the trailer axle, expressed in the trailer frame, is

$$p_{t/h}^{(2)} = \begin{bmatrix} p_{x,t/h}^{(2)} \\ p_{y,t/h}^{(2)} \end{bmatrix} = \begin{bmatrix} -(d_3 + l_3) \\ 0 \end{bmatrix}$$

Because this vector lies along the trailer longitudinal axis, the same vector expressed in the tractor frame is

$$p_{t/h}^{(1)} = R(\varphi_r) p_{t/h}^{(2)} = -(d_3 + l_3) \begin{bmatrix} \cos \varphi_r \\ \sin \varphi_r \end{bmatrix}$$

The trailer axle lateral force in the trailer frame is

$$F_2^{(2)} = \begin{bmatrix} F_{2x}^{(2)} \\ F_{2y}^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ F_{2y}^{(2)} \end{bmatrix}$$

Expressed in the tractor frame, this force becomes

$$F_2^{(1)} = R(\varphi_r)F_2^{(2)} = F_{2y}^{(2)} \begin{bmatrix} -\sin \varphi_r \\ \cos \varphi_r \end{bmatrix}$$

For a planar force, the moment about the hitch is given by the two-dimensional cross product. Using the trailer-frame representation, the articulation moment is

$$M_{2z}^{(h)} = p_{x,t/h}^{(2)} F_{2y}^{(2)} - p_{y,t/h}^{(2)} F_{2x}^{(2)}$$

Since $p_{y,t/h}^{(2)} = 0$ and $F_{2x}^{(2)} = 0$, it follows directly that

$$M_{art} = M_{2z}^{(h)} = E_{art,tire}(\rho, \xi)\xi \quad (7.56)$$

with

$$\begin{aligned} E_{art,tire}(\rho, \xi)\xi = & - \left(\frac{(d_3 + l_3) \sin \varphi_r}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) v_x \\ & + \left(\frac{(d_3 + l_3) \cos \varphi_r}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) v_y \\ & - \left(\frac{(d_3 + l_3)(d_1 \cos \varphi_r + d_3 + l_3)}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) \psi_t \\ & - \left(\frac{(d_3 + l_3)^2}{\cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t)} C_{at} \right) \psi_r \end{aligned} \quad (7.57)$$

Tire Force Vector and Input Mapping

$$F_{tire}(\rho, \xi) = E_{tire}(\rho, \xi)\xi + Gu$$

where

$$E_{tire}(\rho, \xi)\xi = \begin{bmatrix} E_{x,tire}\xi \\ E_{y,tire}\xi \\ E_{z,tire}\xi \\ E_{art,tire}\xi \end{bmatrix}, \quad G(\rho, \xi) = \begin{bmatrix} \frac{1}{r_w} & 0 \\ 0 & C_{\alpha f} \\ 0 & l_1 C_{\alpha f} \\ 0 & 0 \end{bmatrix} \quad (7.58)$$

External Forces

Using road grade α , bank angle β , aerodynamic drag coefficient C_a , rolling

resistance coefficient μ_0 , and the articulation damping coefficient C_{art} , the remaining external generalized forces are modeled as

$$F_{grade/bank} = (m_1 + m_2)g \begin{bmatrix} \sin \alpha \\ \sin \beta \\ 0 \\ 0 \end{bmatrix}, \quad F_{aero}(\rho, \xi) = - \begin{bmatrix} C_a v_x |v_x| \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$F_{roll}(\rho, \xi) = -(m_1 + m_2)g\mu_0 \cos \alpha \operatorname{sign}(v_x) \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad F_{artic}(\rho, \xi) = - \begin{bmatrix} 0 \\ 0 \\ 0 \\ C_{art}\psi_r \end{bmatrix}$$

For the forward-driving case $v_x \geq 0$ these terms may be grouped into

$E(\rho, \xi)\xi + D$, with

$$E(\rho, \xi)\xi = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{12} & e_{22} & e_{23} & e_{24} \\ e_{13} & e_{23} & e_{33} & e_{34} \\ e_{14} & e_{24} & e_{34} & e_{44} \end{bmatrix} \quad (7.59)$$

$$D = (m_1 + m_2)g \begin{bmatrix} \sin \alpha - \mu_0 \cos \alpha \\ \sin \beta \\ 0 \\ 0 \end{bmatrix} \quad (7.60)$$

where

$$\begin{aligned} e_{11} &= - \left(C_a v_x + \frac{s^2 C_{at}}{v_{2x}^{(2)}} \right), & e_{22} &= - \left(\frac{C_{af} + C_{ar}}{v_x} + \frac{c^2 C_{at}}{v_{2x}^{(2)}} \right), \\ e_{33} &= - \left(\frac{l_1^2 C_{af} + l_2^2 C_{ar}}{v_x} + \frac{l_{eff}^2 C_{at}}{v_{2x}^{(2)}} \right), & e_{44} &= - \left(\frac{l_t^2 C_{at}}{v_{2x}^{(2)}} + C_{art} \right) \\ e_{12} &= \frac{cs C_{at}}{v_{2x}^{(2)}}, & e_{13} &= - \frac{l_{eff} s v_x C_{at}}{v_{2x}^{(2)}}, & e_{14} &= - \frac{l_t s v_x C_{at}}{v_{2x}^{(2)}} \\ e_{23} &= - \left(\frac{l_1 C_{af} - l_2 C_{ar}}{v_x} - \frac{l_{eff} c C_{at}}{v_{2x}^{(2)}} \right), & e_{24} &= \frac{l_t c C_{at}}{v_{2x}^{(2)}}, & e_{34} &= - \frac{l_t l_{eff} C_{at}}{v_{2x}^{(2)}} \end{aligned}$$

$$v_{2x}^{(2)} = \cos \varphi_r v_x + \sin \varphi_r (v_y - d_1 \psi_t), \quad l_{eff} = d_1 \cos \varphi_r + d_3 + l_3,$$

$$l_t = d_3 + l_3$$

Small Angle Approximations

Under small angle (using $\sin \varphi_r \approx 0$, $\cos \varphi_r \approx 1$) and forward motion ($v_x \geq 0$), the expressions derived in the previous sections admit below simplified forms:

$$E(\xi) \approx \begin{bmatrix} C_a v_x & 0 & 0 & 0 \\ 0 & -\frac{1}{v_x} C_{eff} & -\frac{1}{v_x} \begin{pmatrix} l_1 C_{af} - l_2 C_{ar} - \\ l_{eff} C_{at} \end{pmatrix} & \frac{1}{v_x} l_t C_{at} \\ 0 & -\frac{1}{v_x} \begin{pmatrix} l_1 C_{af} - l_2 C_{ar} - \\ l_{eff} C_{at} \end{pmatrix} & -\frac{1}{v_x} \begin{pmatrix} l_1^2 C_{af} + l_2^2 C_{ar} + \\ l_{eff}^2 C_{at} \end{pmatrix} & -\frac{1}{v_x} (l_{eff} l_t C_{at}) \\ 0 & \frac{1}{v_x} l_t C_{at} & -\frac{1}{v_x} (l_{eff} l_t C_{at}) & -\left(\frac{1}{v_x} l_t^2 C_{at} + C_{art} \right) \end{bmatrix} \quad (7.61)$$

$$M \approx \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m & -m_2(d_1 + d_3) & -m_2 d_3 \\ 0 & -m_2(d_1 + d_3) & I_{z_1} + I_{z_2} + m_2(d_1 + d_3)^2 & I_{z_2} + m_2(d_3^2 + d_1 d_3) \\ 0 & -m_2 d_3 & I_{z_2} + m_2(d_3^2 + d_1 d_3) & I_{z_2} + m_2 d_3^2 \end{bmatrix} \quad (7.62)$$

$$H(\xi) \approx \begin{bmatrix} 0 & -m \psi_t & m_2((d_1 + d_3) \psi_t + d_3 \psi_r) & m_2 d_3 (\psi_t + \psi_r) \\ m \psi_t & 0 & 0 & 0 \\ m_2 d_3 \psi_r & 0 & -m_2(d_1 + d_3) v_x & -m_2 d_3 v_x \\ m_2 d_3 \psi_r & 0 & -m_2 d_3 v_x & -m_2 d_3 v_x \end{bmatrix} \quad (7.63)$$

where

$$l_{eff} = d_1 + d_3 + l_3, \quad C_{eff} = C_{af} + C_{ar} + C_{at}, \quad l_t = d_3 + l_3, \quad m = m_1 + m_2$$

7.6 Rigid-Body Transformations for Collision Modeling

Accurate collision-avoidance modeling requires the poses of the geometric centroids of both the tractor and trailer bodies. The dynamic model is expressed

using the tractor CG pose and the articulation-joint pose. Therefore, fixed rigid-body offsets are used to map these dynamic-model frames to the centroid frames of the rectangles used for collision-distance evaluation. The tractor CG pose in the world frame is

$$T_G^W = \begin{bmatrix} \cos \varphi_t & -\sin \varphi_t & X \\ \sin \varphi_t & \cos \varphi_t & Y \\ 0 & 0 & 1 \end{bmatrix}.$$

If the tractor rectangle centroid is offset from the tractor CG by

$$r_{C_{tr}}^G = \begin{bmatrix} d_{x,tr} \\ d_{y,tr} \end{bmatrix},$$

then

$$T_{C_{tr}}^W = \begin{bmatrix} \cos \varphi_t & -\sin \varphi_t & X + d_{x,tr} \cos \varphi_t - d_{y,tr} \sin \varphi_t \\ \sin \varphi_t & \cos \varphi_t & Y + d_{x,tr} \sin \varphi_t + d_{y,tr} \cos \varphi_t \\ 0 & 0 & 1 \end{bmatrix} \quad (7.64)$$

Similarly, the articulation-joint pose in the world frame is

$$T_J^W = \begin{bmatrix} \cos(\varphi_t + \varphi_r) & -\sin(\varphi_t + \varphi_r) & X_J \\ \sin(\varphi_t + \varphi_r) & \cos(\varphi_t + \varphi_r) & Y_J \\ 0 & 0 & 1 \end{bmatrix}$$

If the trailer rectangle centroid is offset from the articulation-joint frame by

$$r_{C_{trl}}^J = \begin{bmatrix} d_{x,trl} \\ d_{y,trl} \end{bmatrix}$$

Then

$$T_{C_{trl}}^W = \begin{bmatrix} \cos \varphi_{trl} & -\sin \varphi_{trl} & X_J + d_{x,trl} \cos \varphi_{trl} - d_{y,trl} \sin \varphi_{trl} \\ \sin \varphi_{trl} & \cos \varphi_{trl} & Y_J + d_{x,trl} \sin \varphi_{trl} + d_{y,trl} \cos \varphi_{trl} \\ 0 & 0 & 1 \end{bmatrix} \quad (7.65)$$

where

$$\varphi_{trl} = \varphi_t + \varphi_r$$

These transformations map the articulated-vehicle states into the centroid-attached frames used for evaluating smooth convex distance functions for collision avoidance.

7.7 Obstacle-Aware Cold-Start

At the first MPC update, or whenever the shifted solution from the preceding control cycle is unavailable or unsuitable, the optimizer requires an initial horizon trajectory. A trivial initialization based on repeated current states or nominal path-following inputs may be ineffective when obstacles obstruct the direct recovery direction, since it can introduce large collision-constraint residuals into the initial iterate. An obstacle-aware cold-start procedure is therefore used to generate a dynamically propagated seed trajectory before the full NMPC iterations begin.

The proposed procedure combines nominal path recovery with local collision-distance information. Its construction is motivated by real-time potential-field obstacle avoidance [106], elastic-band path deformation [107], and admissible local velocity selection [108]. The obstacle-aware cold start is used only when a shifted solution from the preceding MPC cycle is unavailable or unsuitable; otherwise, the solver proceeds from the shifted previous solution, consistent with real-time NMPC strategies that exploit solution continuity between successive control updates [109].

7.7.1 Cold-Start Seed Construction

Let the prediction horizon contain $n_p + 1$ state stages. The cold-start seed is defined as

$$\mathcal{S}_{cs} = \{x_k^{cs}, \eta_k^{cs}, u_k^{cs}, \alpha_k^{cs}\}_{k=t}^{t+n_p}$$

where x_k^{cs} , η_k^{cs} , u_k^{cs} and α_k^{cs} denote the vehicle-state, path-state, control-input, and path-timing input components of the cold-start seed, respectively. The rollout begins from the estimated vehicle and path states,

$$x_0^{cs} = x_0, \quad \eta_0^{cs} = \eta_0$$

At each seed stage, the local seed direction is constructed from two pieces of information: the collision-clearance direction and the nominal path-recovery direction. Let d_k denote the minimum collision distance relative to the active obstacle set and let $d_{\min}$ denote the required minimum clearance. The local clearance margin is

$$\gamma_k = d_k - d_{\min}.$$

The analytical distance gradient developed previously in chapter 4 is used to define a normalized clearance-increasing direction,

$$\hat{n}_k = \frac{\nabla_x d_k}{\|\nabla_x d_k\| + \varepsilon_n} \quad (7.66)$$

Only vehicle-state components that influence the collision geometry contribute to $\nabla_x d_k$; the remaining components are assigned zero contribution in this local direction.

The local reference-path position and heading error, denoted by e_k^{ref} , are mapped into the geometric vehicle-state coordinates to define the nominal path-recovery direction. Here, Π_x selects the vehicle-state components corresponding to the reference-path error e_k^{ref} and maps all other components to zero

$$r_k = \Pi_x e_k^{\text{ref}}, \quad \hat{r}_k = \frac{r_k}{\|r_k\| + \varepsilon_r} \quad (7.67)$$

To provide an alternative direction when direct recovery conflicts with obstacle clearance, a tangential component is obtained by removing the component of r_k aligned with the clearance gradient:

$$t_k = (I - \hat{n}_k \hat{n}_k^\top) r_k, \quad \hat{t}_k = \frac{t_k}{\|t_k\| + \varepsilon_t} \quad (7.68)$$

The desired local seed direction is then constructed by blending reference recovery, tangential motion, and clearance increase:

$$v_k^{cs} = w_r(\gamma_k) \hat{r}_k + w_t(\gamma_k) \hat{t}_k + w_n(\gamma_k) \hat{n}_k \quad (7.69)$$

The vector v_k^{cs} defines a desired local motion for the seed trajectory. Since this motion must be realized through the tractor-trailer dynamics and admissible inputs, a target state is first formed from the current seed state,

$$x_{k+1}^{tar} = x_k^{cs} + \tau v_k^{cs} \quad (7.70)$$

where τ is the seed propagation interval. The target x_{k+1}^{tar} is not directly assigned as the next seed state. Instead, it is used only as a reference for selecting an admissible input. Let,

$$f_{x,seed}(x, u)$$

denote the initialization propagation map, which returns the next vehicle state obtained by applying input u from state x . At stage k , the input is chosen from the admissible set $U(x_k^{cs})$ so that the propagated state is closest to the target in (7.70):

$$u_k^{cs} = \underset{u \in U(x_k^{cs})}{\operatorname{argmin}} \|W_{cs} [f_{x,seed}(x_k^{cs}, u) - x_{k+1}^{tar}]\| \quad (7.71)$$

Here, W_{cs} weights the vehicle-state components relevant to the rollout. In the present implementation, the minimization is performed by directly evaluating a

small, fixed set of admissible wheel-torque and steering-angle combinations, rather than by solving a nested iterative optimization problem.

After u_k^{cs} is selected, the same propagation map is evaluated once more to define the candidate seed state,

$$x_{k+1}^{cand} = f_{x,seed}(x_k^{cs}, u_k^{cs}) \quad (7.72)$$

The candidate is accepted if it satisfies the required clearance,

$$d(x_{k+1}^{cand}) \geq d_{min} \quad (7.73)$$

If this condition is not satisfied, the candidate-generation step is repeated after modifying the local seed direction. The modification reduces the reference-directed contribution, increases the tangential or clearance-increasing contribution, selects the alternative tangential direction, or reduces the propagation step before advancing to the next horizon stage.

7.7.2 Cold Start Algorithm

Algorithm 7.1: Obstacle-Aware Cold-Start Initialization for the Tractor-trailer Unified MPPFC Problem

Inputs:

- Measured vehicle state x_t , current path state η_t and admissible control set $\mathcal{U}$, and admissible path-timing input set P
- Reference-path representation and active obstacle set,
- Seed vehicle-state propagation function $f_{x,seed}$ and path propagation function $f_{p,seed}$,
- Horizon length n_p , minimum clearance d_{min} , seed propagation interval τ , and

nominal path-timing input α_{nom}

Output: Cold-start seed trajectory $\mathcal{S}_{\text{cs}}$

Algorithm:

1. Initialize the seed at the current control cycle: $x_t^{\text{cs}} \leftarrow x_t, \eta_t^{\text{cs}} \leftarrow \eta_t$
2. For $k = t, \dots, t + n_p$:

1. Evaluate the local position and heading error relative to the reference path e_k^{ref}

2. Compute the nominal recovery direction:

$$r_k = \Pi_x e_k^{\text{ref}}, \quad \hat{r}_k = \frac{r_k}{\|r_k\| + \varepsilon_r}.$$

3. Evaluate the minimum obstacle distance and its analytical gradient at the current seed state, d_k and $\nabla_x d_k$
4. Compute the clearance margin and the normalized clearance-increasing direction:

$$\gamma_k = d_k - d_{\min}, \quad \hat{n}_k = \frac{\nabla_x d_k}{\|\nabla_x d_k\| + \varepsilon_n}$$

5. Compute the tangential obstacle-avoidance direction:

$$t_k = (I - \hat{n}_k \hat{n}_k^T) r_k, \quad \hat{t}_k = \frac{t_k}{\|t_k\| + \varepsilon_t}$$

6. Construct the clearance-guided seed direction:

$$v_k^{\text{cs}} = w_r(\gamma_k) \hat{r}_k + w_t(\gamma_k) \hat{t}_k + w_n(\gamma_k) \hat{n}_k$$

7. Construct the target state for the next seed stage:

$$x_{k+1}^{\text{tar}} = x_k^{\text{cs}} + \tau v_k^{\text{cs}}$$

8. Select an admissible seed control input:

$$u_k^{cs} = \underset{u \in \mathcal{U}(x_k^{cs})}{\operatorname{argmin}} \|W_{cs}[f_{x,seed}(x_k^{cs}, u) - x_{k+1}^{tar}]\|$$

9. Propagate the candidate vehicle-state seed:

$$x_{k+1}^{cand} = f_{x,seed}(x_k^{cs}, u_k^{cs})$$

10. Evaluate the candidate clearance:

$$d(x_{k+1}^{cand}) \geq d_{\min}$$

If the clearance condition is satisfied, accept the candidate:

$$x_{k+1}^{cs} \leftarrow x_{k+1}^{cand}$$

Otherwise, modify the local seed direction or reduce the propagation step and repeat Steps 2.7–2.10.

11. Project the accepted vehicle-state seed onto the reference path to determine the corresponding path-position component:

$$\eta_{1,k+1}^{cs} = \arg \min_{\eta_1} \|y(x_{k+1}^{cs}) - p_{\text{ref}}(\eta_1)\|^2$$

12. Using the integrator-chain path model from Chapter 3, estimate the remaining path-state components by finite differences

$$\eta_{i,k+1}^{cs} = \frac{\eta_{i-1,k+1}^{cs} - \eta_{i-1,k}^{cs}}{\tau}, \quad i = 2, \dots, n_\eta$$

13. The path-timing input is then selected as the highest-order derivative required by the path integrator chain

$$\alpha_k^{cs} = \frac{\eta_{n_\eta,k+1}^{cs} - \eta_{n_\eta,k}^{cs}}{\tau}$$

3. Return the cold-start seed trajectory:

$$\mathcal{S}_{\text{CS}} = \{x_k^{\text{CS}}, \eta_k^{\text{CS}}, u_k^{\text{CS}}, \alpha_k^{\text{CS}}\}_{k=t}^{t+n_p}$$

7.7.3 Computational Role and Limitations

The cold-start rollout is intended to provide a geometrically informed and dynamically propagated initial iterate when a suitable warm start is unavailable. If n_{obs} denotes the number of active obstacles and C_u denotes the computational cost of the admissible control-selection step, its approximate computational cost is

$$\mathcal{O}\left((n_p + 1)(n_{\text{obs}} + C_u)\right)$$

with storage proportional to the horizon length. In the implemented framework, online obstacle pruning limits the number of obstacles considered during initialization and therefore bounds this additional computational burden. The resulting seed is not a collision-free global plan and does not replace the constrained NMPC solution. Instead, it reduces poor initial collision geometry and provides a more useful initial iterate for the full optimizer. Final constraint satisfaction, optimality, and applied closed-loop control actions remain determined by the unified NMPC solve.

7.8 Simulation Framework and Setup

This section describes the closed-loop simulation framework and numerical configuration used to evaluate the proposed unified model predictive path-following and collision-avoidance framework for the articulated tractor-trailer system. The simulation environment is implemented in MATLAB/Simulink and integrates the vehicle model developed in Sections 7.1–7.6 with the path-generation, collision-distance, and structure-exploiting NMPC methods developed in the preceding

chapters.

All simulations and numerical experiments were conducted using MATLAB R2025b on a workstation equipped with an Intel Core i9-10885H CPU with 8 cores, a 2.4 GHz base clock, and 48 GB RAM, running Windows 11 Pro 64-bit. All algorithms, including derivative evaluation and NMPC simulations, were executed in interpreted MATLAB/Simulink mode. Therefore, the numerical results should be interpreted as algorithmic validation in a research environment rather than as a hardware-specific real-time benchmark. It is expected that code generation and compiled execution using MATLAB Coder or Simulink Coder would yield faster runtimes by reducing interpretation overhead and enabling more optimized numerical kernels.

7.8.1 Closed-Loop Simulation Architecture

Figure 7.2 shows the closed-loop simulation architecture. The Offline Scenario Database stores the reference routes, static obstacles, no-go zones, and prescribed dynamic-obstacle definitions. At each environment update, the Scenario Reader extracts the local path preview and active obstacle information near the current vehicle position. The active obstacle set may contain static map-based objects and dynamic obstacles supplied by an onboard perception system. To bound the online optimization size, only the most relevant nearby obstacles are retained at each control cycle.

The Composite Bézier Path Generator converts the local waypoint sequence into the smooth reference-path representation developed in Chapter 5, while the

Closest Point Projection Module determines the associated path-progress variables used by the predictive controller. The Unified MPPFC Controller solves the constrained nonlinear optimization problem formulated in Chapter 3 using the structure-exploiting interior-point method developed in Chapter 6. The controller simultaneously determines the vehicle-control inputs and path-progression variables subject to articulated vehicle dynamics, actuator and state limits, speed regulation, and collision-avoidance constraints. The Tractor-Trailer Vehicle Model propagates the nonlinear vehicle dynamics using the computed control inputs.

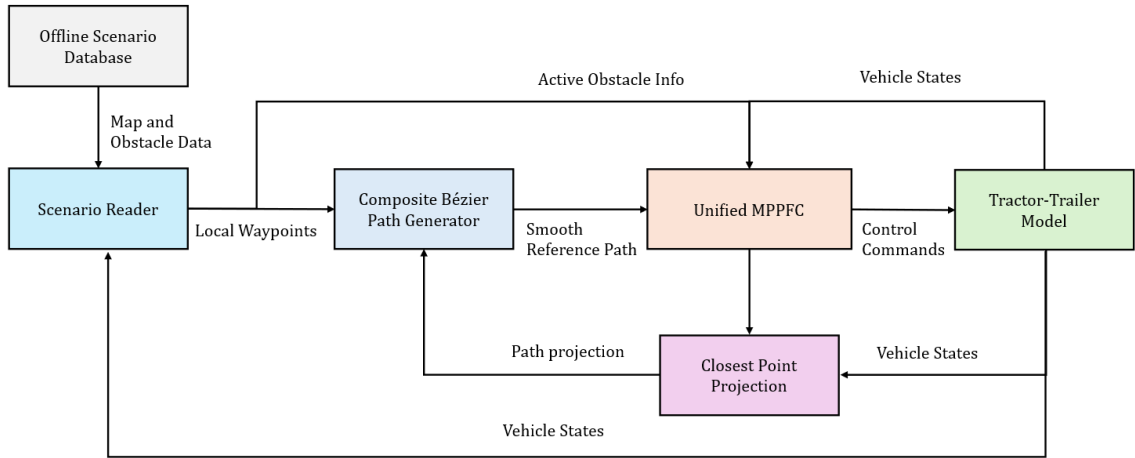


Figure 7.2: Closed-Loop Simulation Architecture for Tractor-Trailer Unified Path Following and Collision Avoidance

Collision avoidance is enforced using the smooth convex-distance formulation and its analytical derivatives developed previously in this dissertation. The ellipsoidal obstacle boundaries are selected conservatively to provide a geometric margin for uncertainties that arise in perception-based systems, including sensor noise, partial observability, limited resolution, state-estimation error, and uncertainty in obstacle pose and predicted motion. Consequently, an ellipsoid may

extend beyond the visible boundary of the represented object. This additional coverage is intentional and provides a computationally tractable safety margin while preserving the smoothness required for analytical collision-distance derivatives and second-order optimization.

7.8.2 Vehicle, Controller, and Simulation Parameters

The articulated tractor-trailer model parameters used in the closed-loop simulations are summarized in Table 7.2 and are representative of Class-8 heavy-duty vehicle configurations reported in the articulated vehicle-dynamics literature [103]– [105]. The simulation timing, controller configuration, preview, and obstacle-selection parameters are summarized in Table 7.3.

Parameter	Value	Parameter	Value	Parameter	Value
l_1	2.6 m	$d_{y,trl}$	0.0 m	C_a	3.5 N·s/m
l_2	3.2 m	m_1	9000 kg	μ_0	0.008
d_1	2.7 m	m_2	27000 kg	α	0.0 rad
l_3	4.8 m	I_{z_1}	52000 kg·m ²	β	0.0 rad
d_3	6.2 m	I_{z_2}	750000 kg·m ²	k_{art}	4500000 N·m/rad
$d_{x,tr}$	0.1 m	$C_{\alpha f}$	300000 N/rad	c_{art}	4500000 N·m·s/rad
$d_{y,tr}$	0.0 m	$C_{\alpha r}$	600000 N/rad	C_a	3.5 N·s/m
$d_{x,trl}$	5.8 m	$C_{\alpha t}$	960000 N/rad	μ_0	0.008

Table 7.2: Numerical Parameters of the Tractor-Trailer Model

Configuration Parameter	Value	Description
Vehicle model cycle time	0.01 s	Sampling time used for numerical integration of the nonlinear tractor-trailer vehicle dynamics model
Environment model cycle time	0.2 s	Update interval for environment and obstacle information
Controller model cycle time	0.05 s	Internal controller update rate used for supervisory and intermediate computations
MPC cycle time	0.2 s	Receding-horizon optimization update interval

MPC prediction model sample time	0.2 s	Discretization interval used within the MPC prediction model
Look-ahead distance	1000 m	Forward path-preview distance used for local path extraction and trajectory generation
Look-back distance	200 m	Backward path-preview distance retained behind the current vehicle position
Maximum number of Bézier segments	20	Maximum number of composite Bézier path segments used for local path representation
Maximum number of active obstacles	5	Maximum number of nearby obstacles retained after obstacle pruning
Maximum number of iterations per MPC update	15	Maximum number of interior-point iterations allowed for each receding-horizon optimization solve
MPC prediction horizon duration	10 s	Time duration covered by the finite-horizon MPC prediction window
Number of MPC prediction steps	50	Number of discretization intervals in the MPC horizon, computed from the 10 s horizon and 0.2 s prediction-model sample time

Table 7.3: Tractor-Trailer Simulation and Controller Parameters

Variable	Minimum	Maximum	Scaling Factor
Tractor position (X)	-	-	2500 m
Tractor position (Y)	-	-	2500 m
Tractor heading angle (φ_t)	-	-	2π rad
Articulation angle (φ_r)	-0.15 rad	0.15 rad	0.35 rad
Tractor longitudinal velocity (v_x)	5 m/s	30 m/s	40 m/s
Tractor lateral velocity (v_y)	-5 m/s	5 m/s	5 m/s
Tractor yaw rate (ψ_t)	-0.125 rad/s	0.125 rad/s	0.35 rad/s
Articulation rate (ψ_r)	-0.10 rad/s	0.10 rad/s	0.35 rad/s
Wheel torque (T_w)	-5000 Nm	5000 Nm	10000 Nm
Wheel angle (δ)	-0.145 rad	0.145 rad	0.45 rad
Path position (η_1)	-	-	300 m
Path Speed (η_2)	1 m/s	40 m/s	20 m/s

Table 7.4: Tractor-Trailer MPPFC Constraints and Scaling Factors

The state and control constraints and corresponding scaling factors are summarized in Table 7.4. Throughout the simulation studies, the reference speed is 20m/s, the minimum obstacle clearance is 20m, and the interior-point solver

termination tolerance is 10^{-4} . The selected tolerance is appropriate for online NMPC, where the optimization problem is re-solved at successive control updates using updated feedback and warm-start initialization.

7.9 Closed-Loop Simulation Results

Scenario	Scenario Description	Primary Objective
S1	Static-map recovery scenario with large initial path offset and favorable initial heading	Evaluate unified path recovery and static obstacle avoidance under large tracking error conditions
S2	Challenging static-obstacle recovery scenario with large initial path offset and unfavorable initial heading requiring significant obstacle-induced deviation	Evaluate recovery behavior under severe maneuvering and constraint-limited conditions
S3	Combined dynamic-obstacle scenario including slower-moving, stopped, and opposing obstacles	Evaluate simultaneous path following and dynamic obstacle avoidance
S4	Multi-obstacle following scenario with two slower vehicles traveling along the reference path at the same speed	Evaluate coordinated avoidance of multiple moving obstacles within one predictive optimization problem
C1	Scenario S2 using the Gauss–Newton collision-Hessian approximation	Evaluate the effect of retaining full analytical collision Hessians on solver convergence

Table 7.5: Tractor–Trailer Closed-Loop Simulation Scenarios

The closed-loop simulation studies evaluate the proposed unified tractor–trailer MPPFC framework under static and dynamic obstacle conditions. The objective is to assess path-recovery capability, collision-clearance satisfaction, recursive convergence under a limited interior-point iteration budget, and the effect of retaining full analytical collision-constraint Hessians. The evaluated scenarios are

summarized in Table 7.5.

7.9.1 Static-Map Path-Convergence Scenarios

The first simulation set evaluates convergence toward the reference path in the presence of static obstacles and no-go regions. The reference route is generated from the static map and converted online into the smooth composite Bézier representation developed in Chapter 5. The resulting closed-loop trajectories are therefore determined jointly by the path-following objective, obstacle-clearance constraints, articulated vehicle dynamics, and actuator and state limits.

Scenario 1: Large Path Offset with Favorable Initial Heading

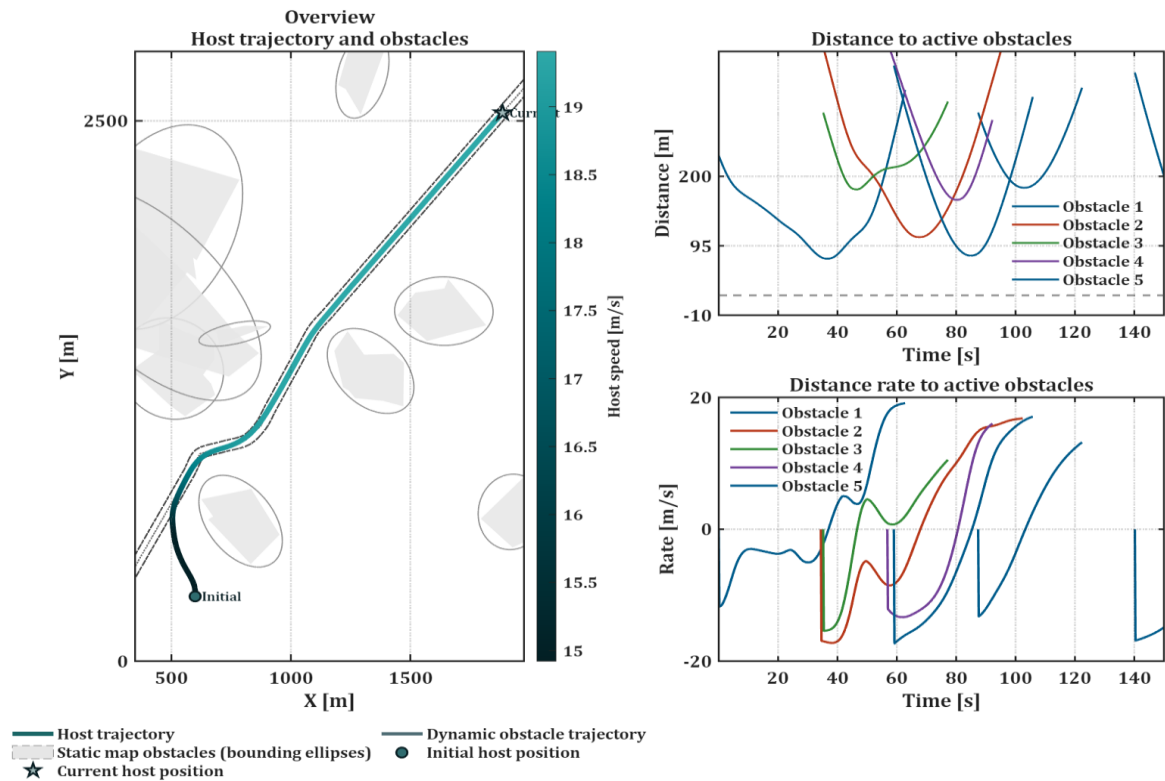


Figure 7.3: Static-map recovery scenario S1 with large initial path offset and favorable initial heading.

Scenario S1 considers a large initial displacement from the reference corridor

with an initial heading favorable to recovery. Figure 7.3 shows the static-map environment, reference path, host trajectory, and the convex obstacle approximations used by the collision-distance formulation. Although the host vehicle begins far from the path and several static obstacles are present near the recovery region, the controller generates a smooth trajectory that converges toward the reference corridor while maintaining the required collision clearance.

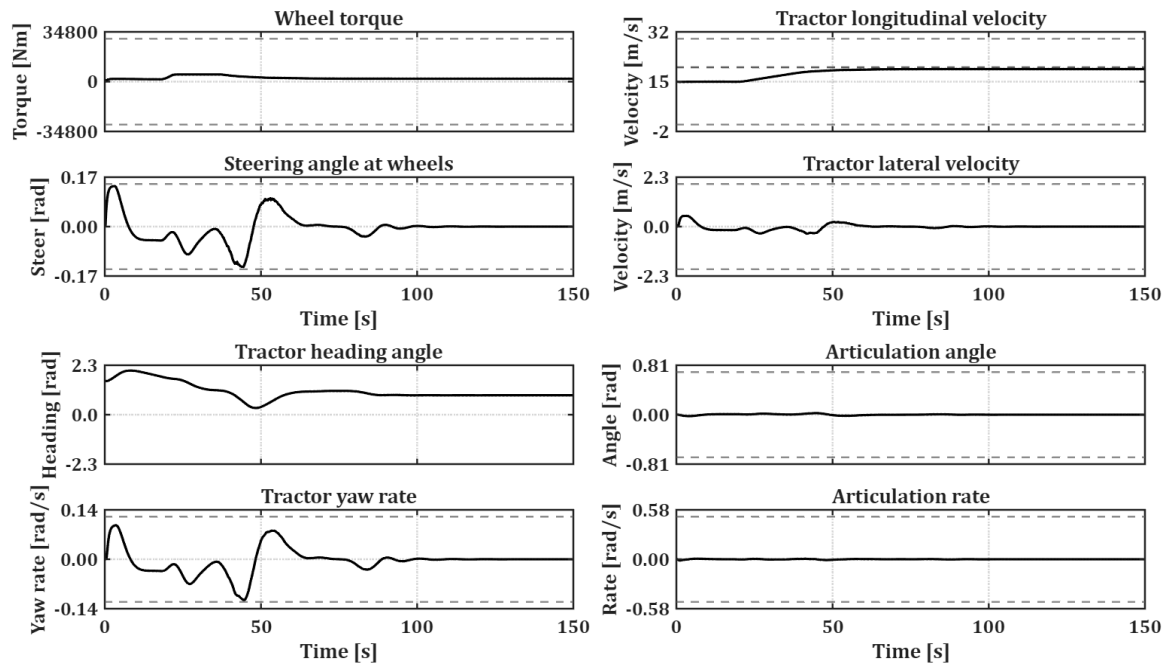


Figure 7.4: Vehicle states and control responses for static-map recovery scenario S1.

Figures 7.4 and 7.5 show the corresponding state, input, and path-following responses. The steering angle reaches its bound during the initial heading correction and again as the vehicle completes alignment with the reference path. The lateral and heading errors decrease as the host vehicle converges toward the corridor, while the path-position variable indicates continuous forward progression. A small steady-state longitudinal speed error remains, but it does not affect the collision-

aware recovery objective considered in this study.

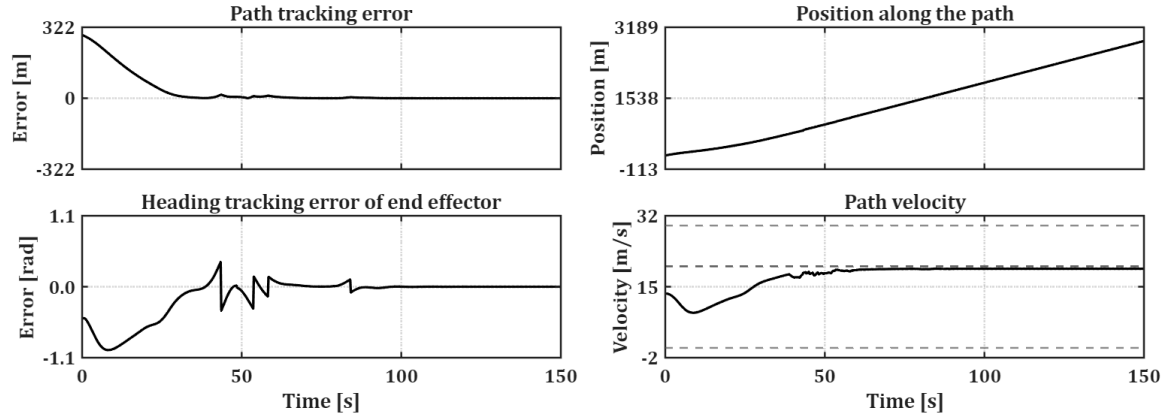


Figure 7.5: Path-following performance for static-map recovery scenario S1.

Overall, Scenario S1 demonstrates recovery from a large initial path offset while satisfying collision-clearance, vehicle-dynamics, and actuator constraints in the presence of nearby static obstacles.

Scenario 2: Challenging Obstacle-Constrained Recovery

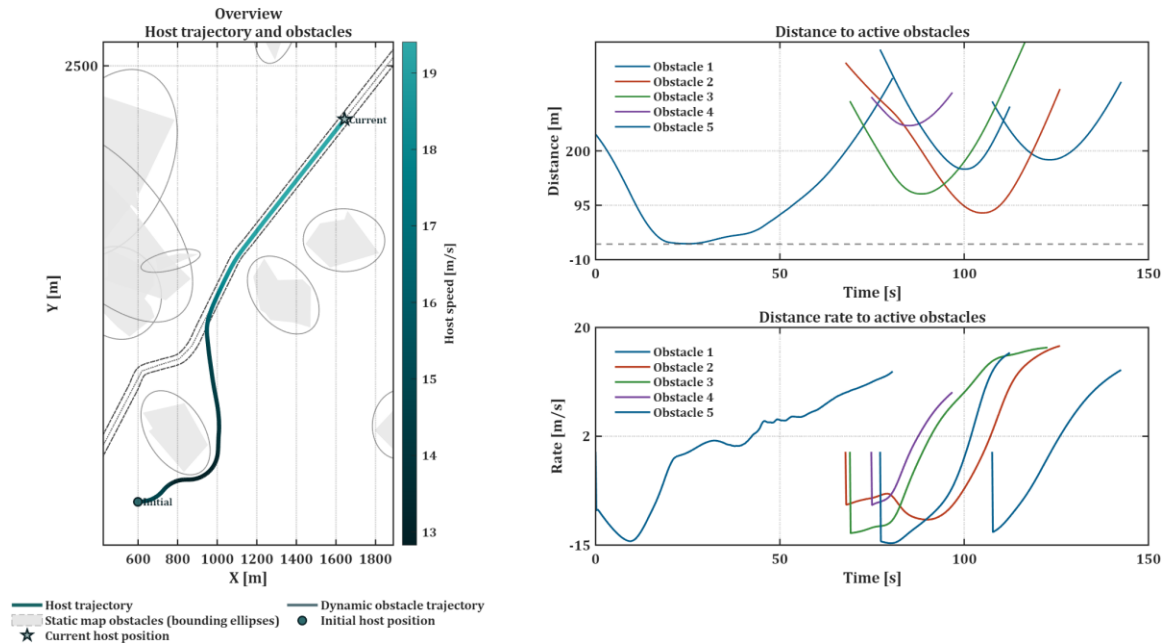


Figure 7.6: Static-map recovery scenario S2 with unfavorable initial heading and obstacle-constrained recovery

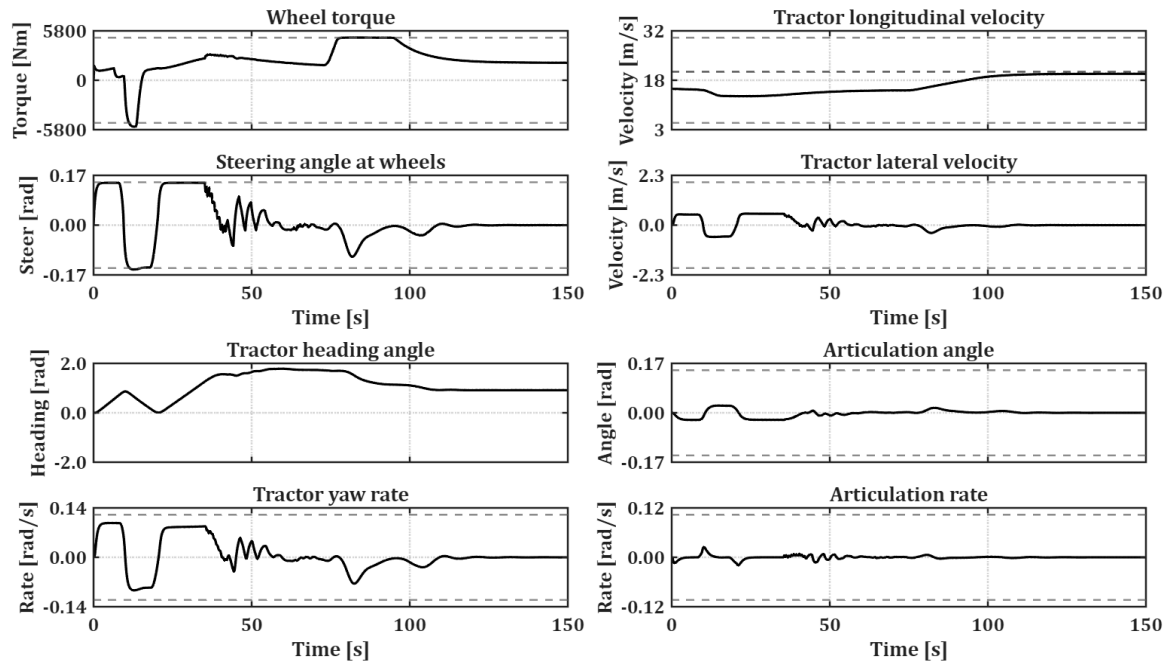


Figure 7.7: Vehicle states and control responses for static-map recovery scenario S2

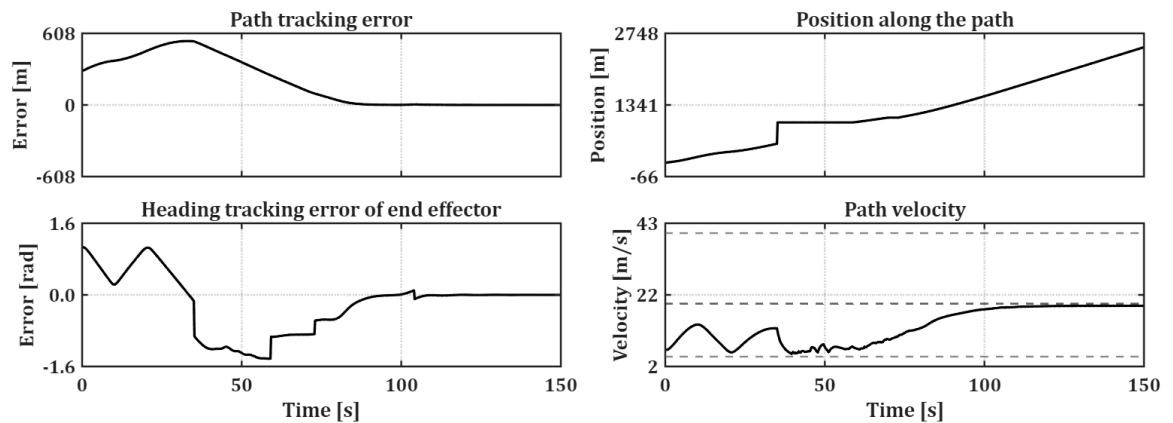


Figure 7.8: Path-following performance for static-map recovery scenario S2

Figure 7.6 shows the more challenging static-map recovery scenario S2. In this case, the vehicle begins far from the reference path with an unfavorable initial heading, while a large obstacle lies near the recovery region. Consequently, direct convergence toward the path is not feasible, and the controller initially drives the vehicle farther from the reference corridor before completing the reorientation and

obstacle-avoidance maneuver required for recovery.

Figures 7.7 and 7.8 show that the maneuver is strongly constraint-limited. Compared with Scenario S1, the steering angle remains at its bound for a longer duration, and the wheel torque reaches both acceleration and deceleration limits as the controller coordinates vehicle reorientation, speed regulation, and obstacle avoidance. The path-tracking error initially increases because direct recovery toward the reference path is obstructed by the nearby obstacle; after completing the avoidance maneuver, the vehicle progressively converges toward the reference corridor while maintaining collision clearance.

Figure 7.9 shows the number of interior-point iterations required at each MPC update. Despite the severity of the maneuver, the prescribed 10^{-4} termination tolerance is satisfied during approximately 99.5% of the closed-loop control cycles, corresponding to a non-convergence rate of approximately 0.5%.

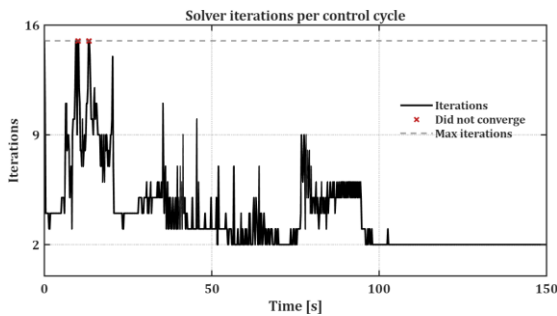


Figure 7.9: Interior-point solver iterations for static-map scenario S2

During the isolated cycles that do not satisfy the tolerance within the iteration limit, the closed-loop maneuver continues without loss of collision clearance or instability.

7.9.2 Dynamic Obstacle Avoidance Scenarios

Scenario S3 evaluates the proposed controller in the presence of dynamic obstacles introduced during closed-loop operation. In contrast to the static map

obstacles, these objects represent online obstacle information that may be provided by onboard perception systems. The scenario contains three representative interactions along the same route: D01 is a slower obstacle traveling in the same direction as the host vehicle, D02 is a stopped obstacle located along the reference corridor, and D03 is an opposing obstacle traveling toward the host vehicle. No separate local planner, passing rule, or obstacle-specific maneuver generator is used.

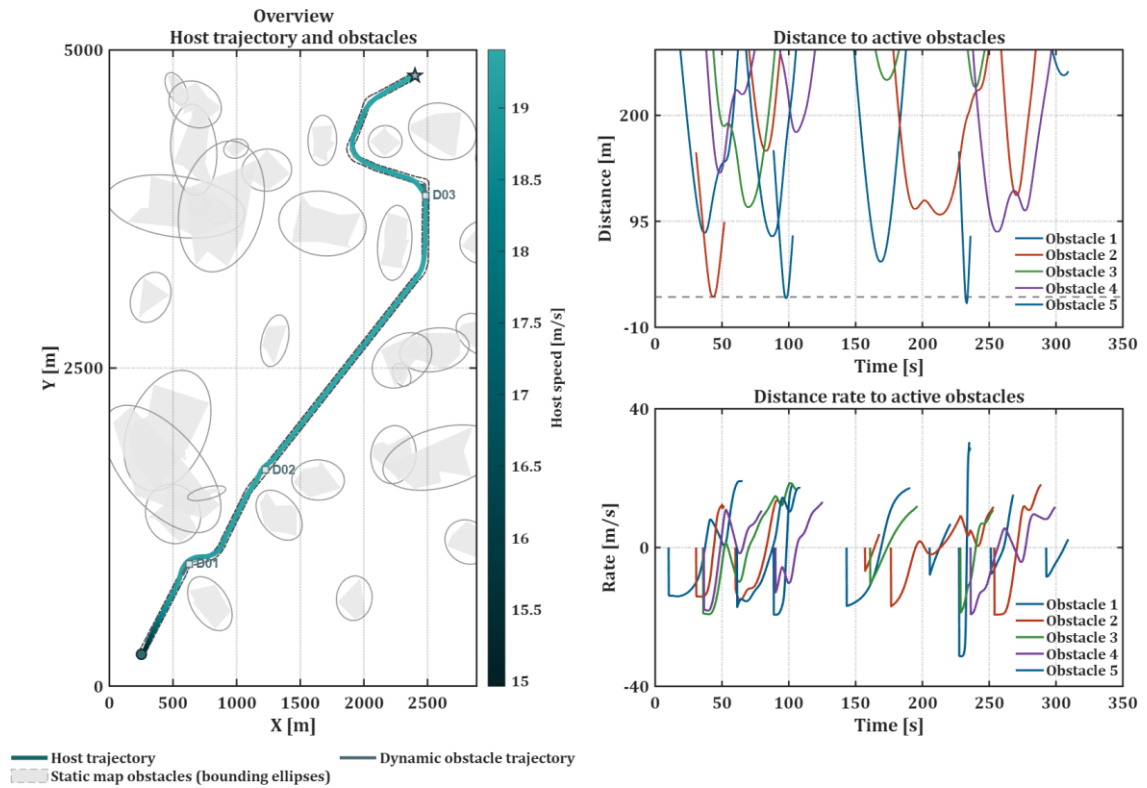


Figure 7.10: Combined dynamic-obstacle scenario including slower-moving, stopped, and opposing obstacles.

Figure 7.10 shows the complete dynamic-obstacle simulation together with the obstacle-distance and distance-rate histories. The online pruning procedure retains at most five nearby obstacles in the optimization problem at each control

cycle. The required 20m minimum clearance is maintained during all three interactions.

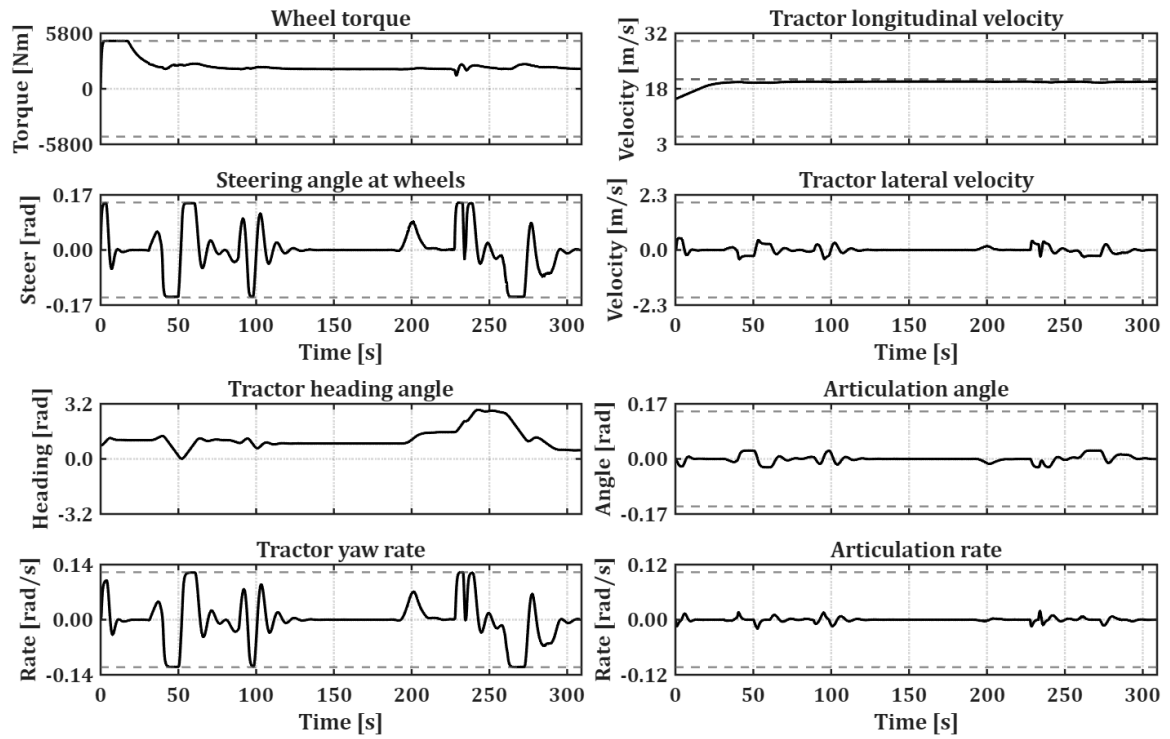


Figure 7.11: Vehicle states and controls during the combined dynamic-obstacle scenario

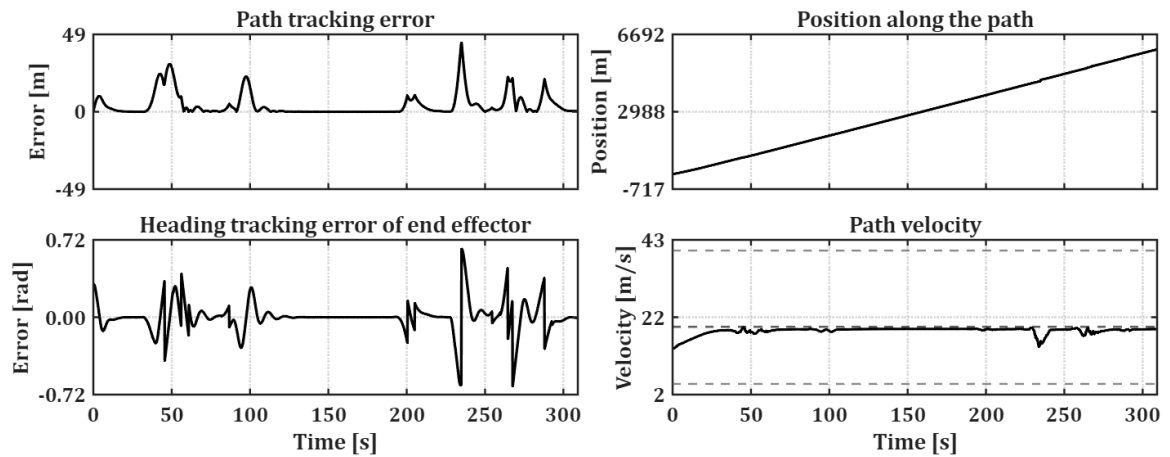


Figure 7.12: Path-following and path-progression variables during the combined dynamic-obstacle scenario

Figures 7.11 and 7.12 show the corresponding vehicle responses and path-progression variables. The steering angle and yaw rate become strongly active during the avoidance maneuvers, while the wheel torque reaches its upper limit during the initial acceleration toward the reference speed. The lateral path error increases only when deviation from the reference path is required to maintain collision clearance and subsequently decreases as the vehicle recovers after each interaction. The path-position variable increases continuously, confirming uninterrupted forward progression along the route.

Figure 7.13 shows the interior-point iteration count over the simulation, which extends beyond 300s. Only one cycle fails to satisfy the prescribed convergence tolerance within the allocated iteration budget. The increased iteration counts occur primarily during obstacle interactions, where collision and actuator constraints become active; after these events, many cycles again converge within two to five interior-point iterations.

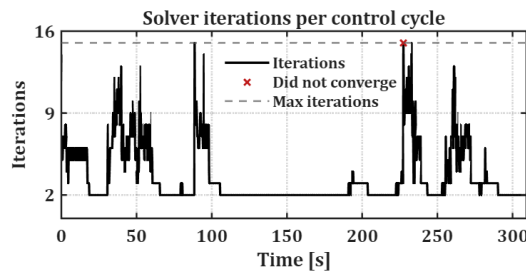


Figure 7.13: Interior-point iteration count during the combined dynamic-obstacle scenario

This behavior indicates that the shifted warm start remains effective during successive MPC updates, including through transient collision-avoidance events.

Interaction with D01: Slower Same-Direction Obstacle

Figure 7.14 shows the interaction with D01, which travels in the same direction as the host vehicle at approximately 5m/s as both vehicles approach a

right-hand curve. The host vehicle passes D01 on the left without an explicit passing rule. This maneuver emerges from the optimization problem and may be attributed to the combined effects of local path curvature, map geometry, vehicle orientation, and steering and yaw-rate constraints, which make the left-side trajectory more favorable in this interaction.

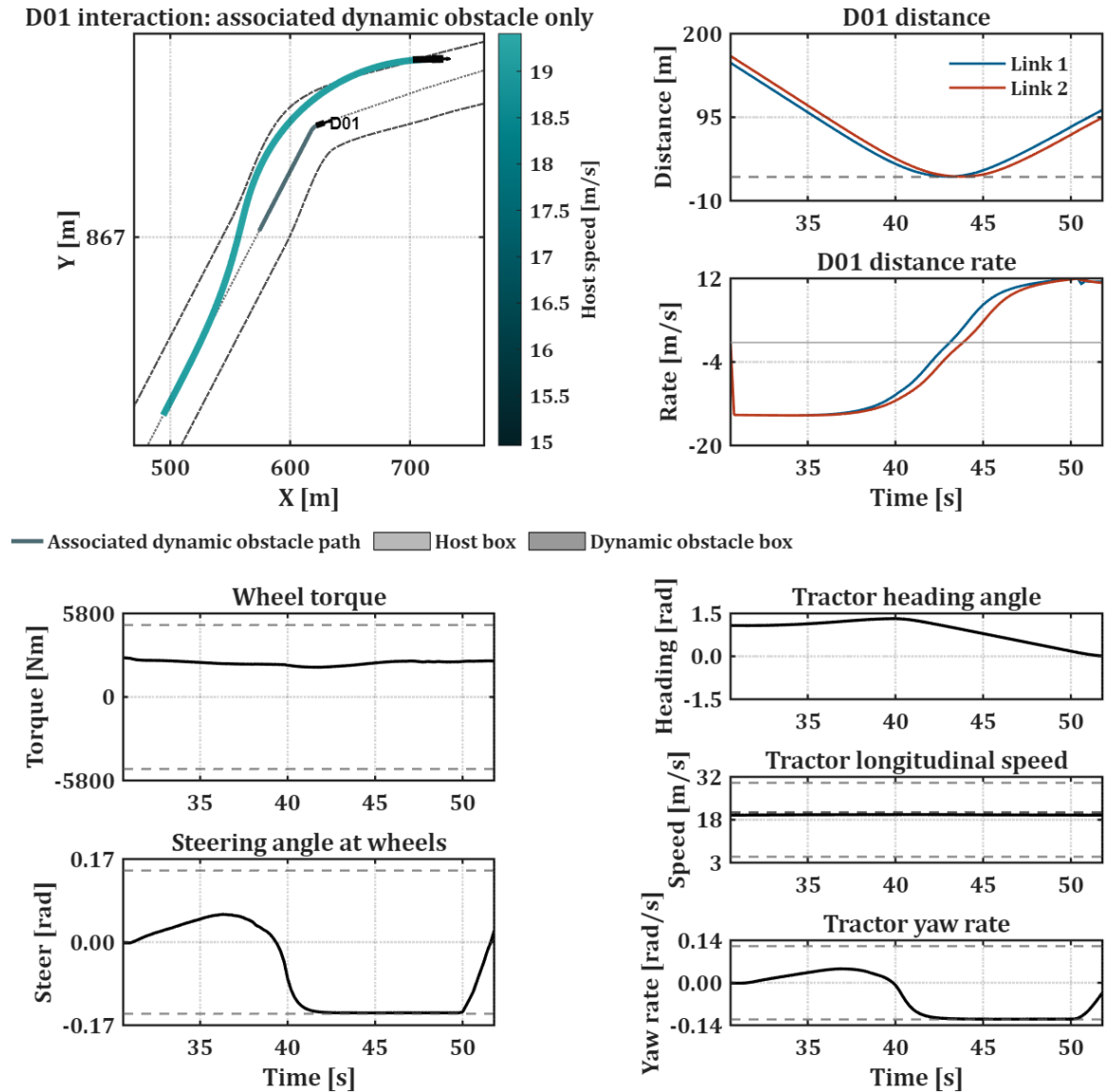


Figure 7.14: Local interaction with dynamic obstacle D01 traveling in the same direction as the host vehicle

The required 20 m minimum clearance is maintained throughout the maneuver, while the steering angle and yaw rate approach their limits, indicating that the pass is dynamically demanding but feasible.

Interaction with D02: Stopped Obstacle Along the Reference Corridor

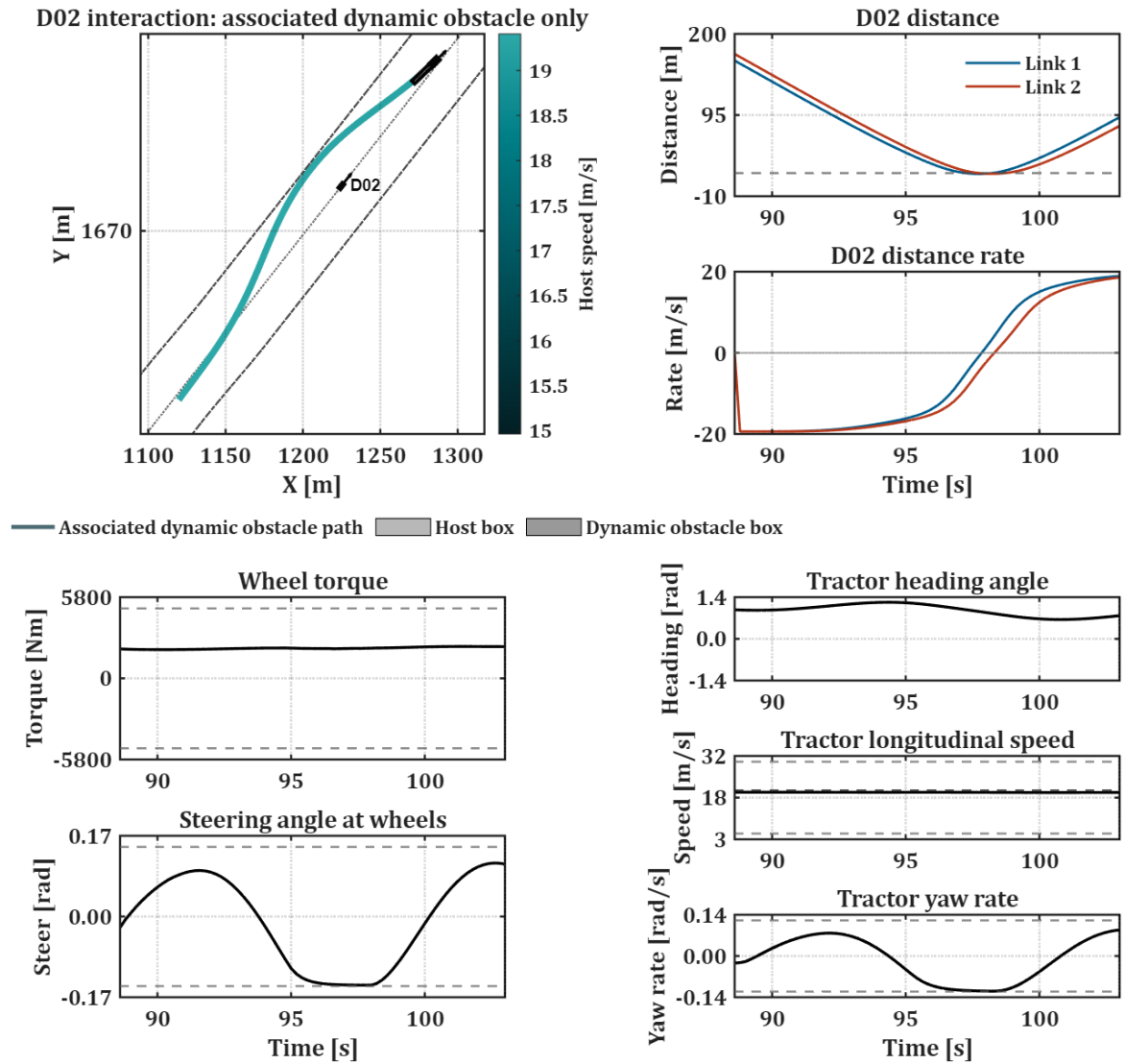


Figure 7.15: Local interaction with dynamic obstacle D02 stopped along the reference corridor

Figure 7.15 shows the interaction with D02, which is stopped along the reference corridor. The controller initiates a lateral deviation before reaching the

obstacle, maintains the required clearance during the pass, and returns the host vehicle toward the path after the obstacle has been cleared.

Interaction with D03: Opposing Dynamic Obstacle

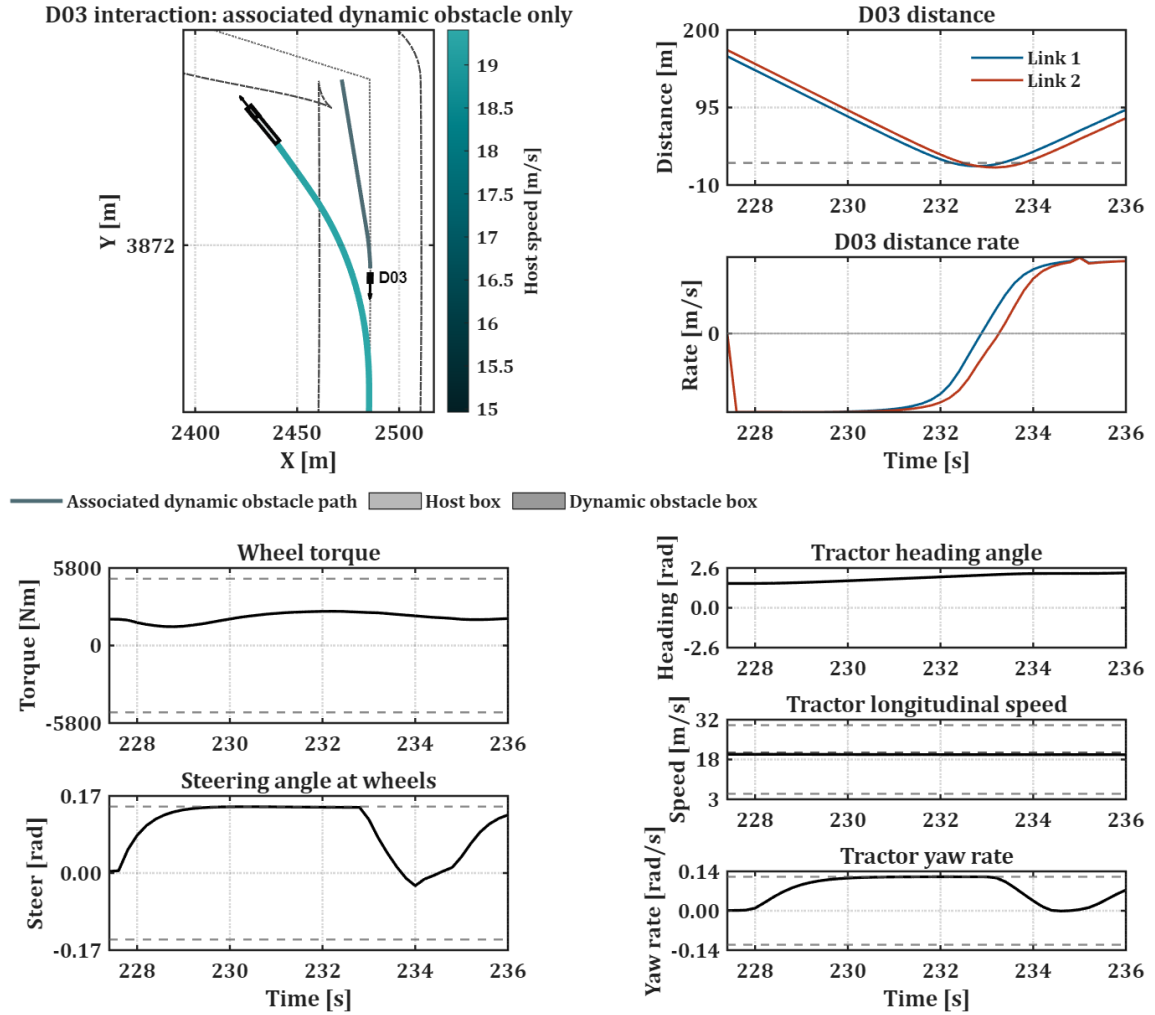


Figure 7.16: Local interaction with dynamic obstacle D03 traveling opposite to the host vehicle

Figure 7.16 shows the interaction with D03, which approaches the host vehicle in the opposite direction at approximately 12m/s near a curved path segment. Because of the higher relative closing speed, this is the most time-critical interaction in Scenario S3. Nevertheless, the controller generates a feasible lateral

avoidance maneuver, maintains the required clearance, and subsequently returns toward the reference corridor.

Overall, Scenario S3 demonstrates that the unified NMPC framework can handle slower-moving, stopped, and opposing dynamic obstacles within a single optimization problem while maintaining collision clearance, forward progression, and strong recursive convergence behavior.

7.9.3 Multiple Dynamic Obstacles

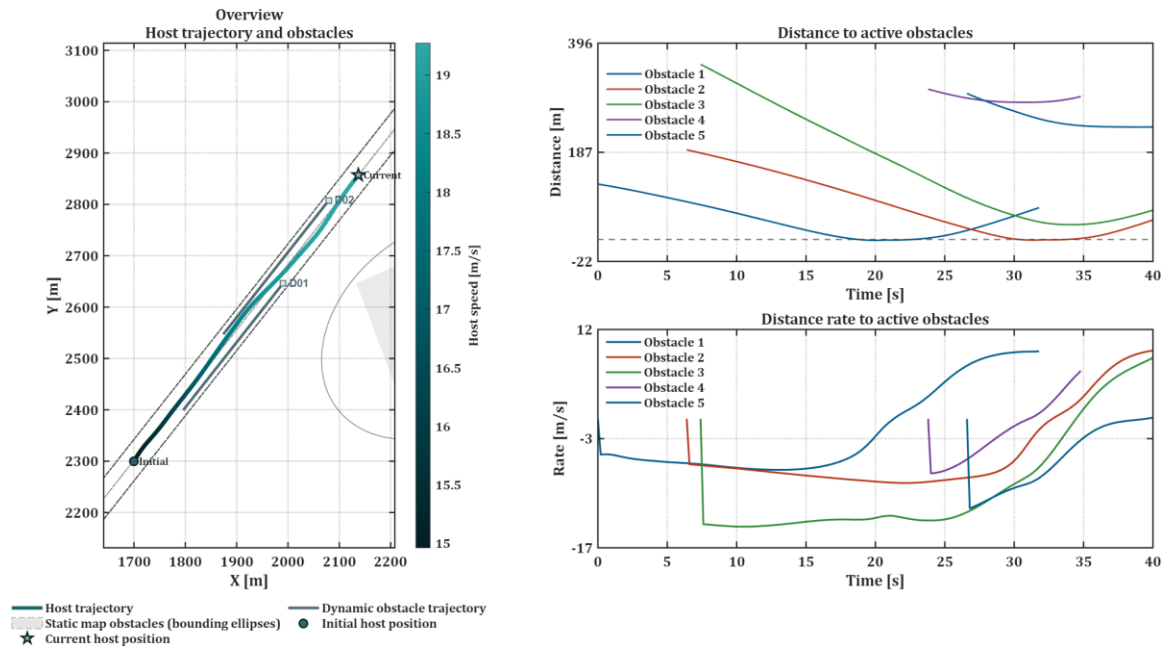


Figure 7.17: Multi-obstacle same-direction scenario with two slower vehicles traveling along the reference corridor

Scenario S4 considers two slower dynamic obstacles traveling along the reference corridor at approximately 10m/s, with an initial separation of approximately 100m. As shown in Figure 7.17, the host vehicle moves laterally away from the reference path to avoid the first obstacle, continues along the laterally displaced trajectory to maintain clearance from the second obstacle, and then

returns toward the path. Although the maneuver resembles a double lane change response, it is not intended to represent a conventional lane level maneuver because the prescribed 20 m minimum clearance is larger than a typical lane width.

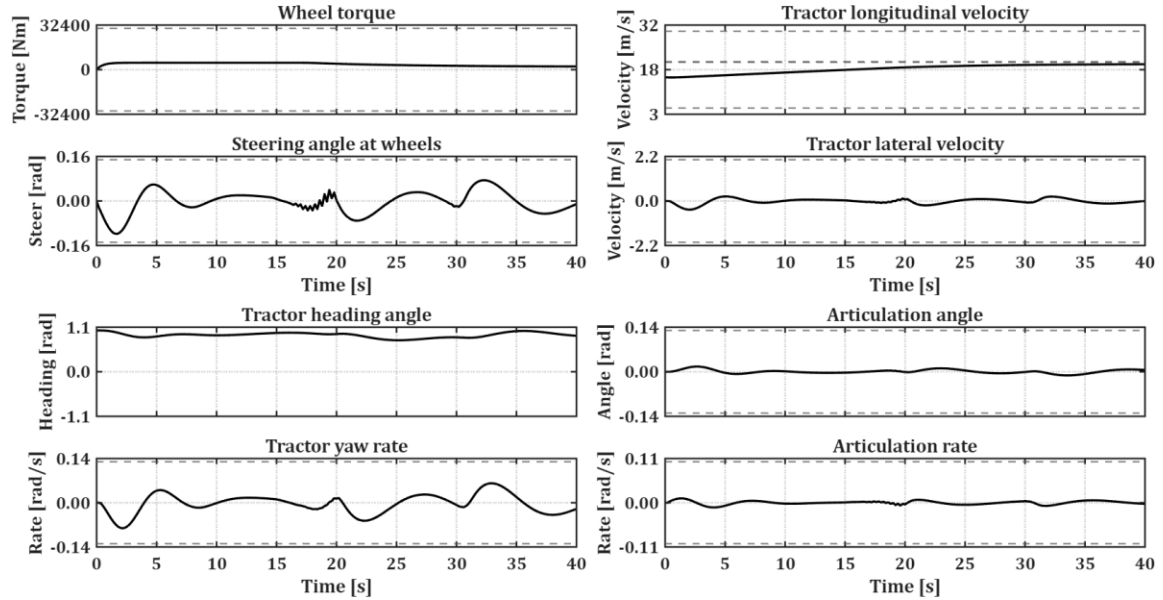


Figure 7.18: Vehicle states and control responses for the multi-obstacle same-direction scenario

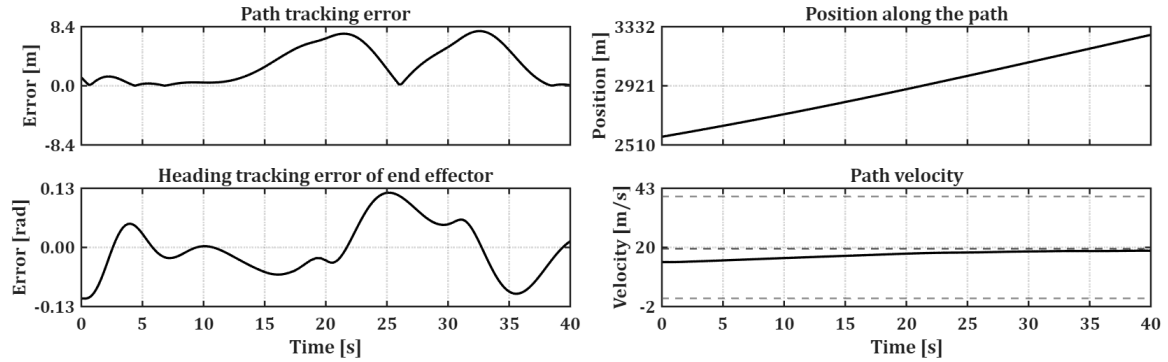


Figure 7.19: Path-following and path-progression variables for the multi-obstacle same-direction scenario

Figures 7.18 and 7.19 show that the required clearance is maintained while the state and input trajectories remain within their imposed limits. Two temporary increases in lateral tracking error correspond to the sequential avoidance phases,

after which the vehicle converges back toward the reference corridor. A small switching behavior appears in the steering response during the transition between the two avoidance phases. This occurs because the optimizer is balancing clearance from two nearby moving obstacles while also reducing path error and maintaining forward progression. Since the steering input remains within its imposed limit, this switching does not compromise feasibility.

Figure 7.20 shows the local interaction with the first slower same-direction obstacle. The host vehicle establishes the lateral displacement required to clear the first obstacle while accounting for the second obstacle farther along the reference corridor. This confirms that the maneuver is optimized with respect to the simultaneously active obstacle set rather than generated as an isolated reaction to a single obstacle.

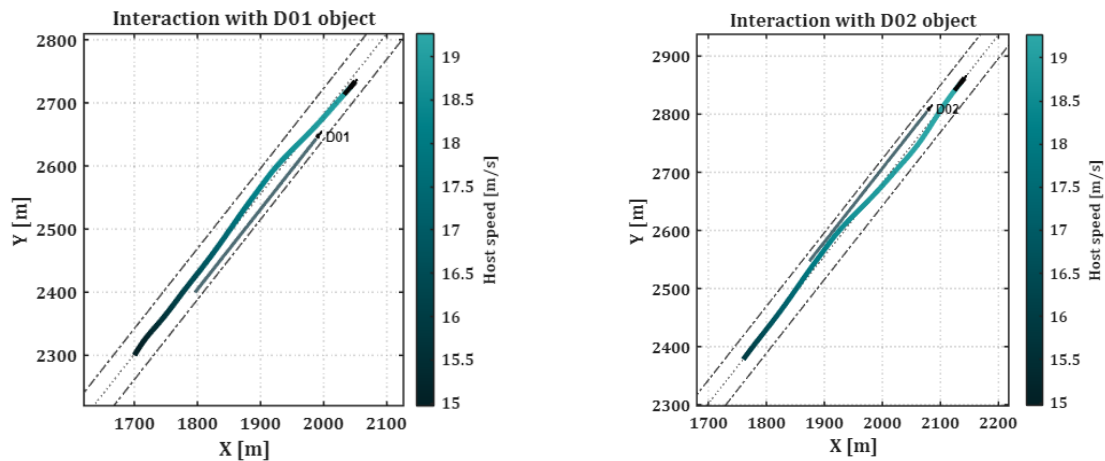


Figure 7.20: Local interaction with same-direction obstacles in Scenario S4

Overall, Scenario S4 demonstrates that the proposed unified NMPC framework can coordinate avoidance of multiple same-direction moving obstacles within one predictive optimization problem while preserving collision clearance,

feasible vehicle responses, and uninterrupted forward path progression.

7.9.4 Effect of Full Collision-Constraint Hessians

This comparison evaluates the effect of retaining the full analytical Hessian terms of the collision-avoidance constraints using the challenging static-recovery scenario S2.

The only change between the two cases is the collision-curvature representation: the Gauss–Newton approximation neglects second-order collision-curvature terms, whereas the full-Hessian formulation retains the analytical collision Hessian developed

previously in this dissertation.

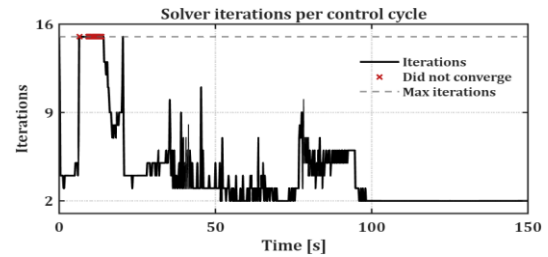


Figure 7.21: Interior-point iteration history for Scenario S2 using the Gauss–Newton collision-Hessian approximation.

Figure 7.21 shows the interior-point iteration history for the Gauss–Newton case. With this approximation, 21 of 751 control cycles do not satisfy the prescribed 10^{-4} convergence criterion, corresponding to a non-convergence rate of 2.8%. In comparison, the full-Hessian formulation shown in Figure 7.9 produces only 4 non-converged cycles over the same 751 control cycles, corresponding to a non-convergence rate of 0.5%. The Gauss–Newton case exhibits a prolonged interval of elevated iteration count near the large static obstacle, where the collision constraints become strongly active. Of the 21 non-converged cycles, 18 remain acceptable for closed-loop control under the implemented real-time feasibility and control-quality checks, while only 3 fail these broader acceptance conditions. In

addition, only 7 of the non-converged cycles have feasibility error greater than 10^{-4} . These results indicate that the Gauss–Newton approximation mainly slows the reduction of optimality error near active collision constraints rather than causing persistent feasibility loss. Overall, this comparison shows that retaining the full collision Hessian primarily improves convergence robustness near active collision constraints, while preserving the trajectory-level behavior of closed-loop controller.

7.10 Chapter Summary

This chapter evaluated the proposed unified nonlinear model predictive path-following and collision-avoidance framework for an articulated tractor–trailer vehicle. The closed-loop simulations show that the controller can recover from large initial path-tracking errors, avoid static and dynamic obstacles, satisfy vehicle-dynamics and actuator constraints, and maintain forward path progression within a single predictive optimization problem.

The static-map studies demonstrate successful recovery under both moderate and highly constrained conditions. In Scenario S1, the vehicle recovered from a large initial path offset while maintaining collision clearance. In the more challenging Scenario S2, the vehicle began with an unfavorable orientation and encountered a large obstacle near the recovery region, requiring a substantial reorientation and avoidance maneuver. Despite active steering and torque limits, the solver satisfied the prescribed 10^{-4} tolerance in approximately 99.5% of the closed-loop control cycles. The dynamic-obstacle studies show that the same formulation can handle slower same-direction traffic, stopped obstacles, opposing obstacles, and

multiple same-direction moving obstacles without a separate local planner or rule-based maneuver logic. Temporary path deviations emerged naturally when required for clearance, after which the vehicle returned toward the reference path.

The Hessian comparison confirms the benefit of retaining full analytical collision-constraint Hessians. In Scenario S2, the Gauss–Newton approximation produced 21 non-converged cycles out of 751, while the full-Hessian formulation reduced this number to 4. This indicates that the Gauss–Newton approximation mainly slows optimality-error reduction near active collision constraints, whereas the full-Hessian formulation improves convergence robustness.

To further test robustness, the prediction horizon was increased from 10s to 20s for Scenario S2, producing a significantly larger and more difficult nonlinear optimization problem. Even in this case, the solver maintained approximately 97% convergence to the prescribed tolerance, with the remaining loss in convergence expected to be improvable through better warm-start initialization for the longer-horizon problem.

Overall, the results support the main objective of this dissertation: a unified NMPC framework with differentiable constraints, analytical derivatives, obstacle-aware initialization, warm starting, and sparse KKT system solves can provide reliable closed-loop path following and obstacle avoidance for articulated vehicle systems. The strong convergence behavior under fixed iteration limits further supports the suitability of the proposed formulation for fast online computation and future real-time implementation through code generation and compiled execution.

CHAPTER 8 – APPLICATION 2: PATH FOLLOWING FOR AN UNDERACTUATED LOG-ARM ROBOT

8.1 System Model and Control Setup

The log-arm system considered in this chapter, taken from [76], is a 3-link, 3-joint underactuated mechanical system consisting of a 2-joint robot arm mounted on a cylindrical log, as shown in Fig. 8.1. The generalized coordinates are the log rotation angle θ_1 , the arm revolute angle θ_2 , and the prismatic displacement d_3 .

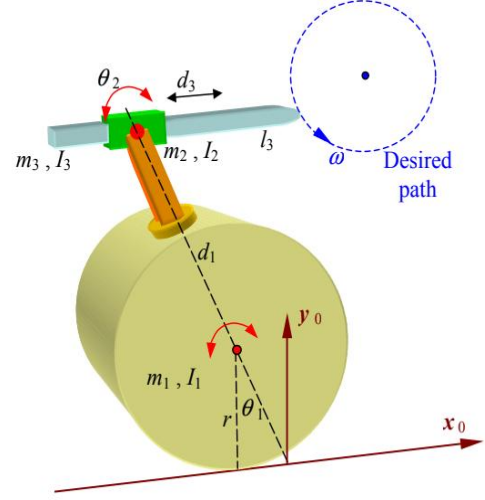


Figure 8.1: Underactuated Log-Arm Robot and Desired Circular Path

The corresponding generalized position, velocity vector and control input are defined as

$$\rho = [\theta_1 \quad \theta_2 \quad d_3]^\top, \quad \xi = [\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{d}_3]^\top, \quad u = [\tau_2 \quad F_3]^\top$$

where, τ_2 is the torque applied at the revolute arm joint and F_3 is the force applied along the prismatic joint. Since the log coordinate θ_1 is not directly actuated, the system is underactuated.

The term “three-link, three-joint system” refers to the mechanical and dynamic model used in [76]. This is distinct from the convex-body representation introduced in Chapter 3 for collision and distance calculations. In the geometric framework of Chapter 3, a physical link may be represented by one or more convex

bodies, and multiple fixed components may be grouped when they move together. Therefore, the mechanical link count used in this chapter does not restrict the number of convex bodies used for geometric modeling. The link and joint terminology used here follows the mechanical-system definition in [76], while the convex-body link representation used for distance and collision modeling in Chapter 3 is treated separately.

The physical parameters used in the simulation are listed in Table 8.1. The state and input constraints used in the NMPC problem are listed in Table 8.2.

Parameter	Value
Mass of link 2, m_1	4 kg
Mass of link 2, m_2	1 kg
Mass of link 3, m_3	1 kg
Radius of log, r	1 m
Log center to link 3 joint distance, d_1	1.5 m
Link 3 length, l_3	0.8 m
Inertia of log, I_1	2 kg/m ²
Inertia of link 2, I_2	0.3 kg/m ²
Inertia of link 3, I_3	0.3 kg/m ²

Table 8.1: Physical Parameters of the Log-Arm System

Variable	Minimum	Maximum
θ_1	$-\pi/2$ rad	$\pi/2$ rad
θ_2	$-\pi/2$ rad	$\pi/2$ rad
d_3	-0.8 m	0.8 m
$\dot{\theta}_1$	-2π rad /s	2π rad /s
$\dot{\theta}_2$	-2π rad /s	2π rad /s
$\dot{d}_3$	-4 m/s	4 m/s
τ_2	-10 Nm	10 Nm
F_2	-10 N	10 N

Table 8.2: Log-Arm MPPFC State and Control Constraints

The log-arm dynamics are expressed using the robot model introduced in Chapter 3, Eq. (3.26),

$$\begin{aligned}\dot{\rho} &= \xi \\ \dot{\xi} &= M^{-1}(\rho)(-H(\rho, \xi)\xi - K(q) + G(\rho, \xi)u)\end{aligned}$$

This case study evaluates obstacle-free output path following for the underactuated log-arm system. The objective is to drive the end effector along a circular path while satisfying the nonlinear dynamics and the imposed state and input constraints. The

main challenges are:

- Underactuation: The log coordinate θ_1 is not directly actuated, so the controller must influence it indirectly through τ_2 and F_3 .
- Nonlinear coupling: The coordinates θ_1 , θ_2 , and d_3 are strongly coupled through the mass matrix and passive dynamic terms.
- Output-space path following: The circular path is defined in end-effector space, requiring coordinated motion of all three generalized coordinates.
- State and input constraints: The bounds on the prismatic motion and actuator inputs restrict the reachable output motion and can lead to constrained path-following behavior.

The purpose of this chapter is therefore to demonstrate that the proposed MPPFC formulation can produce constrained output-path-following behavior for a severely underactuated nonlinear robot. The results are evaluated using the closed-loop end-effector trajectory, path-error and state and input histories.

8.2 Dynamics and Output Model

The total kinetic energy of the log-arm system is given by [76]:

$$\begin{aligned}
 T = & \frac{1}{2}m_1r^2\dot{\theta}_1^2 + \frac{1}{2}m_2 \left[(r\dot{\theta}_1 + d_1\dot{\theta}_1c_1)^2 + d_1^2\dot{\theta}_1^2s_1^2 \right] \\
 & + \frac{1}{2}m_3 \left[(d_1\dot{\theta}_1c_2 + r\dot{\theta}_1c_{12} - \dot{d}_3)^2 + (d_1\dot{\theta}_1s_2 + r\dot{\theta}_1s_{12} + d_3(\dot{\theta}_1 + \dot{\theta}_2))^2 \right] \quad (8.1) \\
 & + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}(I_2 + I_3)(\dot{\theta}_1 + \dot{\theta}_2)^2
 \end{aligned}$$

and the total potential energy is

$$V = m_2gd_1c_1 + m_3g(d_1c_1 + d_3s_{12}) \quad (8.2)$$

where

$$\begin{aligned}
c_1 &= \cos \theta_1, & s_1 &= \sin \theta_1, \\
c_2 &= \cos \theta_2, & s_2 &= \sin \theta_2, \\
c_{12} &= \cos(\theta_1 + \theta_2), & s_{12} &= \sin(\theta_1 + \theta_2)
\end{aligned}$$

The output y , defined as the end-effector position with respect to the fixed base, is:

$$y = h_e(\rho) = \begin{bmatrix} (d_3 + l_3)c_{12} - d_1s_1 - r\theta_1 \\ (d_3 + l_3)s_{12} + d_1c_1 + r \end{bmatrix} \quad (8.3)$$

The system matrices are obtained from the Euler-Lagrange equation [76]

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\rho}} \right) - \frac{\partial L}{\partial \rho} = Q_\rho, \quad L(\rho, \xi) = T(\rho, \xi) - V(\rho),$$

where Q_ρ is the vector of generalized nonconservative forces. Since the system is actuated only through τ_2 and F_3 , the generalized force vector is

$$Q_\rho = [0 \quad \tau_2 \quad F_3]^\top = Gu, \quad G = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (8.4)$$

The mass matrix is obtained from the kinetic energy as

$$M_{ij}(\rho) = \frac{\partial^2 T}{\partial \xi_i \partial \xi_j}$$

Using the definitions

$$a_1 = \frac{1}{2} [I_1 + I_2 + I_3 + m_1 r^2 + (m_2 + m_3)(d_1^2 + r^2)]$$

$$a_2 = (m_2 + m_3)d_1 r$$

$$a_3 = \frac{1}{2} (I_2 + I_3)$$

the mass matrix becomes

$$M(\rho) = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{12} & m_{22} & 0 \\ m_{13} & 0 & m_3 \end{bmatrix} \quad (8.5)$$

where

$$m_{11} = 2a_1 + 2a_2c_1 + m_3d_3^2 + 2m_3d_1d_3s_2 + 2m_3rd_3s_{12}$$

$$m_{12} = m_3d_3^2 + m_3d_1d_3s_2 + m_3rd_3s_{12} + 2a_3$$

$$m_{13} = -m_3(d_1c_2 + rc_{12})$$

$$m_{22} = 2a_3 + m_3d_3^2$$

The gravitational generalized force vector is obtained as

$$K(\rho) = \nabla_\rho V$$

Therefore,

$$K(\rho) = \begin{bmatrix} -m_2gd_1s_1 - m_3gd_1s_1 + m_3gd_3c_{12} \\ m_3gd_3c_{12} \\ m_3gs_{12} \end{bmatrix} \quad (8.6)$$

The remaining velocity-dependent passive terms are obtained from

$$N(\rho, \xi) = \frac{d}{dt} \left(\frac{\partial T}{\partial \xi} \right) - \frac{\partial T}{\partial \rho}.$$

The full passive vector satisfies

$$N(\rho, \xi) + K(\rho) = \frac{d}{dt} \left(\frac{\partial T}{\partial \xi} \right) - \frac{\partial T}{\partial \rho} + \frac{\partial V}{\partial \rho}$$

The velocity-dependent vector $N(\rho, \xi)$ can be written in the form

$$N(\rho, \xi) = H(\rho, \xi)\xi.$$

This factorization is not unique. A convenient representation consistent with the implemented dynamics is

$$H(\rho, \xi) = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & 0 & h_{23} \\ h_{31} & h_{32} & 0 \end{bmatrix} \quad (8.7)$$

where

$$h_{11} = (-a_2 s_1 + m_3 r d_3 c_{12}) \dot{\theta}_1 + m_3 d_3 (d_1 c_2 + r c_{12}) \dot{\theta}_2,$$

$$h_{12} = m_3 d_3 (d_1 c_2 + r c_{12}) (\dot{\theta}_1 + \dot{\theta}_2),$$

$$h_{13} = 2m_3 (d_3 + d_1 s_2 + r s_{12}) (\dot{\theta}_1 + \dot{\theta}_2),$$

$$h_{21} = -m_3 d_1 d_3 c_2 \dot{\theta}_1,$$

$$h_{23} = 2m_3 d_3 (\dot{\theta}_1 + \dot{\theta}_2),$$

$$h_{31} = -m_3 (d_3 + d_1 s_2) \dot{\theta}_1 - m_3 d_3 \dot{\theta}_2,$$

$$h_{32} = -m_3 d_3 (\dot{\theta}_1 + \dot{\theta}_2),$$

The output path-following objective uses the end-effector error

$$e_p = h_e(\rho) - r_p(\eta_1)$$

where $r_p(\eta_1)$ is the desired geometric path and η_1 is the optimized path coordinate.

For the circular path used in this chapter, $y_{1,c}$ and $y_{2,c}$ denote the center coordinates of the reference circle in the y_1 - y_2 plane, and,

$$r_p(\eta_1) = \begin{bmatrix} y_{1,c} - r_c \cos \eta_1 \\ y_{2,c} - r_c \sin \eta_1 \end{bmatrix} \quad (8.8)$$

and the path tangent is

$$\frac{\partial r_p}{\partial \eta_1} = \begin{bmatrix} r_c \sin \eta_1 \\ -r_c \cos \eta_1 \end{bmatrix} \quad (8.9)$$

Thus, positive path velocity

$$\eta_2 = \dot{\eta}_1 > 0$$

corresponds to counterclockwise motion along the circle in the y_1 - y_2 plane, while

$$\eta_2 < 0$$

corresponds to clockwise motion.

The prediction problem penalizes the path tracking error e_p , input usage,

selected state deviations, and path-speed tracking. Here, t is the current optimization-cycle index, n_p is the horizon length, and k is the stage index, consistent with Chapter 3. Consistent with the objective terms introduced in Eqns. (3.51) and (3.54), the position and path-speed penalties are written as

$$J_p(\rho, \eta_1) = \sum_{k=t}^{t+n_p} e_{pos}^\top(\rho_k, \eta_{1,k}) Q_p e_{pos}(\rho_k, \eta_{1,k})$$

$$J_v(\eta_2) = \frac{1}{2} \sum_{k=t}^{t+n_p} Q_v (\eta_{2,k} - r_v)^2$$

The path-speed term encourages progression along the path without imposing a hard lower bound on η_2 . This is important for the log-arm system because the system is underactuated and may need to temporarily reduce or reverse path motion while recovering to the desired path. Unlike a hard progress constraint, this soft penalty allows the optimizer to trade path progression against dynamic feasibility, state limits, and output-tracking performance.

8.3 Simulation Results

The simulation and solver parameters used for the log-arm path-following study are summarized in Table 8.3. Three circular-path cases are considered to evaluate the closed-loop behavior of the proposed NMPC formulation.

Parameter (Units)	Value
Log arm system sample time	0.01 s
Controller cycle time	0.1 s
MPC prediction model sample time	0.1 s
Number of MPC prediction steps	50
Maximum number of iterations per MPC update	15

Table 8.3: Log-Arm MPPFC Simulation and Solver Parameters

The main difficulty in this application is that the desired output path must be produced through the coupled motion of θ_1 , θ_2 , and d_3 , as shown in Eq. (8.3), while satisfying the imposed state and input constraints.

In all three simulations, the solver achieved the prescribed convergence tolerance in more than 95% of the MPC updates. Some of the remaining updates reached the maximum iteration budget, but the accepted iterates still produced bounded closed-loop behavior and usable control inputs.

8.3.1 Small-Radius Circular Path

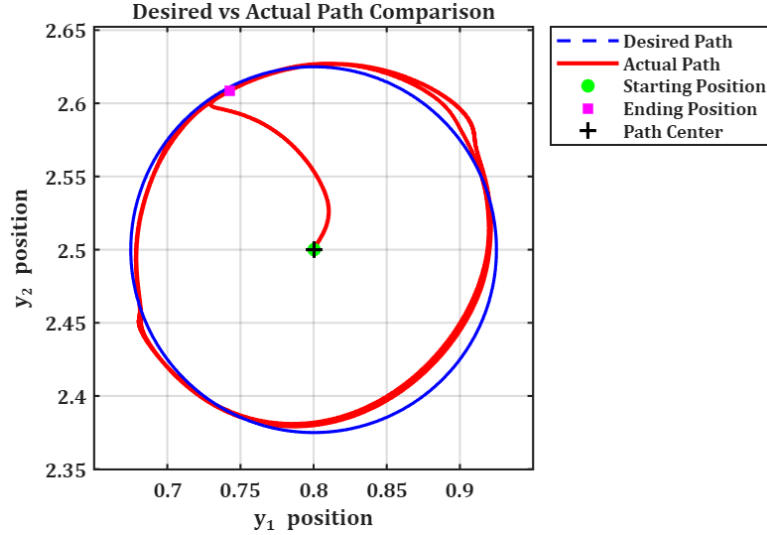


Figure 8.2: Path tracking performance of the log-arm system for the 0.125 m radius case

For $r_c = 0.125$ m, the end effector maintains sustained motion along the desired circular path, as shown in Fig. 8.2. The actual trajectory remains close to the reference path over most of the simulation, with only small, localized deviations. These deviations are consistent with the nonlinear coordination required among the generalized coordinates rather than with loss of closed-loop control.

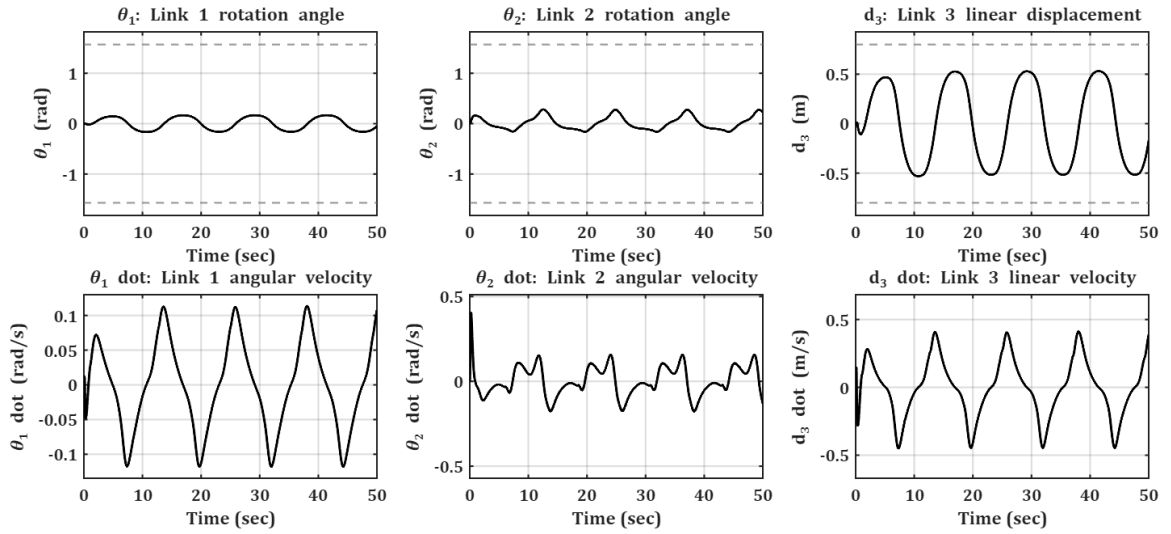


Figure 8.3: Evolution of system log-arm states for 0.125 m radius case

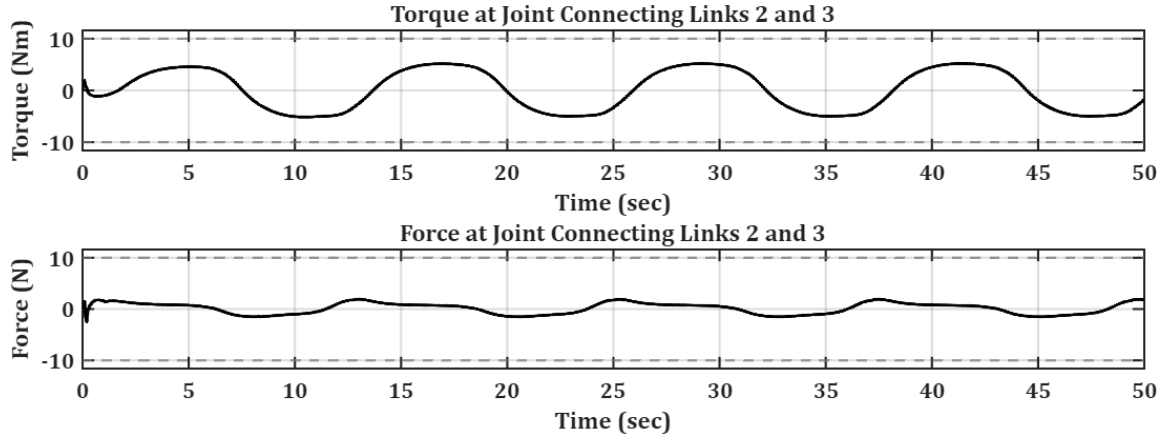


Figure 8.4: Log-arm system control inputs for the 0.125 m radius case

The state histories in Fig. 8.3 show that the generalized coordinates and velocities remain bounded. The prismatic displacement reaches approximately 0.5 m, which is below the 0.8 m upper bound. Thus, sustained traversal is achieved without persistent activation of the prismatic displacement constraint. The control inputs shown in Fig. 8.4 remain within their imposed limits. The torque and force histories indicate that the path-following behavior is obtained without excessive

actuation or persistent input saturation.

Figure 8.5 shows that the optimized path coordinate progresses continuously, while the path velocity remains bounded. The variation in path velocity reflects the optimizer balancing path progression against output tracking and dynamic feasibility. The iteration history in Fig. 8.6 shows that most MPC updates converge within the prescribed iteration budget.

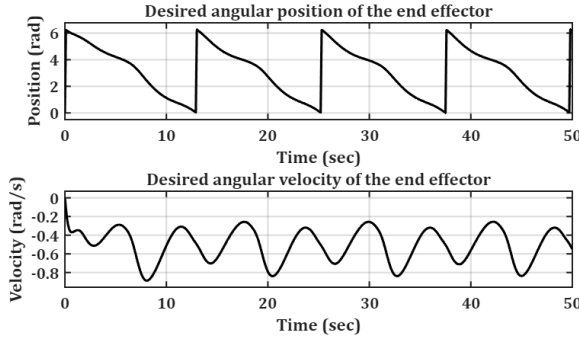


Figure 8.5: Log-arm system path position and path velocity for 0.125 m radius case

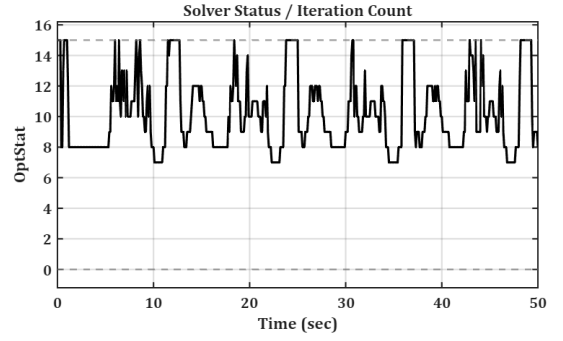


Figure 8.6: Number of interior-point iterations for the 0.125 m radius case

8.3.2: Medium-Radius Circular Path

For $r_c = 0.25$ m, the controller maintains overall traversal of the circular path, but the trajectory exhibits larger localized deviations than in the previous case, as shown in Fig. 8.7. The larger path requires more demanding coordination of the angular and prismatic motion, and this is reflected in both the output trajectory and the state histories.

The state histories in Fig. 8.8 show that the localized deviations occur when the prismatic displacement reaches its imposed limits for extended portions of the maneuver. Once the prismatic constraint becomes active, the controller has reduced

radial positioning freedom and must compensate through the angular coordinates θ_1 and θ_2 . The resulting motion remains bounded, but the active state constraint increases the difficulty of maintaining accurate output tracking throughout the full circle.

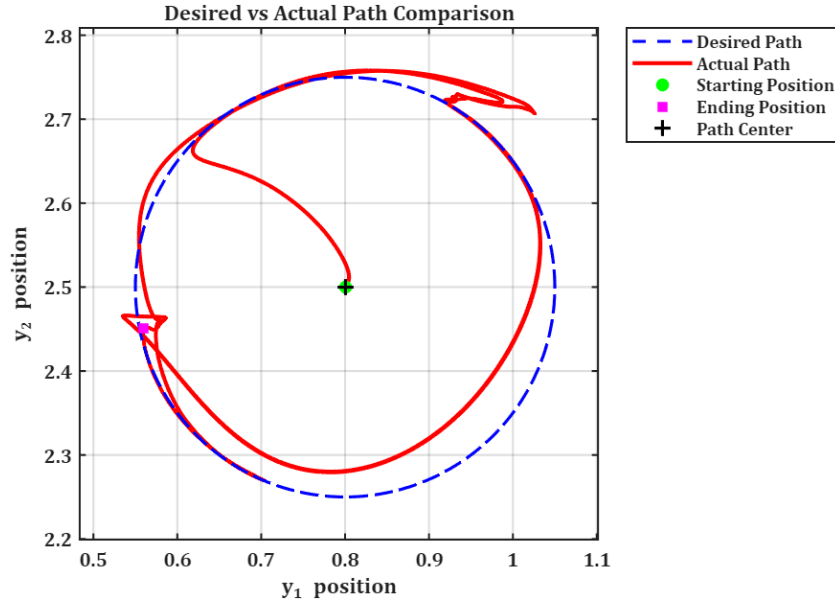


Figure 8.7: Path tracking performance of the log-arm system for the 0.25 m radius case

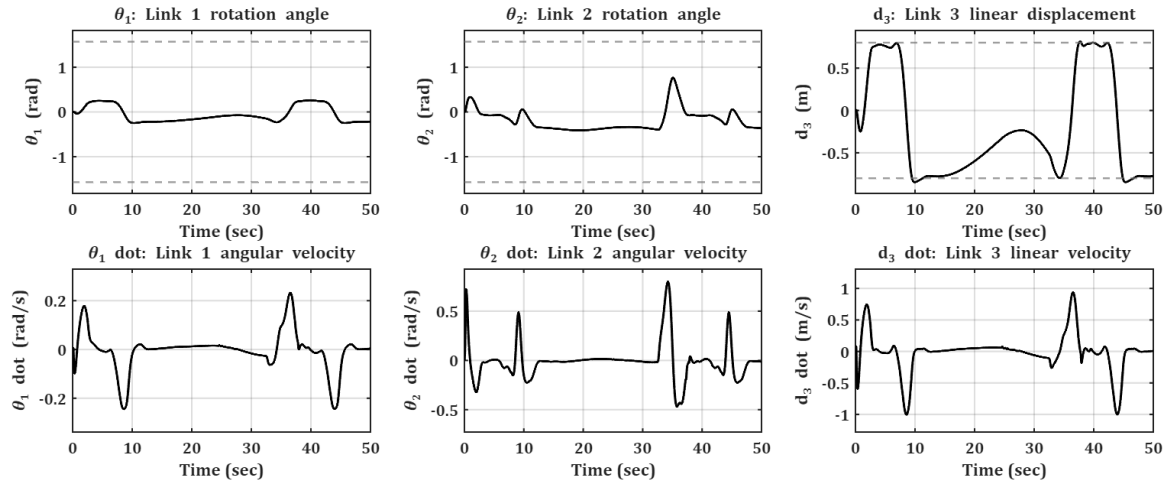


Figure 8.8: Evolution of system log-arm states for 0.25 m radius case

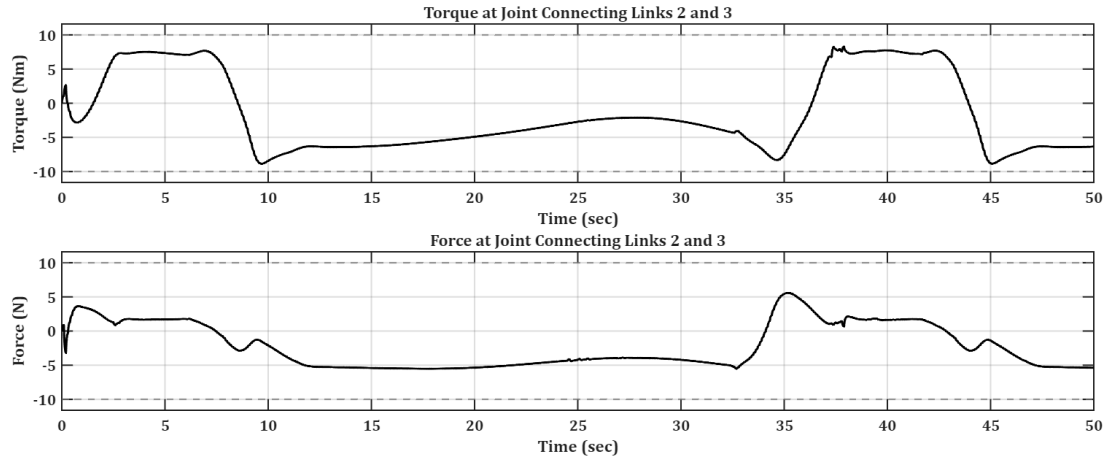


Figure 8.9: Log-arm system control inputs for the 0.25 m radius case

The control inputs in Fig. 8.9 remain bounded despite the activation of the prismatic displacement constraint. The torque and force profiles show the additional effort required to maintain path progression and reduce output error under the more demanding path geometry.

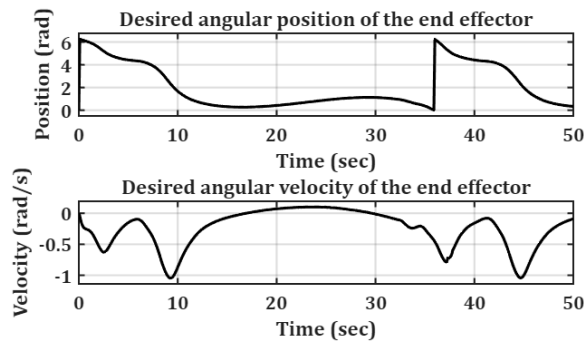


Figure 8.10: Log-arm system path position and path velocity for the 0.25 m radius case

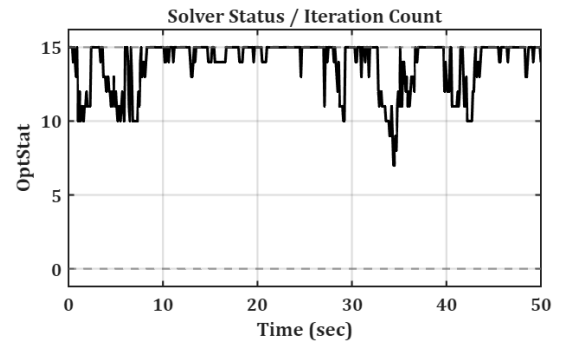


Figure 8.11: Number of interior-point iterations for the 0.25 m radius case

A notable feature of this case is the optimized path velocity shown in Fig. 8.10. During recovery portions of the maneuver, the controller temporarily slows down and changes the direction of path progression. This behavior is possible because the path coordinate and path velocity are optimized within the NMPC

problem. A formulation with prescribed fixed path speed would not naturally provide this freedom and could force the system to continue progressing along the path even when the constrained underactuated dynamics require temporary reversal or slowdown. The solver iteration history in Fig. 8.11 shows that several MPC updates reach the maximum iteration budget of 15 iterations. This is consistent with the difficulty of the $r_c = 0.25$ m case, where d_3 remains near its imposed limits for significant portions of the maneuver. The active prismatic constraint makes the NMPC subproblems more tightly constrained and increases the nonlinear solver effort.

8.3.3: Large-Radius Circular Path

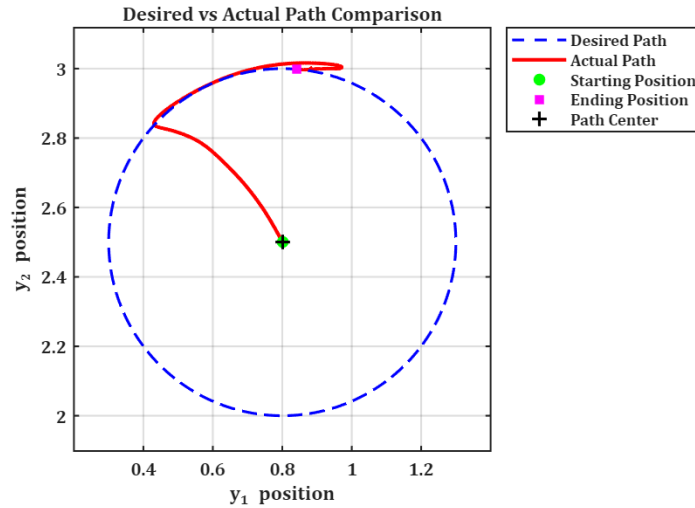


Figure 8.12: Path tracking performance of the log-arm system for the 0.5 m radius case

For $r_c = 0.5$ m, the controller drives the end effector to the desired path, but sustained traversal around the full circle is not maintained, as shown in Fig. 8.12. Instead, the output settles at a point on the path after the transient.

For $r_c = 0.5$ m, the state histories in Fig. 8.13 show that d_3 reaches its upper limit during the transient but is not saturated at the final stationary point. This indicates that the difficulty is not determined only by the final value of d_3 , but by the feasibility of the future motion required for continued traversal. Continued progression around the larger circle may drive d_3 back toward its upper limit, making it difficult for the optimizer to find a stable progressing solution that both follows the path and remains away from the prismatic bound. This provides a plausible explanation for why the controller converges to a stationary point on the path: the stationary solution remains feasible, whereas forced progression could lead to repeated recovery attempts, oscillatory behavior, or future constraint violation. Since path progression is imposed as a soft objective rather than a hard constraint, the NMPC formulation allows this feasible stationary response instead of forcing continued traversal.

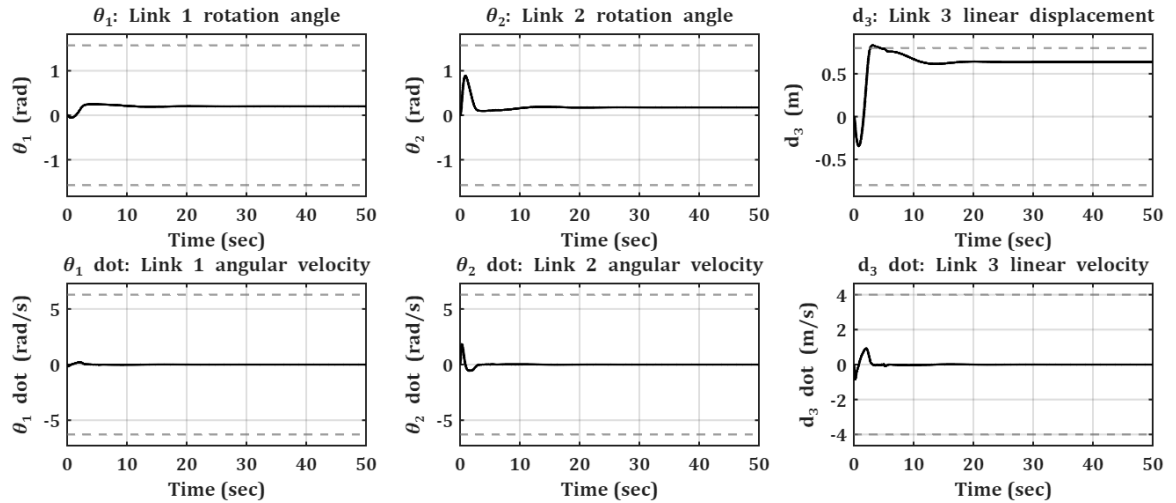


Figure 8.13: Evolution of system log-arm states for 0.5 m radius case

The control inputs in Fig. 8.14 remain bounded during this response,

indicating that the stationary behavior is not caused by loss of control authority.

Figure 8.15 further shows that the path velocity decreases after the transient as the optimizer settles at the stationary path point. The solver iteration count in Fig. 8.16 reaches the 15-iteration limit only during the initial path-merging transient; after this phase, it settles to approximately 10 iterations per MPC update.

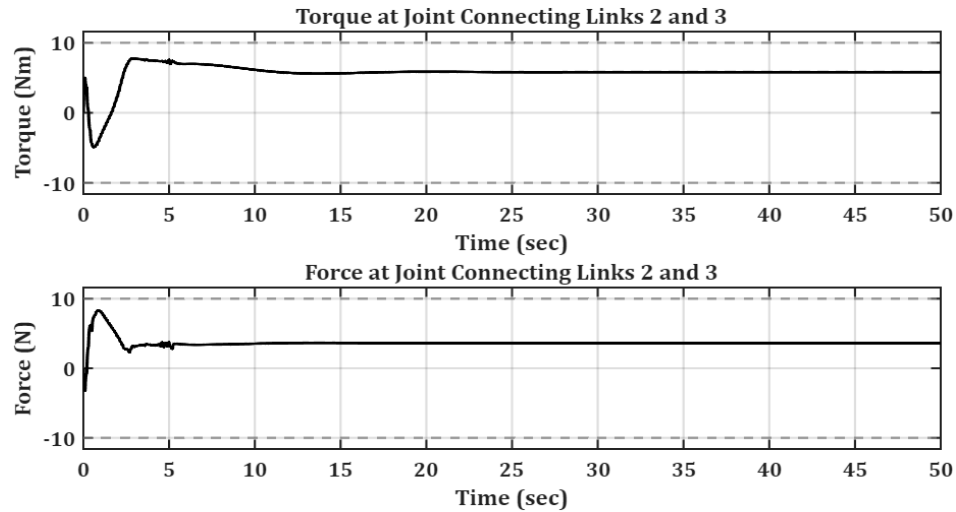


Figure 8.14: Log-arm system control inputs for the 0.5 m radius case

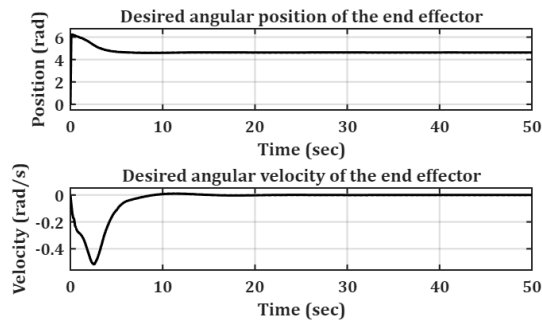


Figure 8.15: Log-arm system path position and path velocity for the 0.5 m radius case

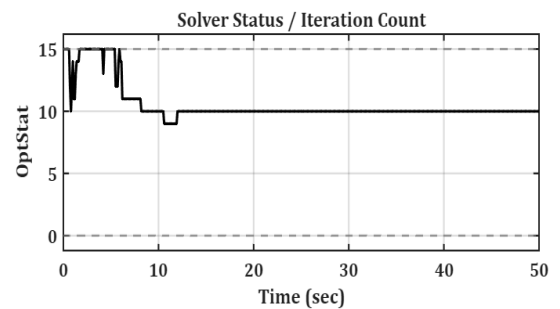


Figure 8.16: Number of interior-point iterations for the 0.5 m radius case

8.3.4 Supplementary Bound Relaxation Study

To clarify the role of the imposed state and input bounds in the previous

simulations, additional studies were performed with relaxed prismatic displacement and torque bounds. In the original problem setup, the prismatic displacement was constrained by $-0.8 \text{ m} \leq d_3 \leq 0.8 \text{ m}$. The supplementary results in this section assess whether the tracking limitations observed in Sections 8.3.2 and 8.3.3 are caused primarily by the imposed state and input bounds, rather than by an inherent inability of the NMPC formulation to generate path-following motion.

For the $r_c = 0.25 \text{ m}$ case, the symmetric prismatic displacement bound was relaxed to $-1.2 \text{ m} \leq d_3 \leq 1.2 \text{ m}$. The resulting path-tracking performance is shown in Fig. 8.17.

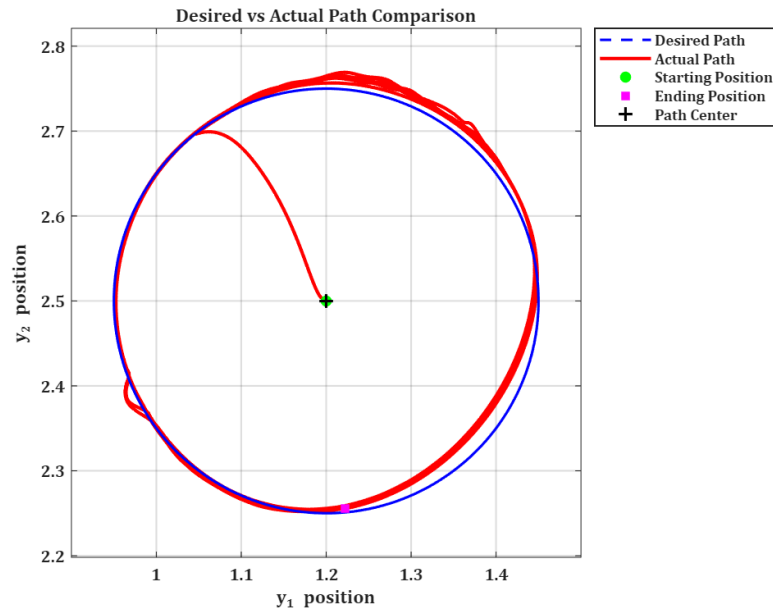


Figure 8.17: Path tracking performance of the log-arm system for the 0.25 m radius case with relaxed prismatic displacement bounds.

Compared with the original case in Fig. 8.7, the closed-loop trajectory remains much closer to the desired circular path. The corresponding state histories in Fig. 8.18 show that d_3 slightly exceeds the previous 0.8 m limit during the maneuver. Although

the exceedance of the original limit is small, the improvement in path-following performance is significant. This indicates that the degraded tracking performance in the original $r_c = 0.25$ m case was mainly caused by the prismatic displacement bound being restrictive for the required motion. The control inputs in Fig. 8.19 remain bounded, showing that the improved path following is achieved without loss of control regularity.

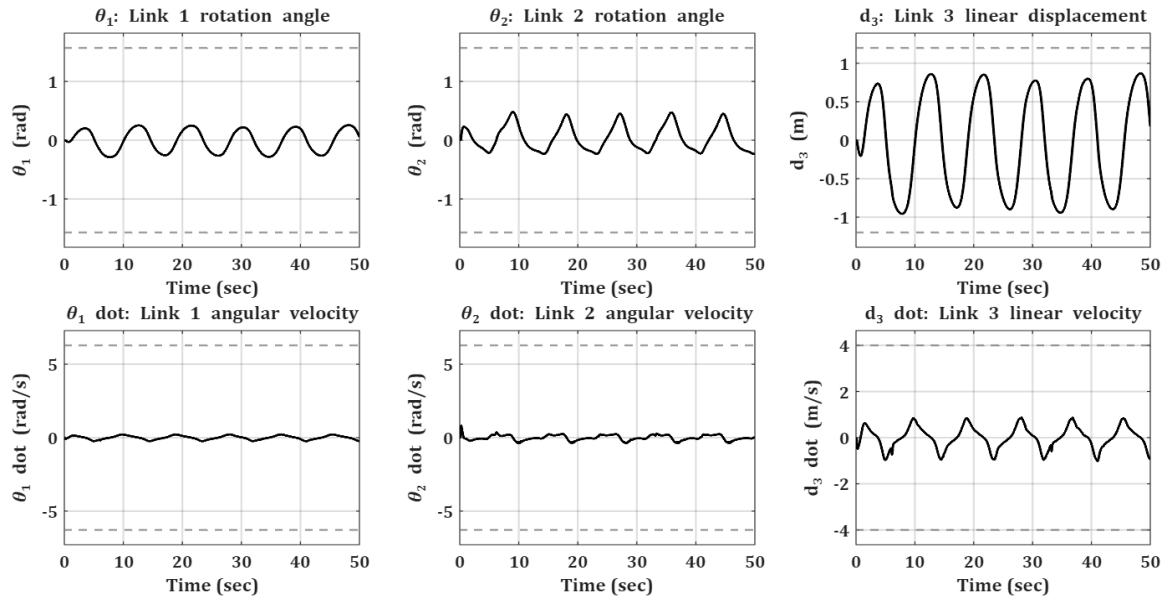


Figure 8.18: Evolution of system log-arm states for the 0.25 m radius case with relaxed prismatic displacement bounds.

For the $r_c = 0.5$ m case, relaxing the symmetric prismatic displacement bound to $-1.2 \text{ m} \leq d_3 \leq 1.2 \text{ m}$ was still not sufficient to obtain consistently good path-following performance. Depending on the tuning, the system either converged to a stationary point on the path or traversed the path with poor tracking accuracy. However, when the symmetric prismatic displacement bound was further relaxed to $-1.6 \text{ m} \leq d_3 \leq 1.6 \text{ m}$ and the torque bounds were also relaxed to ± 20 Nm, the system achieved good path-following performance, as shown in Fig. 8.20. The

corresponding state histories in Fig. 8.21 show that the larger prismatic range is used during the maneuver, while the control inputs in Fig. 8.22 remain bounded under the relaxed torque limits.

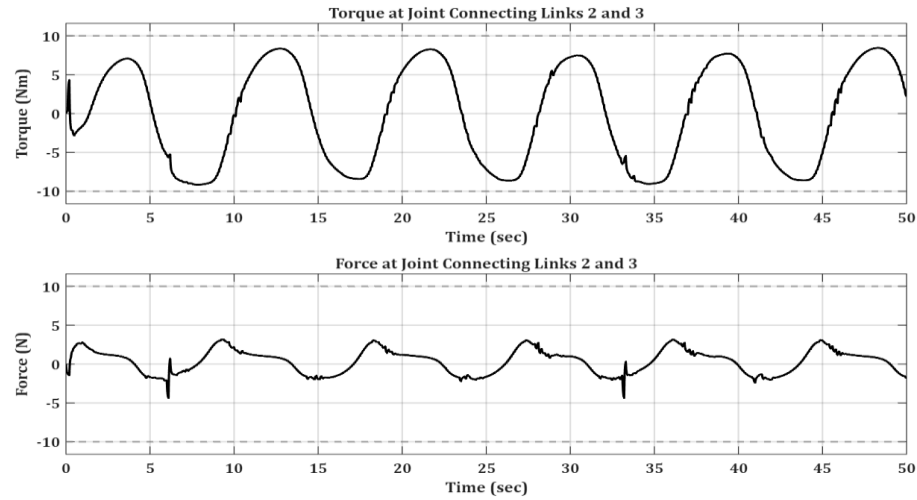


Figure 8.19: Log-arm system control inputs for the 0.25 m radius case with relaxed prismatic displacement bounds.

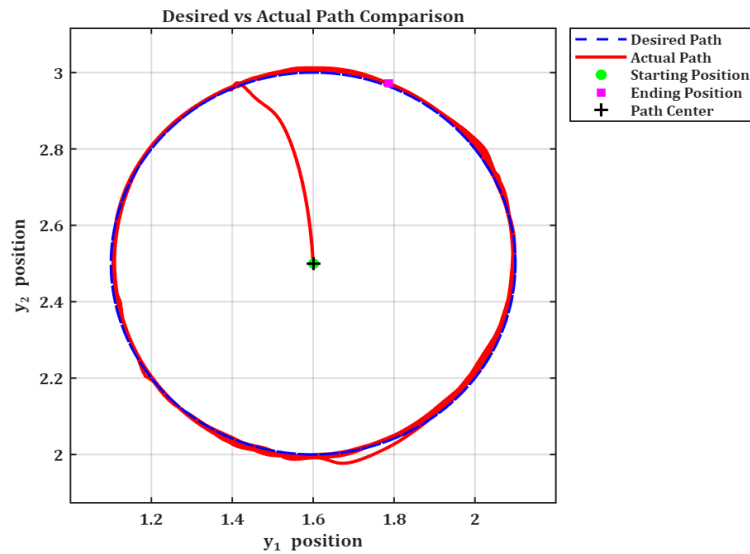


Figure 8.20: Path tracking performance of the log-arm system for the 0.5 m radius case with relaxed prismatic displacement and torque bounds.

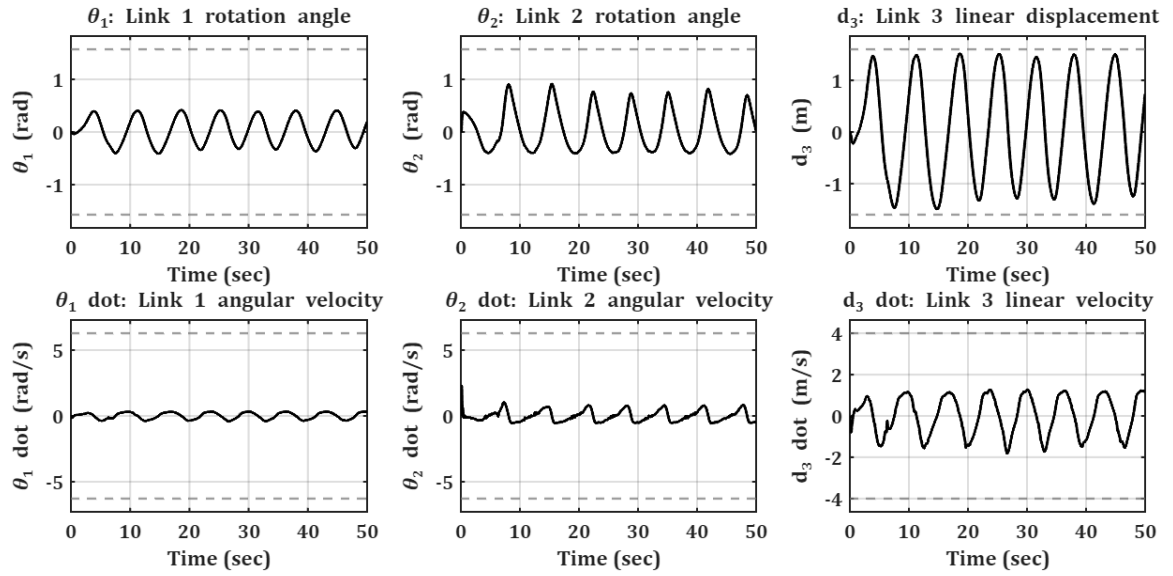


Figure 8.21: Evolution of system log-arm states for the 0.5 m radius case with relaxed prismatic displacement and torque bounds.

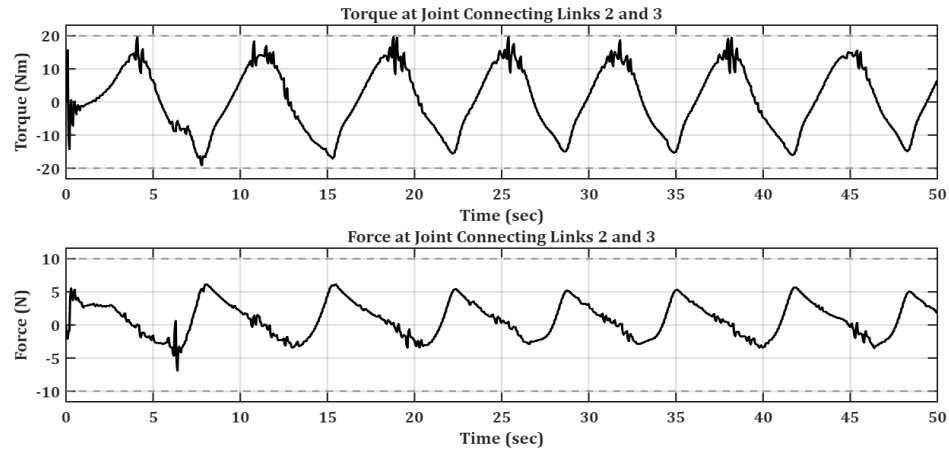


Figure 8.22: Log-arm system control inputs for the 0.5 m radius case with relaxed prismatic displacement and torque bounds.

These supplementary simulations support the interpretation of the original results in Sections 8.3.2 and 8.3.3. For $r_c = 0.25$ m, relaxing the prismatic displacement bound significantly improves tracking, showing that the original limitation was mainly the available prismatic range. For $r_c = 0.5$ m, good tracking is recovered only after relaxing both the prismatic displacement and torque bounds.

Therefore, the stationary response in the original $r_c = 0.5$ m case is interpreted as a constrained local solution of the soft-progress NMPC problem, not as a fundamental inability to follow the circular path.

8.4 Chapter Summary

The log-arm results demonstrate the value of unified NMPC path following rather than prescribing fixed path progression externally. Because the system is severely underactuated and the end-effector motion depends on the coupled evolution of θ_1 , θ_2 , and d_3 , treating the path coordinate and path velocity as decision variables allows the optimizer to slow down, reverse, or continue path progression based on the feasible motion available to the constrained system.

The simulations show increasing sensitivity to the imposed prismatic displacement and actuator bounds as the path radius increases. The supplementary bound-relaxation study confirms this: relaxing the symmetric prismatic displacement bound significantly improves the $r_c = 0.25$ m case, while good path following for the $r_c = 0.5$ m case is recovered only after relaxing both prismatic displacement and torque bounds. Thus, the stationary response under the original bounds is interpreted as a feasible local optimum of the constrained soft-progress NMPC problem, not as inability to follow the circular path.

The results also highlight the benefit of second-order optimization in the proposed solver. The NMPC problems are solved within the fixed 15-iteration budget, with the iteration count reaching the maximum mainly during the initial path-merging transient, and the solver convergence rate remains above 95%.

Therefore, the observed deviations and stationary behavior are interpreted as constrained optimal responses when balancing path tracking, path progression, actuator effort, and state limits.

This example is difficult for conventional fixed-speed tracking or decoupled planner-controller approaches. The unified MPPFC formulation incorporates dynamics, constraints, path coordinate, and path velocity in a single optimization problem, producing meaningful closed-loop behavior even when perfect traversal is not possible under the original bounds.

CHAPTER 9 - CONCLUSIONS, COMPUTATIONAL COMPLEXITY, AND FUTURE WORK

9.1 Computational Complexity Analysis

The computational efficiency of the proposed unified MPPFC framework comes from three main components: analytical collision-constraint derivatives, reference-path smoothing, and structure-exploiting primal-dual search-direction computation.

For the analytical collision-avoidance terms, let n_o denote the number of active body-obstacle pairs at each prediction step, and let $n_{q,o}$ denote the number of generalized coordinates affecting pair o . For each active pair, the closest-point sensitivity computation has fixed $O(1)$ complexity because each smooth convex body is described by a fixed number of boundary parameters: one in 2D and two in 3D. This local calculation is independent of physical obstacle size, while the number of active pairs is accounted for through n_o . The configuration-space Hessian contains $n_{q,o}^2$ entries. Therefore, the dominant per-pair second-order derivative assembly cost scales as $O(n_{q,o}^2)$. Over n_p prediction steps, the collision-derivative assembly cost per interior-point iteration becomes

$$O_{col} = O\left(n_p \sum_{o=1}^{N_o} n_{q,o}^2\right)$$

In the worst case, if every active pair depends on all n_q generalized coordinates, this reduces to

$$O_{col} = O(n_p n_o n_q^2).$$

Therefore, over n_{IPM} interior-point iterations, the total collision-derivative assembly cost is

$$O_{col,total} = O(n_{IPM}n_p n_o n_q^2) \quad (9.1)$$

This worst-case bound is conservative because, in articulated systems, each body pair usually depends only on a subset of upstream coordinates in the kinematic chain.

The reference-path smoothing method consists of RDP waypoint reduction followed by composite cubic Bézier fitting. If the original waypoint sequence contains m points and the reduced sequence contains $\tilde{m}$ retained points, the RDP step has worst-case complexity

$$O(m^2)$$

although typical paths are closer to

$$O(m \log m)$$

because many segments terminate early [86], [87]. The Bézier fitting step solves a block-tridiagonal system with block dimension proportional to the path dimension d [88], giving

$$O(\tilde{m}d^3)$$

Since d is usually fixed and small, this stage is nearly linear in the number of retained waypoints. Thus, the expected path-smoothing cost is

$$O_{path} = O(m \log m) + O(\tilde{m}d^3) \quad (9.2)$$

For the primal-dual search direction, the proposed solver exploits the sparse horizon structure instead of solving a dense full-horizon Newton system. Let n_χ be

the retained primal dimension per stage after local eliminations and let n_Λ be the retained equality-multiplier dimension per stage. A dense full-horizon solve would scale approximately as

$$O((n_p(n_\chi + n_\Lambda))^3)$$

Algorithm 6.1 instead factorizes stagewise condensed curvature blocks and solves a block-tridiagonal Schur-complement system for the retained equality multipliers, giving the per-iteration cost

$$O_{IPM} = O(n_p(n_\chi + n_\Lambda)^3)$$

consistent with fast MPC and interior-point MPC formulations [46], [53], [54]. The eliminated slack, relaxation, inequality-multiplier, and bound-dual directions are then recovered stagewise with linear cost. Therefore, over n_{IPM} interior-point iterations

$$O_{IPM,total} = O(n_{IPM}n_p(n_\chi + n_\Lambda)^3) \quad (9.3)$$

Thus, for fixed stage dimensions, the search-direction computation scales linearly with the prediction horizon and with the number of interior-point iterations. The dual variables and softened inequality constraints increase the per-stage dimension but do not change the linear-in-horizon structure because their directions are either retained in the structured system or recovered by elimination.

9.2 Summary of Contributions

This dissertation developed a unified nonlinear Model Predictive Path-Following and Collision Avoidance Control framework for robotic systems operating in constrained environments. The central objective was to formulate path following,

obstacle avoidance, nonlinear dynamics, actuator limits, and path progression within a single predictive optimization problem. Rather than prescribing the path coordinate externally, the proposed formulation treats the path variable and its rate of evolution as decision variables. This allows the controller to regulate physical motion and path progression simultaneously, including slowing, pausing, or resuming progress when required by feasibility, state constraints, actuator limits, or collision-clearance requirements.

A major contribution of this work is the integration of differentiable minimum-distance constraints into the MPPFC formulation. Robot bodies, links, vehicle components, and obstacles are represented using smooth convex surface models, allowing collision avoidance to be expressed directly through Euclidean clearance constraints. Analytical gradients and Hessians of the minimum-distance function were derived by differentiating the closest-point optimality conditions. These derivatives account for rigid-body motion, closest-point variation, and surface curvature, and they are propagated through kinematic chains to obtain sensitivities with respect to generalized coordinates and predicted states. The numerical validation showed that the analytical derivatives agree with finite-difference approximations while providing substantial computational speedups, especially for Hessian evaluation. This confirms the value of using analytical collision sensitivities instead of finite-difference approximations in second-order optimization.

The dissertation also developed a computational framework required to solve the resulting nonlinear programs efficiently. A structure-exploiting primal-

dual interior-point method was formulated for the softened unified MPPFC problem. The solver uses scaling, barrier terms, slack and relaxation variables, inertia regularization, filter globalization, second-order correction, warm starting, and sparse KKT-system organization. By preserving the stagewise horizon structure of the MPPFC problem, the method allows dynamics, path-following objectives, bounds, and collision constraints to be handled within a single sparse optimization cycle. This computational structure is essential because collision-aware MPPFC problems can otherwise become expensive when many body–obstacle interactions are included.

The proposed framework was evaluated on two representative nonlinear systems. The articulated tractor–trailer application demonstrated collision-aware path following in cluttered environments with static obstacles, dynamic obstacles, actuator limits, articulation effects, and multiple body–obstacle interactions. The closed-loop simulations showed that the controller can recover from large initial path-tracking errors, avoid obstacles, satisfy constraints, and return toward the reference path without requiring a separate local planner or rule-based maneuver logic. The results also showed that retaining full analytical collision-constraint Hessians improved convergence robustness compared with a Gauss–Newton approximation, particularly when collision constraints became active. A longer-horizon stress test further demonstrated that the solver maintained strong convergence behavior under a significantly larger optimization problem.

The underactuated log-arm application demonstrated the importance of

optimizing the path coordinate for constrained nonlinear systems whose feasible motion does not necessarily permit fixed-speed traversal. In this case, the controller was able to adjust path progression according to the available dynamics and constraints. The smaller and nominal-radius cases showed sustained path traversal, while the most constrained case showed convergence to a feasible stationary response when continued path progression became unfavorable. This behavior illustrates a key feature of the unified MPPFC formulation: path progression is encouraged through the objective, but feasibility and constraint satisfaction remain dominant. Therefore, the controller can produce meaningful constrained behavior even when perfect traversal is not possible.

Overall, this dissertation connects geometric collision modeling, analytical sensitivity analysis, smooth path generation, nonlinear path-following control, and sparse second-order optimization into a unified framework. The results show that differentiable minimum-distance constraints with analytical derivatives can be embedded directly into a unified MPPFC problem and solved efficiently using a structure-exploiting primal–dual interior-point method. The framework provides a foundation for fast collision-aware predictive control of robotic and articulated systems, while also identifying important directions for future work.

9.3 Limitations of the Present Work

The proposed framework has several limitations that define the scope of the present dissertation. First, the complete collision-aware MPPFC framework is validated through numerical simulations rather than experiments on a physical

robotic or vehicle platform. Although the simulations include nonlinear articulated dynamics, actuator and state constraints, static and dynamic obstacles, and limited solver-iteration budgets, they do not capture all effects associated with hardware implementation. Real-world deployment would additionally require compiled implementation, integration with perception, localization, state estimation, low-level control, communication interfaces, and safety-monitoring systems, together with experimental calibration and validation.

Second, the formulation is nominal and deterministic. The controller assumes that the system state, model parameters, reference path, and obstacle states are available with sufficient accuracy. Modeling errors, sensor noise, state-estimation errors, external disturbances, actuator delays, perception latency, and uncertainty in predicted obstacle motion are not explicitly represented. The conservative obstacle boundaries used in tractor-trailer simulations in Chapter 7 provide a geometric safety margin but do not constitute a formal robustness guarantee. Model parameters are also treated as fixed rather than being estimated or adapted online.

Third, the second-order optimization framework requires the system dynamics, body-pose maps, objective functions, and constraint functions used within the nonlinear program to possess the required smoothness. The input reference waypoints themselves are not assumed to define a smooth path; they are only required to be indexed in path-traversal order, after which the smooth reference path is constructed using the method developed in Chapter 5. Nonsmooth physical phenomena such as impacts, contact-mode transitions, discontinuous friction,

switching dynamics, logical constraints, and discrete control decisions are not represented directly and would require smooth approximations, hybrid formulations, or mixed-integer extensions.

Fourth, the collision formulation assumes separated smooth convex bodies with parametric boundary-set descriptions and positive clearance. It therefore does not directly accommodate contact, penetration, nonsmooth polygonal or mesh geometries, or sensor-based representations such as point clouds, voxel maps, occupancy grids, and discrete distance fields. These representations must first be converted into suitable smooth parametric body models.

Fifth, the continuous-time dynamics are discretized using trapezoidal integration. Prediction accuracy consequently depends on the selected discretization interval and the time scales of the modeled dynamics. Systems with stiff dynamics, rapid transients, or discontinuous inputs may require a finer prediction grid or a higher-order integration method, increasing the optimization size and computational cost.

Sixth, the present framework is limited to rigid-body robotic systems whose configurations and body poses can be described by finite-dimensional kinematic and dynamic models. Soft robots, deformable bodies, and continuum robots are not directly covered because their geometries and dynamics depend on distributed deformation states. Applying the framework to such systems would require suitable finite-dimensional approximations and corresponding collision-distance derivatives.

Seventh, the dissertation does not establish recursive feasibility or formal

closed-loop stability for the general nonlinear collision-aware MPPFC problem. It also does not guarantee convergence from arbitrary initial conditions, near singular configurations, or operating conditions in which the available actuator authority is insufficient to satisfy the path-following and collision-avoidance requirements.

Eighth, the unified MPPFC problem is nonlinear and nonconvex because of the system dynamics, collision constraints, and articulated geometry. The proposed primal-dual method does not guarantee global optimality and may converge to a locally optimal or stationary solution. Performance can depend on the initial trajectory, shifted warm start, prediction horizon, obstacle configuration, scaling, and regularization parameters. Transitions between substantially different avoidance maneuvers or trajectory homotopy classes may require more informed initialization or higher-level guidance.

Finally, the finite prediction horizon limits the controller's ability to account for path and obstacle interactions beyond the optimization window. Increasing the horizon improves anticipatory behavior but also increases the size and computational cost of the nonlinear program.

9.4 Future Work

9.4.1 Extension to General Geometric Environment Models

The distance-derivative framework developed in this dissertation is based on smooth convex body representations. This assumption is required to obtain well-defined analytical gradients and Hessians under the proposed formulation, but it does not cover all geometry types encountered in robotic applications.

One important extension is to non-smooth geometry. Many obstacles are more naturally represented as polytopes, boxes, or mesh models. For these shapes, the distance is still well defined, but the closest feature can switch between a face, edge, or vertex as the configuration changes. These switches can make the gradient discontinuous and the Hessian undefined. A practical way to address this issue is to replace non-smooth geometry with smooth convex approximations, such as super-ellipsoids or other analytic bounding surfaces. This would preserve the differentiability needed by the proposed NMPC solver.

A second extension is to signed-distance modeling. The current formulation assumes separated bodies and positive clearance. However, some applications may require contact handling, compliant interaction, or small allowable penetration. Signed-distance and distance-field representations are more suitable in these cases because they distinguish separation, contact, and penetration and provide continuous geometric representations for proximity modeling [30], [112], [113]. Extending the present analytical derivatives to signed distance would require smooth and well-conditioned definitions inside the penetration region, especially near contact transitions where the closest-point mapping can change abruptly [40]–[42]. Such an extension could improve the robustness of trajectory optimization and MPC in settings where strict non-intersection is too restrictive [34], [35], [106].

A third extension is to sensor-based environment models. Real systems often perceive the environment as point clouds, voxel maps, occupancy grids, or signed distance fields rather than explicit convex surfaces. Future work could connect the

present framework to these representations by fitting local smooth convex primitives, using local quadratic surface approximations, or extracting derivatives from precomputed distance fields. This would make the method more directly applicable to real-time robotic navigation using perception data.

9.4.2 Pairwise Collision Penalties

Another useful extension is to add pairwise collision-avoidance penalties to the objective function, while still keeping the collision-clearance constraints. The current formulation enforces safety using

$$G_c = d_{\min} - d \leq 0$$

where $d_{\min}$ is the required clearance and d is the actual distance. This constraint is necessary for safety, but the optimizer may react mainly when the constraint becomes active or nearly active. For short prediction horizons, this can delay obstacle-avoidance behavior. To encourage earlier avoidance, a smooth velocity-based penalty can be added for each host-target pair. Let

$$g_i = \nabla G_{c,i}$$

be the collision-constraint gradient in Cartesian position coordinates, and define the local normal direction as

$$n_i = \frac{g_i}{\|g_i\| + \epsilon_n}.$$

Since $G_c = d_{\min} - d$, this normal points approximately from the host body toward the target. If v_h and v_t are the host and target velocities at the closest-point pair, define the relative velocity as

$$v_{\text{rel},i} = v_h - v_t$$

The normal and tangential components are

$$v_{N,i} = n_i^\top v_{\text{rel},i}$$

$$v_{T,i} = (I - n_i n_i^\top) v_{\text{rel},i}$$

$$s_{T,i} = \| v_{T,i} \|$$

A positive $v_{N,i}$ corresponds to motion toward the target and therefore increasing collision risk. A smooth pairwise penalty can then be written as

$$J_{\text{col},i} = \frac{1}{2} \lambda_N w_i \phi(v_{N,i})^2 + \frac{1}{2} \lambda_T w_i \phi(V_{T,\text{des}} - s_{T,i})^2 \quad (9.4)$$

where $\phi(\cdot)$ is a smooth hinge function, w_i is a smooth activation weight based on proximity, λ_N penalizes inward normal motion, and λ_T penalizes insufficient tangential escape speed. The stage cost is

$$J_{\text{col},k} = \sum_i J_{\text{col},i} \quad (9.5)$$

This term does not replace the collision constraints. Instead, it shapes the optimization problem so that the controller begins avoiding obstacles earlier and approaches active collision constraints less abruptly. Because the penalty uses the relative velocity $v_h - v_t$, it can distinguish between targets moving away from the host and targets moving toward it. This can improve solver robustness and is especially useful for short-horizon MPC and dynamic environments.

9.4.3 Multi-End-Effector and Multi-Path Following

A natural extension is to test the proposed MPPFC formulation on systems with multiple end effectors or task outputs. The formulation in Chapter 3 already supports multiple output maps and path corridors, allowing different end effectors

or task outputs to track separate reference paths within prescribed admissible regions. However, this capability was not evaluated in the applications presented in this dissertation.

Future work may therefore apply the framework to systems where multiple outputs may be coordinated under shared dynamics, actuator limits, corridor constraints, and collision-avoidance constraints. Examples include dual-arm manipulators, and articulated robots with multiple controlled body points.

Another related extension is multi-reference tracking for a single output. For example, an autonomous vehicle may track candidate corridors for staying in the current lane, changing lanes, or preparing for a future maneuver. Solving these corridor-constrained references within the MPPFC framework could provide feasibility-aware cost and constraint information to a higher-level planner, helping it select the most suitable maneuver.

9.4.4 Solver Robustness Enhancements

Additional robustness enhancements can also be added to the structure-exploiting interior-point solver. In particular, centrality-correction methods, such as Mehrotra's predictor-corrector method and Gondzio's multiple centrality corrections, are natural candidates when the Newton direction is strongly limited by fraction-to-the-boundary step restrictions. These corrections can improve the admissible step length and help maintain centrality, while still using the existing sparse Newton-system factorization and filter line-search framework [49], [114], [115].

A second direction is to incorporate trust-region iterations when the Newton step becomes unreliable due to poor local curvature, strong nonlinearities, or difficult collision-constraint activity. A trust-region mechanism could limit the step to a region where the local model is reliable, improving robustness during challenging transients without replacing the sparse Newton structure.

Finally, the warm-start procedure can be further improved by exploiting robotics-specific structure. Instead of only shifting the previous horizon solution, future warm starts could use kinematic prediction, obstacle motion estimates, path-progress trends, and active collision-pair information to initialize states, controls, slacks, and dual variables more intelligently. Such tailored warm starts could reduce iteration count, improve feasibility recovery, and make the solver more reliable for real-time implementation.

9.4.5 Lyapunov-Based Stability Constraints

Another important direction is to add Lyapunov-based stability constraints to the unified MPPFC problem. Future work could define a Lyapunov function for the path-following output using quantities such as lateral error, heading error, path-coordinate error, and possibly speed-tracking error. This function could then be included in the NMPC problem through an inequality constraint that enforces decrease along the prediction horizon. For example, let e_k denote the selected path-following output error at prediction step k , and let $V(e_k)$ be a positive-definite Lyapunov function. A possible decrease condition is

$$V(e_{k+1}) - V(e_k) \leq -\alpha(\|e_k\|),$$

where $\alpha(\cdot)$ is a positive function. In practice, this condition could also be softened as

$$V(e_{k+1}) - V(e_k) \leq -\alpha(\|e_k\|) + \sigma_k, \quad \sigma_k \geq 0 \quad (9.6)$$

with σ_k included as a relaxation variable and penalized in the objective, so that violations of the nominal Lyapunov decrease condition are discouraged.

If such a Lyapunov function can be constructed and the monotonic decrease condition can be enforced recursively, then recursive feasibility of the Lyapunov-constrained MPPFC problem could provide a path toward proving input-to-output stability of the closed-loop system. In this setting, the controlled path-following outputs would remain bounded under bounded disturbances, reference variations, and admissible inputs. If the output map is chosen to include the relevant tracking, velocity, and internal-state components, or if an appropriate detectability condition can be established, this result could also be extended toward boundedness of the full closed-loop state. This would strengthen the theoretical foundation of the proposed framework by connecting recursive feasibility, constrained path following, and closed-loop stability within a single optimization-based formulation.

9.4.6 Reinforcement Learning-Assisted Unified MPPFC

A natural direction for future work is to incorporate reinforcement learning (RL) into the unified MPPFC framework as a supporting component rather than a replacement for the optimization-based controller. The unified MPPFC formulation is suitable for this integration because its nonlinear optimization problem already captures dynamics, path progress, constraints, path-following objectives, and collision-avoidance requirements. In this setting, RL can be used to improve

computational efficiency, optimization initialization, and long-term decision making, while the MPPFC solver continues to guarantee constraint satisfaction through the underlying model.

One possible extension is to use RL or imitation learning to generate warm-start trajectories for the unified MPPFC solver. In a receding-horizon framework, the initial guess is usually obtained by shifting the previous optimal solution forward by one time step. Although this strategy works well when the optimal solution changes gradually, it can become less effective when collision-avoidance constraints become active or inactive, when the optimizer needs to move between different homotopy classes, or when the current solution is close to an unfavorable local minimum. Since path following and collision avoidance are handled simultaneously within the same nonconvex optimization problem, these situations can significantly affect solver performance. A learned policy could be trained offline using MPPFC simulation data to predict an initial primal trajectory, and potentially an initial dual trajectory, from the current state, path-progress state, local path information, and obstacle configuration. The optimizer would still compute the final solution but would begin from a more informed starting point. Previous studies on learning-assisted MPC warm starts have shown that neural networks can provide useful initializations when conventional shifted solutions are no longer adequate [116]. More broadly, learning-based warm-start techniques have been shown to reduce optimization time and iteration counts by predicting high-quality initial guesses [117]. Within the unified MPPFC framework, the effectiveness of this approach could be assessed

through metrics such as convergence rate, solver iterations, KKT residuals, and closed-loop feasibility.

Another promising direction is the use of RL for adaptive tuning of MPPFC parameters. The unified formulation contains several tuning parameters, including weights associated with path following, path progress, control effort, soft constraints, and collision-clearance requirements. These parameters balance tracking accuracy, path progress, control smoothness, obstacle avoidance, and numerical performance. In this dissertation, these values are selected manually and remain fixed across different scenarios. However, the most suitable parameter values may vary depending on the operating conditions, path geometry, surrounding obstacles, and constraint activity. RL could be used to learn a supervisory policy that adjusts these parameters online according to the characteristics of the current problem. Since RL would only modify controller settings, the underlying MPPFC optimization framework would remain unchanged. Previous research has demonstrated the use of RL for automated tuning of nonlinear MPC weights [118], and other studies have explored MPC as a structured policy whose parameters can be optimized through reinforcement learning [119]. Applying these ideas to unified MPPFC could improve adaptability while preserving the analytical gradients, Hessians, and sparse optimization structure developed in this work.

A third possibility is to use RL or approximate dynamic programming to learn terminal cost functions for the unified MPPFC problem. Because the optimization is performed over a finite prediction horizon, the controller may not always account

for the long-term effects of current decisions. This becomes particularly relevant when path following, path progress, and collision avoidance interact over time scales that extend beyond the prediction horizon. A learned terminal value function could approximate the cost-to-go beyond the horizon using the terminal state, path-progress state, and local constraint environment. Incorporating such a terminal cost would encourage decisions that are not only feasible in the short term but also beneficial for future performance. Existing work on learning-based terminal costs for MPC has demonstrated the potential of approximate value iteration and related approaches for capturing long-term behavior [119]. For unified MPPFC, a learned terminal cost could allow shorter prediction horizons while maintaining desirable long-term performance and reducing computational requirements.

Overall, RL is best viewed as a complementary tool within the unified MPPFC framework rather than as substitute. The main strength of the proposed formulation lies in its ability to combine system dynamics, path evolution, and collision avoidance within a single constrained nonlinear optimization problem while exploiting derivative information and sparsity for efficient computation. A practical approach is to use RL to address aspects that are difficult to design manually, such as solver initialization, adaptive parameter tuning, and long-horizon cost approximation. In this way, data-driven learning can be incorporated without sacrificing the model-based and safety-oriented characteristics of unified MPPFC.

9.4.7 Adaptive MPPFC with Moving Horizon Estimation

Another natural extension of this research is adaptive MPPFC for applications

in which model parameters are time-varying, uncertain, or only partially known. In such cases, online estimation methods such as recursive least squares or moving horizon estimation (MHE) could be used to estimate selected model parameters during operation. The estimated parameters could then be incorporated into the MPPFC prediction model at each control update, allowing the controller to adapt its control actions as the system dynamics or operating conditions change.

Moving horizon estimation is particularly attractive in this setting because it estimates states and parameters over a finite measurement window while enforcing model dynamics and constraints. Moreover, the structure-exploiting primal-dual optimization framework developed in this thesis could be extended to the moving-horizon estimation problem, since both MPPFC and MHE lead to sparse finite-horizon nonlinear programs [47], [53]. This could enable fast estimation and control updates within a unified adaptive NMPC architecture, improving robustness for systems with changing payloads, uncertain inertial properties, actuator degradation, tire-road variations, or other slowly varying dynamics.

REFERENCES

- [1] Kavraki, L.E., LaValle, S.M. (2016). Motion Planning. In: Siciliano, B., Khatib, O. (eds) Springer Handbook of Robotics. Springer Handbooks. Springer, Cham. doi: 10.1007/978-3-319-32552-1_7.
- [2] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotović, "Path-following for nonminimum phase systems removes performance limitations," IEEE Trans. Autom. Control, vol. 50, no. 2, pp. 234–239, Feb. 2005.
- [3] A. Isidori, Nonlinear Control Systems, 3rd ed. London, U.K.: Springer-Verlag, 1995.
- [4] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotović, "Performance limitations in reference tracking and path following for nonlinear systems," Automatica, vol. 44, no. 3, pp. 598–610, Mar. 2008.
- [5] A. P. Aguiar, D. B. Dačić, J. P. Hespanha, and P. V. Kokotović, "Path-following or reference-tracking? An answer relaxing the limits to performance," in Proc. 5th IFAC/EURON Symp. Intelligent Autonomous Vehicles (IAV), Lisbon, Portugal, Jul. 2004.
- [6] J. Hauser and R. Hindman, "Maneuver regulation from trajectory tracking: Feedback linearizable systems," in Proc. IFAC Symp. Nonlinear Control Systems Design (NOLCOS), Lake Tahoe, CA, USA, Jun. 1995, pp. 595–600.
- [7] P. Encarnação and A. M. Pascoal, "3D path following control of autonomous underwater vehicles," in Proc. 39th IEEE Conf. Decision and Control (CDC), Sydney, Australia, Dec. 2000, pp. 2977–2982.
- [8] S. Yu, X. Li, H. Chen, and F. Allgöwer, "Nonlinear model predictive control for path following problems," Int. J. Robust Nonlinear Control, vol. 24, no. 17, pp. 2599–2611, Jan. 2014.
- [9] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," IEEE Trans. Autom. Control, vol. 61, no. 4, pp. 1026–1031, Apr. 2016.
- [10] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in Proc. IEEE Conf. Decis. Control (CDC), Shanghai, China, Dec. 2009, pp. 8642–8647.
- [11] T. Faulwasser, V. Hagenmeyer, and R. Findeisen, "Optimal exact path-following

for constrained differentially flat systems,” in Proc. 19th IFAC World Congr., Cape Town, South Africa, Aug. 2014, pp. 5767–5772.

- [12] T. Faulwasser, Optimization-Based Solutions to Constrained Trajectory-Tracking and Path-Following Problems. Aachen, Germany: Shaker Verlag, 2013.
- [13] T. Faulwasser, T. Weber, P. Zometa and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," in IEEE Transactions on Control Systems Technology, vol. 25, no. 4, pp. 1505-1511, July 2017, doi: 10.1109/TCST.2016.2601624.
- [14] I. Sánchez, A. Ferramosca, G. Raffo, A. H. González, and A. D’Jorge, “Obstacle avoiding path following based on nonlinear model predictive control using artificial variables,” in Proc. Int. Conf. Adv. Robot., Dec. 2019, pp. 245–250.
- [15] I. Sánchez, A. D’Jorge, G. V. Raffo, A. H. González, and A. Ferramosca, “Nonlinear model predictive path following controller with obstacle avoidance,” J. Intell. Robot. Syst., vol. 102, pp. 1–18, 2021.
- [16] Y. Wen and P. Pagilla, “Path-constrained and collision-free optimal trajectory planning for robot manipulators,” IEEE Trans. Autom. Sci. Eng., early access, Apr. 2023.
- [17] M. Abdelaal and A. Hahn, “Predictive path following and collision avoidance of autonomous vessels in narrow channels,” in Proc. 21st IFAC World Congr., 2021, pp. 10249–10254.
- [18] M. Böck and A. Kugi, “Real-time nonlinear model predictive path-following control of a laboratory tower crane,” IEEE Trans. Control Syst. Technol., vol. 22, no. 4, pp. 1461–1473, Jul. 2014.
- [19] M. Lukassek, A. Völz, T. Szabo, and K. Graichen, “Model predictive path-following control for general n-trailer systems with an arbitrary guidance point,” in Proc. Eur. Control Conf. (ECC), Delft, The Netherlands, Jul. 2021, pp. 1335–1340.
- [20] A. Alessandretti, A. P. Aguiar, and C. N. Jones, “Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control,” in Proc. Eur. Control Conf. (ECC), Zurich, Switzerland, Jul. 2013.
- [21] M. Bonfè, C. Secchi, and E. Scioni, “Online trajectory generation for mobile robots with kinodynamic constraints and embedded control systems,” in Proc.

IFAC Symp. Robot Control (SYROCO), Dubrovnik, Croatia, Sep. 2012, pp. 214–219.

- [22] da C. Vangasse, A., J. R. Freitas, E., V. Raffo, G. *et al.* Safe Navigation on Path-Following Tasks: A Study of MPC-based Collision Avoidance Schemes in Distributed Robot Systems. *J Intell Robot Syst* 110, 166 (2024). doi: 10.1007/s10846-024-02202-3.
- [23] N. N. Deniz and F. A. Cheein, “Model predictive path-following framework for generalized N-trailer vehicles in the presence of dynamic obstacles modeled as soft constraints,” *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 7018–7032, 2025, doi: 10.1109/TASE.2024.3458809.
- [24] X. Li, Z. Sun, Q. Zhu, and D. Liu, “A unified approach to local trajectory planning and control for autonomous driving along a reference path,” in Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA), Tianjin, China, Aug. 2014.
- [25] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, “A potential-field-based model predictive path planning controller for autonomous road vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, May 2017.
- [26] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, “Path planning for autonomous vehicles using model predictive control,” in Proc. IEEE Intell. Vehicles Symp. (IV), Redondo Beach, CA, USA, Jun. 2017, pp. 174–179.
- [27] Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Sacramento, CA, USA, Apr. 1991, pp. 1398–1404.
- [28] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, “A fast procedure for computing the distance between complex objects in three-dimensional space,” *IEEE J. Robot. Autom.*, vol. 4, no. 2, pp. 193–203, Apr. 1988.
- [29] E. G. Gilbert and C.-P. Foo, “Computing the distance between general convex objects in three-dimensional space,” *IEEE Trans. Robot. Autom.*, vol. 6, no. 1, pp. 53–61, Feb. 1990.
- [30] G. van den Bergen, “Proximity queries and penetration depth computation on 3D game objects,” in Proc. Game Developers Conf., 2001.
- [31] M. C. Lin and J. F. Canny, “A fast algorithm for incremental distance calculation,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Sacramento, CA, USA, Apr. 1991, pp. 1008–1014.

- [32] S. A. Ehmann and M. C. Lin, "Accelerated proximity queries between convex polyhedra by multi-level Voronoi marching," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2000, pp. 2101–2106.
- [33] J. Pan, S. Chitta, and D. Manocha, "FCL: A general-purpose library for collision and proximity queries," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), St. Paul, MN, USA, May 2012, pp. 3859–3866.
- [34] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Kobe, Japan, May 2009, pp. 489–494.
- [35] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in Robotics: Science and Systems IX, 2013, pp. 1–9.
- [36] J. Nocedal and S. J. Wright, Numerical Optimization, 2nd ed. New York, NY, USA: Springer, 2006.
- [37] M. Gifftthaler, M. Neunert, M. Stäuble, J. Buchli, M. Frigerio, and C. Semini, "Automatic differentiation of rigid body dynamics for optimal control and estimation," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Singapore, May 2017, pp. 5661–5668.
- [38] X. Zhang, A. Liniger and F. Borrelli, "Optimization-Based Collision Avoidance," in *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, May 2021, doi: 10.1109/TCST.2019.2949540.
- [39] S. Zimmermann, M. Busenhardt, S. Huber, R. Poranne, and S. Coros, "Differentiable collision avoidance using collision primitives," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2022, pp. 8086–8093.
- [40] K. Tracy, T. A. Howell, and Z. Manchester, "Differentiable collision detection for a set of convex primitives," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), London, U.K., 2023, pp. 3663–3670.
- [41] L. Montaut, Q. Le Lidec, V. Petrik, J. Sivic, and J. Carpentier, "Collision detection accelerated: An optimization perspective," in Robotics: Science and Systems XVIII, 2022.
- [42] L. Montaut, Q. L. Lidec, A. Bambade, V. Petrik, J. Sivic and J. Carpentier, "Differentiable Collision Detection: a Randomized Smoothing Approach," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, 2023, pp. 3240–3246, doi:

10.1109/ICRA48891.2023.10160251.

- [43] Fiacco, A.V., Ishizuka, Y. Sensitivity and stability analysis for nonlinear programming. *Ann Oper Res* 27, 215–235 (1990). doi: 10.1007/BF02055196.
- [44] J.-F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*. New York, NY, USA: Springer, 2000, doi: 10.1007/978-1-4612-1394-9.
- [45] A. L. Dontchev and R. T. Rockafellar, *Implicit Functions and Solution Mappings: A View from Variational Analysis*, 2nd ed. New York, NY, USA: Springer, 2014. doi: 10.1007/978-1-4939-1037-3.
- [46] Y. Wang and S. Boyd, “Fast model predictive control using online optimization,” *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.
- [47] M. Diehl, H. J. Ferreau, and N. Haverbeke, “Efficient numerical methods for nonlinear MPC and moving horizon estimation,” in *Nonlinear Model Predictive Control*, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Berlin, Germany: Springer, 2009, pp. 391–417.
- [48] D. Ternet and L. T. Biegler, “Interior point methods for reduced Hessian successive quadratic programming,” *Comput. Chem. Eng.*, vol. 23, no. 6, pp. 859–873, Jun. 1999. doi: 10.1016/S0098-1354(99)00013-7.
- [49] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.
- [50] R. A. Bartlett, A. Wachter and L. T. Biegler, “Active set vs. interior point strategies for model predictive control,” *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, Chicago, IL, USA, 2000, pp. 4229–4233 vol.6, doi: 10.1109/ACC.2000.877018.
- [51] A. Wächter and L. T. Biegler, “Line search filter methods for nonlinear programming: Motivation and global convergence,” *SIAM J. Optim.*, vol. 16, no. 1, pp. 1–31, 2005, doi: 10.1137/S1052623403426556.
- [52] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, “Auto-generated algorithms for nonlinear model predictive control on long and on short horizons,” in *Proc. IEEE Conf. Decis. Control (CDC)*, Florence, Italy, Dec. 2013, pp. 5113–5118.
- [53] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Madison, WI, USA: Nob Hill Publishing, 2017.

- [54] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *J. Optim. Theory Appl.*, vol. 99, no. 3, pp. 737–757, Dec. 1998.
- [55] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*, 2nd ed. Cham, Switzerland: Springer, 2017.
- [56] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on $SO(3)$," in *Proc. IEEE Multi-Conf. Syst. Control (MSC)*, Sydney, Australia, Sep. 2015, pp. 1270–1276.
- [57] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados: A modular open-source framework for fast embedded optimal control," *Math. Program. Comput.*, vol. 14, pp. 147–183, 2022. doi: 10.1007/s12532-021-00208-8.
- [58] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020. doi: 10.1016/j.ifacol.2020.12.073.
- [59] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia, PA, USA: SIAM, 1997, doi: 10.1137/1.9781611971453.
- [60] E. A. Yildırım and S. J. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM J. Optim.*, vol. 12, no. 3, pp. 782–810, 2002. doi: 10.1137/S1052623400369235.
- [61] J. Gondzio and A. Grothey, "Reoptimization with the primal-dual interior point method," *SIAM J. Optim.*, vol. 13, no. 3, pp. 842–864, 2003.
- [62] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2012, doi: 10.1017/CBO9781139020411.
- [63] E. V. Haynsworth, "Determination of the inertia of a partitioned Hermitian matrix," *Linear Algebra Appl.*, vol. 1, no. 1, pp. 73–81, Jan. 1968, doi: 10.1016/0024-3795(68)90050-5.
- [64] F. Zhang, *The Schur Complement and Its Applications*. New York, NY, USA: Springer, 2005.
- [65] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*, Version 20121115. Lyngby, Denmark: Technical University of Denmark, Nov. 2012. [Online]. Available: <https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>.

- [66] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*, 3rd ed. Hoboken, NJ, USA: Wiley, 2019.
- [67] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Aug. 2009, doi: 10.1137/07070111X.
- [68] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000, doi: 10.1137/S0895479896305696.
- [69] W. Hackbusch, "Tensor Spaces and Numerical Tensor Calculus." Springer Series in Computational Mathematics (2012).
- [70] G. A. Holzapfel, *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*. Hoboken, NJ, USA: Wiley, 2000.
- [71] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed. Philadelphia, PA, USA: SIAM, 2008, doi: 10.1137/1.9780898717761.
- [72] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 1994.
- [73] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems: Modeling, Analysis, and Design for Simple Mechanical Control Systems*. New York, NY, USA: Springer, 2005.
- [74] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2005.
- [75] R. Featherstone, *Rigid Body Dynamics Algorithms*. New York, NY, USA: Springer, 2008.
- [76] E. Y. L. Gu, *A Journey from Robot to Digital Human: Mathematical Principles and Applications with MATLAB Programming*. Berlin, Germany: Springer, 2013.
- [77] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.
- [78] B. A. Dubrovin, A. T. Fomenko, and S. P. Novikov, *Modern Geometry—Methods and Applications: Part I: The Geometry of Surfaces, Transformation Groups, and Fields*, 2nd ed. New York, NY, USA: Springer, 1992.

- [79] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9–10, pp. 1164–1193, Aug. 2013. doi: 10.1177/0278364913488805.
- [80] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014. doi: 10.1177/0278364914528132.
- [81] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," *Workshop on the Algorithmic Foundations of Robotics* (2014).
- [82] S. Liu *et al.*, "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments," in *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, July 2017, doi: 10.1109/LRA.2017.2663526.
- [83] S. Bernstein, "Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités," *Commun. Soc. Math. Kharkow*, 2nd ser., vol. 13, no. 1, pp. 1–2, 1912.
- [84] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. San Diego, CA, USA: Morgan Kaufmann, 2001.
- [85] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. Berlin, Germany: Springer, 1997, doi: 10.1007/978-3-642-59223-2.
- [86] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Comput. Graph. Image Process.*, vol. 1, no. 3, pp. 244–256, 1972. doi: 10.1016/S0146-664X(72)80017-0.
- [87] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica*, vol. 10, no. 2, pp. 112–122, 1973, doi: 10.3138/FM57-6770-U75U-7727.
- [88] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: Johns Hopkins University Press, 2013.
- [89] R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," *Math. Program.*, vol. 91, pp. 239–269, 2002. doi: 10.1007/s101070100244.

- [90] Ulbrich, M., Ulbrich, S. & Vicente, L. A globally convergent primal-dual interior-point filter method for nonlinear programming. *Math. Program., Ser. A* 100, 379–410 (2004). doi: 10.1007/s10107-003-0477-4.
- [91] Mayne, D.Q., Maratos, N. A first order, exact penalty function algorithm for equality constrained optimization problems. *Mathematical Programming* **16**, 303–324 (1979). doi: 10.1007/BF01582118.
- [92] D. Q. Mayne and E. Polak, “A superlinearly convergent algorithm for constrained optimization problems,” in *Algorithms for Constrained Minimization of Smooth Nonlinear Functions*, A. G. Buckley and J. L. Goffin, Eds. Berlin, Germany: Springer, 1982, pp. 45–61, doi: 10.1007/BFb0120947.
- [93] R. H. Byrd, M. E. Hribar, and J. Nocedal, “An interior point algorithm for large-scale nonlinear programming,” *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, 1999.
- [94] A. Forsgren, P. E. Gill, and M. H. Wright, “Interior methods for nonlinear optimization,” *SIAM Rev.*, vol. 44, no. 4, pp. 525–597, 2002. doi: 10.1137/S0036144502414942.
- [95] F. A. Potra and S. J. Wright, “Interior-point methods,” *J. Comput. Appl. Math.*, vol. 124, no. 1–2, pp. 281–302, 2000. doi: 10.1016/S0377-0427(00)00433-7.
- [96] A. Shahzad, E. C. Kerrigan, and G. A. Constantinides, “A warm-start interior-point method for predictive control,” in *Proc. UKACC Int. Conf. Control*, Coventry, U.K., 2010, pp. 1–6. doi: 10.1049/ic.2010.0409.
- [97] A. Shahzad and P. J. Goulart, “A new hot-start interior-point method for model predictive control,” *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 2470–2475, 2011.
- [98] A. Zanelli, R. Quirynen, J. Jerez, and M. Diehl, “A Homotopy-based nonlinear interior-point method for NMPC,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13188–13193, 2017. doi: 10.1016/j.ifacol.2017.08.2175.
- [99] E. A. Yildırım and S. J. Wright, “Warm-start strategies in interior-point methods for linear programming,” *SIAM J. Optim.*, vol. 12, no. 3, pp. 782–810, 2002, doi: 10.1137/S1052623400369235.
- [100] J. Gondzio and A. Grothey, “Reoptimization with the primal-dual interior point method,” *SIAM J. Optim.*, vol. 13, no. 3, pp. 842–864, 2003, doi: 10.1137/S1052623401393141.
- [101] J. Gondzio and A. Grothey, “A new unblocking technique to warmstart interior point methods based on sensitivity analysis,” *SIAM J. Optim.*, vol. 19, no. 3, pp.

1184–1210, 2008. doi: 10.1137/060678129.

- [102] E. John and E. A. Yildirim, “Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension,” *Comput. Optim. Appl.*, vol. 41, pp. 151–183, 2008. doi: 10.1007/s10589-007-9096-y.
- [103] C. Chen and M. Tomizuka, “Dynamic modeling of tractor-semitrailer vehicles in automated highway systems,” California PATH Program, Inst. Transp. Studies, Univ. California, Berkeley, Berkeley, CA, USA, Working Paper UCB-ITS-PWP-95-8, Jul. 1995.
- [104] M. De Bernardis, G. Rini, F. Bottiglione, A. E. Hartavi, and A. Sorniotti, “On nonlinear model predictive direct yaw moment control for trailer sway mitigation,” *Veh. Syst. Dyn.*, vol. 61, no. 2, pp. 445–471, 2023, doi: 10.1080/00423114.2022.2054352.
- [105] R. Rajamani, *Vehicle Dynamics and Control*, 3rd ed. New York, NY, USA: Springer, 2012.
- [106] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986. doi: 10.1177/027836498600500106.
- [107] S. Quinlan and O. Khatib, “Elastic bands: Connecting path planning and control,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, 1993, pp. 802–807.
- [108] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, 1997. doi: 10.1109/100.580977.
- [109] M. Diehl, H. G. Bock, and J. P. Schlöder, “A real-time iteration scheme for nonlinear optimization in optimal feedback control,” *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005. doi: 10.1137/S0363012902400713.
- [110] Y. Zhi, N. Das, and M. C. Yip, “DiffCo: Auto-differentiable proxy collision detection with multi-class labels for safety-aware trajectory optimization,” *arXiv preprint arXiv:2102.07413*, 2021.
- [111] J. Pan and D. Manocha, “Efficient configuration space construction and optimization for motion planning,” *Engineering*, vol. 1, no. 1, pp. 46–57, Mar. 2015, doi: 10.15302/J-ENG-2015009.
- [112] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, “Adaptively sampled

- distance fields: A general representation of shape for computer graphics,” in *Proc. ACM SIGGRAPH*, 2000, pp. 249–254, doi: 10.1145/344779.344899.
- [113] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 165–174, doi: 10.1109/CVPR.2019.00025.
 - [114] S. Mehrotra, “On the implementation of a primal-dual interior point method,” *SIAM J. Optim.*, vol. 2, no. 4, pp. 575–601, 1992. doi: 10.1137/0802028.
 - [115] J. Gondzio, “Multiple centrality corrections in a primal-dual method for linear programming,” *Comput. Optim. Appl.*, vol. 6, no. 2, pp. 137–156, 1996. doi: 10.1007/BF00249643.
 - [116] M.-K. Bouzidi, Y. Yao, D. Goehring, and J. Reichardt, “Learning-aided warmstart of model predictive control in uncertain fast-changing traffic,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Yokohama, Japan, May 2024, pp. 14265–14271, doi: 10.1109/ICRA57147.2024.10610472.
 - [117] R. Sambharya, G. Hall, B. Amos, and B. Stellato, “Learning to warm-start fixed-point optimization algorithms,” *J. Mach. Learn. Res.*, vol. 25, no. 166, pp. 1–37, 2024.
 - [118] M. Mehndiratta, E. Camci, and E. Kayacan, “Automated tuning of nonlinear model predictive controller by reinforcement learning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 3016–3021, doi: 10.1109/IROS.2018.8594350.
 - [119] S. Gros and M. Zanon, “Data-driven economic NMPC using reinforcement learning,” *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 636–648, Feb. 2020, doi: 10.1109/TAC.2019.2913768.
 - [120] F. Moreno-Mora, L. Beckenbach, and S. Streif, “Predictive control with learning-based terminal costs using approximate value iteration,” *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3874–3879, 2023, doi: 10.1016/j.ifacol.2023.10.1320.