Time-Optimal Nonlinear Model Predictive Control

Direct Transcription Methods with Variable Discretization and Structural Sparsity Exploitation

DISSERTATION

submitted in partial fulfillment
of the requirements for the degree

Doktor-Ingenieur
(Doctor of Engineering)

in the

Faculty of Electrical Engineering and Information Technology
at Technische Universität Dortmund

by

Christoph Rösmann, M.Sc.

Münster, Germany

Date of submission: April 29, 2019
Place of submission: Dortmund, Germany

First examiner: Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram
Second examiner: Univ.-Prof. Dr.-Ing. Martin Mönnigmann

Date of approval: October 14, 2019

Preface

This thesis was written during my work at the Institute of Control Theory and Systems Engineering of the Faculty of Electrical Engineering and Information Technology at the Technical University of Dortmund.
I would like to thank Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram that he made my scientific work possible and generously supported it. I am very grateful for his valuable feedback, confidence and inspiring suggestions. He has always given me the freedom to develop my own scientific knowledge and present it to a broad audience at numerous local and international conferences. I would also like to thank Univ.-Prof. Dr.-Ing. Martin Mönnigmann, who agreed to review my thesis as second examiner. I greatly appreciate his valuable feedback and expertise.
Additionally, I would like to thank all staff members and former staff members of the Institute of Control Theory and Systems Engineering for the pleasant working atmosphere, the cooperativeness and camaraderie. In particular, I am grateful for the assistance given by apl. Prof. Dr. rer. nat. Frank Hoffmann. His excellent experience in many different engineering and scientific fields, his insightful comments and the intensive discussions have helped me a lot to advance my ideas and research projects. Dr.-Ing. Daniel Schauten deserves my special thanks for the many valuable conversations, advices and support during my time at the department, especially regarding best practices in teaching and research. I am very thankful to Artemi Makarow for all the time we spent discussing and questioning many control theoretical topics, for the conference trips we have experienced together and for the extraordinary support and encouragement especially in the final stages of completing my thesis. Special thanks also go to Maximilian Krämer and Christian Wissing for the intensive exchange in many topics, especially in robotics and software development in C++. For the always kind assistance with administrative and technical questions I would like to thank Nicole Czerwinski, Gabriele Rebbe, Sascha Kersting, Jürgen Limhoff and Rainer Müller-Burtscheid.
I would also like to express my gratitude for the financial support provided by the German Research Foundation (DFG).
Words of gratitude are dedicated to my friends who helped me to find distraction and relaxation. Finally, I am very grateful to my family for their continuous unconditional support and encouragement throughout my life.

Dortmund, October 2019 Christoph Rösmann

Abstract

This thesis deals with the development and analysis of novel time-optimal model predictive control concepts for nonlinear systems. Common realizations of model predictive controllers apply direct transcription methods to first discretize and then optimize the subordinate optimal control problems. The key idea of the proposed concepts is to introduce discretization grids in which the underlying discretization is explicitly treated as temporally variable during optimization. A single optimization parameter for all grid intervals leads to the global uniform grid, while the definition of an individual parameter for each interval results in the local uniform and quasi-uniform grid representations. The proposed grids are well-suited for established direct transcription methods such as multiple shooting and collocation. In addition, a proposed non-uniform grid with extended multiple shooting is highly beneficial for bang-singular-bang control systems with simple constraint sets. The minimization of the local time information of a grid leads to an overall time-optimal transition. Integration with state feedback does not immediately guarantee asymptotic stability and recursive feasibility. To this end, the thesis provides a grid adaptation scheme capable of ensuring practical stability and, under more restricted conditions, also nominal asymptotic stability while maintaining feasibility. The practical stability results facilitate the systematic dual-mode control design that restores asymptotic stability and establishes smooth stabilization.

The secondary objective of this thesis is the computationally efficient realization of time-optimal model predictive control by exploiting the inherent sparse structures in the optimal control problems. In particular, the efficient computation of first- and second-order derivatives required for iterative optimization is facilitated by a so-called hypergraph. The hypergraph captures the structure of the transcribed optimal control problems and enables an almost linear relation between computation time and grid size. In addition, the hypergraph shows negligible computation times for each reconfiguration that is essential for grid adaptation.

Numerous examples in simulation and with a real experimental system demonstrate the capabilities and potentials of the proposed concepts. Extensive benchmarks in C++ compare the proposed methods with each other and the current state of the art. The methods based on variable discretization outperform the current time-optimal model predictive control methods in the literature, especially with regard to computation time.

Contents

Nomenclature

General Symbols

ℓ_1	Absolute value norm
ℓ_2	Euclidean norm
$\bar{T}_\mathrm{p}$	Computation time required for solving the optimization problem
$\bar{T}_\mathrm{s}$	Computation time required for preparing the optimization problem
Δt_cpu	Computation time for determining the control law
τ	Auxiliary time parameter
θ	Scalar weight for the ℓ_1-norm cost function
θ_0	Lower bound on θ to ensure time-optimality
ξ	Auxiliary parameter in the Cauchy-Schwarz inequality

Symbols and Functions for General Dynamic Systems and Control Tasks

A	System matrix of a linear system
B	Input matrix of a linear system
$f(\cdot)$	System dynamics vector field
$f_\mathrm{d}(\cdot)$	Sampled-data system dynamics vector field
K	Linear feedback gain matrix
$\bar{L}$	Lipschitz constant
$\bar{r}$	Scalar parameter in the definition of Lipschitz continuity
p	State dimension
$\varphi(\cdot)$	Solution to the initial value problem of the system dynamics
q	Control dimension
$\bar{t}$	Transformed time parameter (time transformation approach)
t_f	Final time
t_s	Initial time
t	Time
$u_1(t)$	First component of control trajectory $u(t)$
$u_2(t)$	Second component of control trajectory $u(t)$
$u(t)$	Control trajectory
u_f	Final control respectively reference control
$u_\mathrm{dist}(t_\mu)$	Disturbance profile with closed-loop time t_μ
$u_\mathrm{lin}(t)$	Control trajectory of the linear system
u_max	Upper bound on the control
u_min	Lower bound on the control
$x_1(t)$	First component of state trajectory $x(t)$
$x_2(t)$	Second component of state trajectory $x(t)$
$x(t)$	State trajectory
x_f	Final state respectively steady state
$x_\mathrm{f,1}$	First component of the final state

$x_{\mathrm{f},2}$	Second component of the final state
$x_{\mathrm{ref}}(t)$	Reference trajectory
x_{s}	Initial state
$x_{\mathrm{lin}}(t)$	State trajectory of the linear system
$x_{\max}$	Upper bound on the state
$x_{\min}$	Lower bound on the state
$y(t)$	System output
$y_1(t)$	First component of $y(t)$
$y_2(t)$	Second component of $y(t)$

Symbols and Functions for Predictions, Open-Loop Control and Direct Transcription

β_1	Coefficient for the linear term in the quadratic control polynomial
β_2	Coefficient for the quadratic term in the quadratic control polynomial
Δt	Global time interval $t_{k+1} - t_k, \forall k$
Δt_k	Local time interval $t_{k+1} - t_k$
Δt_ϵ	Threshold for inserting or removing intervals (grid adaptation)
$\Delta t_{\max}$	Upper bound on the time interval
$\Delta t_{\min}$	Lower bound on the time interval
Δt_{ref}	Reference time interval for grid adaptation
Δt^*	Optimal global time interval $t_{k+1} - t_k, \forall k$
Δt_k^*	Optimal local time interval $t_{k+1} - t_k$
$t_{\min}$	Lower bound on the final time t_{f} (hybrid cost approach)
ι	Running index for enhanced integration (non-uniform grid)
Δt_{int}	Step size for the integration steps $\Delta t_k / N_{\mathrm{int}}$ (non-uniform grid)
$\Delta t_{\mathrm{rem},k}$	Remaining step width for interval Δt_k (non-uniform grid)
Δu_k	Control error at time instance t_k with respect to u_{f}
Δx_k	State error at time instance t_k with respect to x_{f}
$e_{\hat{x}}(t_{\mathrm{f}})$	Integral error at time t_{f}
$e_{\hat{x}}(t)$	Integral error with respect to $\hat{x}_u(t)$ (ℓ_2-norm)
$\tilde{g}(\cdot)$	Integrand of the additional constraint equation (non-uniform grid)
γ_1	Coefficient for the linear term in the cubic state polynomial
γ_2	Coefficient for the quadratic term in the cubic state polynomial
γ_3	Coefficient for the cubic term in the cubic state polynomial
$h_{\mathrm{hs}}(\cdot)$	State interpolant at $t_{k+0.5}$ in Hermite-Simpson collocation
$h_{\mathrm{fd}}(\cdot)$	Equality constraint for collocation via finite differences
$h_{\mathrm{fe}}(\cdot)$	Equality constraint for multiple shooting with forward Euler
I_{adapt}	Number of iterations for grid adaptation
i	General running index (multiple shooting grid, grid adaptation, fundamentals of nonlinear optimization)
$\bar{k}$	Running index for accumulating time intervals of the local grid
κ	Running index for Levenberg-Marquardt and SQP iterations
k	Index for time instances (prediction)
$\ell(\cdot)$	Running cost
$\tilde{m}$	Number of controls in each shooting interval, $\tilde{m}_i = \tilde{m}\,\forall i$

$\tilde{m}_i$	Number of controls in the i-th shooting interval
M	Number of shooting intervals
N_{init}	Initial grid size N for grid adaptation
$N_{\max}$	Upper bound on the grid size N
$N_{\min}$	Lower bound on the grid size N
N	Discretization grid size or horizon length
N^*	Minimum grid size respectively dead-beat horizon length
N_{crit}	Smallest grid size for which the non-uniform grid is feasible
$\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_6$	Auxiliary parameters in the explicit 5^{th}-order Runge-Kutta method
N_{b}	Number of control interventions (non-uniform grid)
$\tilde{N}_{\text{b}}$	Surplus in the number of control interventions (non-uniform grid)
u_ϵ	Similarity threshold for two consecutive controls (non-uniform grid)
N_{int}	Number of integration steps for interval $\Delta t_k, \forall k$ (non-uniform grid)
$N_{\text{int},k}$	Number of integration steps for interval Δt_k (non-uniform grid)
$\phi_{\text{fd}}(\cdot)$	Finite difference kernel function
$\phi_{\text{hs}}(\cdot)$	Simpson quadrature of $f(\cdot)$ in Hermite-Simpson collocation
ϖ	Running index for interstitial grid points k in each shooting interval
$\tilde{Q}$	Weighting matrix for the state error (Riemann sum approximation)
Q	State error weighting matrix in quadratic form cost
$\tilde{R}$	Weighting matrix for the control effort (Riemann sum approximation)
R	Control effort weighting matrix in quadratic form cost
ρ	Weight for the regularization term (non-uniform grid)
$\bar{s}_k$	Slack variable with index k in the ℓ_1-norm approach
s_i	Shooting node at time instance $\tilde{t}_i$, $x_u(\tilde{t}_i) := s_i$
$\tilde{t}_i$	Time instance with index i (multiple shooting grid)
t_0	Initial time (prediction)
t_k	Time instance with index k (prediction)
$\tilde{u}_i(t)$	Control trajectory on shooting interval $[\tilde{t}_i, \tilde{t}_{i+1}]$, $u(t + \tilde{t}_i) := \tilde{u}_i(t)$
$u^*(t; t_\mu)$	Optimal control trajectory at closed-loop time t_μ
$u^*(t)$	Optimal control trajectory
u_k^*	Optimal control at time instance t_k (prediction)
u_k	Control at time instance t_k, $u(t_k) := u_k$ (prediction)
$V_{\text{dyn},N}$	Cost for the non-uniform grid control problem with size N
$V_{\text{dyn},N}^*$	Optimal cost for the non-uniform grid control problem with size N
$V_{\text{f}}(\cdot)$	Terminal cost
$V_k(\cdot)$	Integral of the running cost on interval $[t_k, t_{k+1}]$
$x_{u,1}(t)$	First component of state trajectory $x_u(t)$ (prediction)
$x_{u,2}(t)$	Second component of state trajectory $x_u(t)$ (prediction)
$\hat{x}_{u,1}(t)$	First component of state trajectory $\hat{x}_u(t)$
$\hat{x}_{u,2}(t)$	Second component of state trajectory $\hat{x}_u(t)$
$\hat{x}_u(t)$	Open-loop state trajectory (for error calculation)
$x_u(t)$	State trajectory (prediction)
x_k	State at time instance t_k, $x_u(t_k) := x_k$ (prediction)
1_p	Vector of ones with dimension $p \times 1$
I_n	Identity matrix with dimension $n \times n$

Symbols and Functions for Closed-Loop Control

$\Delta t_{\mu,n}$	Time interval between time instances $t_{\mu,n}$ and $t_{\mu,n+1}$
$g_{\mathrm{f}}(\cdot)$	Closed-loop system dynamics vector field
$\mu(\cdot)$	Control law
$\mu_1(\cdot)$	First component of the control-law $\mu(\cdot)$
$\mu_2(\cdot)$	Second component of the control-law $\mu(\cdot)$
$\mu_{\mathrm{dual}}(\cdot)$	Dual-mode control-law
$\mu_{\mathrm{lin}}(\cdot)$	Control law of the local controller (dual-mode)
n	Index for time instances (closed-loop)
$\varphi_\mu(\cdot)$	Solution to the initial value problem of $g_{\mathrm{f}}(\cdot)$
t_μ	Time variable (closed-loop)
$t_{\mu,n}$	Time instance with index n (closed-loop)
$x_{\mu,1}(t_\mu)$	First component of state trajectory $x_\mu(t)$
$x_{\mu,2}(t_\mu)$	Second component of state trajectory $x_\mu(t)$
$x_\mu(t_\mu)$	State trajectory (closed-loop)

Symbols and Functions for the Stability Analysis

$\alpha_1(\cdot)$	Comparison function, $\alpha_1 \in \mathscr{K}_\infty$
$\alpha_2(\cdot)$	Comparison function, $\alpha_2 \in \mathscr{K}_\infty$
$\alpha_V(\cdot)$	Comparison function, $\alpha_V \in \mathscr{K}$
$\beta(\cdot)$	Comparison function, $\beta \in \mathscr{KL}$
η	Distance from x_{f} (stability analysis)

Symbols and Functions for Nonlinear Programs and the Hypergraph

$\chi(\cdot)$	Weighted quadratic penalty function for inequality constraints
$\tilde{d}$	Constant term in least-squares derivation
$D\big(\Phi(\cdot), \Delta z\big)$	Directional derivative of $\Phi(\cdot)$ with respect to Δz.
Δz	Parameter update vector
Δz^*	Optimal parameter update vector
e_r^g	Edge in the hypergraph that is associated with $g_r(\cdot)$
e_s^h	Edge in the hypergraph that is associated with $h_s(\cdot)$
e_l^J	Edge in the hypergraph that is associated with $J_l(\cdot)$
η_{LS}	Growth rate for the least-squares approximation
λ_{LM}	Levenberg-Marquardt damping factor
η_{sqp}	Line-search parameter in sequential quadratic programming
$F(z)$	Least-squares cost vector with respect to parameter z
$\tilde{g}(\cdot)$	Quadratic penalty function for inequality constraints
$g(\cdot)$	General inequality constraint function
$g_r(\cdot)$	Component with index r in $g(\cdot)$
H	Hessian of the least-squares problem and Hessian in the SQP
$h(\cdot)$	General equality constraint function
$h_s(\cdot)$	Component with index s in $h(\cdot)$
I_{LM}	Number of Levenberg-Marquardt solver iterations
I_{SQP}	Number of SQP solver iterations
$J(\cdot)$	General cost function
$J_l(\cdot)$	Cost summand with index l in $J(\cdot)$

δ_{D}	Pertubation of finite difference calculations (first-order)
δ_{H}	Pertubation of finite difference calculations (second-order)
$\mathcal{L}(\cdot)$	Lagrangian of a nonlinear program
L	Number of summands in $J(\cdot)$
l	Running index for summands in $J(\cdot)$
λ	Lagrange multiplier for the equality constraint
λ^*	Optimal lagrange multiplier for the equality constraint
λ_i^*	Optimal lagrange multiplier for the i-th equality constraint
λ_i	Lagrange multiplier for the i-th equality constraint
μ	Lagrange multiplier for the inequality constraint
μ^*	Optimal lagrange multiplier for the inequality constraint
μ_i^*	Optimal lagrange multiplier for the i-th inequality constraint
μ_{m}	Penalty parameter in merit function $\Phi(\cdot)$
μ_i	Lagrange multiplier for the i-th inequality constraint
$\Phi(\cdot)$	Merit function for line-search in sequential quadratic programming
$\tilde{p}$	Linear term in least-squares derivation
$\psi(\cdot)$	Weighted quadratic penalty function for equality constraints
R	Dimension of inequality constraint function $g(\cdot)$
r	Running index for components of $g(\cdot)$
S	Dimension of equality constraint function $g(\cdot)$
s	Running index for components of $h(\cdot)$
σ_1	Penalty weighting parameter for equality constraints
$\tilde{\sigma}_1$	Square root of σ_1
σ_2	Penalty weighting parameter for inequality constraints
$\tilde{\sigma}_2$	Square root of σ_2
τ_{sqp}	Line-search parameter in sequential quadratic programming
$\tilde{\mathrm{Y}}$	Number of unfixed vertices in the hypergraph
Y	Number of nominal optimization parameters
v_v	Vertex in the hypergraph that correspond to parameter z_v
w	Vector in constraint qualification conditions and second-order optimality conditions
$\tilde{z}_{\tilde{v}}$	Nominal optimization parameter with index $\tilde{v}$
z_v	Nominal optimization parameter with index v
n_z	Dimension of z
z	Nominal (single) optimization parameter
z^*	Optimal (single) optimization parameter
$z_{\max}$	Upper bound on nominal optimization parameter
$z_{\min}$	Lower bound on nominal optimization parameter
$z_{\mathrm{ref},\omega}$	Fixed vector asigned to z_ω for some $0 \leq \omega \leq \mathrm{Y}$
z_ω	Parameters with trivial assignments (see $z_{\mathrm{ref},\omega}$)

Symbols and Functions for Benchmark Systems

a_{vdp}	Damping coefficient of the Van der Pol oscillator
c_{drag}	Drag coefficient of the rocket system
c_{roc}	Mass rate of change coefficient of the rocket system

$\tilde{c}_1$	Friction coefficient (ECP Model 220)
$\tilde{c}_2$	Friction coefficient (ECP Model 220)
c_1	Linear damping coefficient (ECP Model 220)
c_2	Coefficient for nonlinear damping (ECP Model 220)
c_3	Slope in the nonlinear damping term (ECP Model 220)
F_{inertial}	Inertial force of the rocket system
F_{drag}	Drag force of the rocket system
F_{thrust}	Thrust force of the rocket system
J_{ecp}	Moment of inertia (ECP Model 220)
$\tilde{k}_1$	Motor specific constant (ECP Model 220)
$\tilde{k}_2$	Motor specific constant (ECP Model 220)
k_1	Gain of the first motor input (ECP Model 220)
k_2	Gain of the second motor input (ECP Model 220)
$m_{\text{r}}(t)$	Mass of the rocket system at time t
$m_{\text{r,f}}$	Final mass of the rocket system
$s_{\text{r}}(t)$	Position of the rocket system at time t
$s_{\text{r,f}}$	Final position of the rocket system
$v_{\text{r}}(t)$	Velocity of the rocket system at time t
$v_{\text{r,f}}$	Final velocity of the rocket system

Sets

$\mathscr{B}_\eta(x_{\text{f}})$	Ball around x_{f} with distance η
$\mathcal{C}(\cdot)$	Critical cone
$\emptyset$	Empty set
$\mathcal{E}$	Set of edges in the hypergraph
$\mathcal{E}_{\text{e}}$	Set of indices of equality constraints
$\mathcal{F}(z)$	Linearized feasible directions at z
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Hypergraph
$\mathcal{A}_{\mathcal{I}}(z)$	Set of indices of active inequality constraints with respect to z
$\mathcal{I}$	Set of indices of inequality constraints
I	Open time interval, $I \subset \mathbb{R}$
$\mathcal{K}$	Set of indices that relate to redundant controls (non-uniform grid)
$\mathcal{N}$	Neighborhood of an optimal parameter z^*
$\mathbb{N}$	Positive natural numbers excluding zero
$\mathbb{N}_0$	Natural numbers including zero
Ω	Feasible set for the nominal nonlinear program
$\mathcal{P}_{\text{global}}$	Set of optimization parameters for the global grid
$\mathcal{P}_{\text{local}}$	Set of optimization parameters for the local grid
$\mathcal{P}_{\text{local},N}$	Set of optimization parameters for the local grid with size N
P	Generic set for practical asymptotic stability
$P_c(\cdot)$	Controllability region with respect to $\mathscr{U}^N$
$P_{\Delta t_{\min}}(x_{\text{f}})$	Controllability region to x_{f} up to time $\Delta t_{\min}$
$\mathbb{R}^+$	Positive real numbers excluding zero
$\mathbb{R}_0^+$	Positive real numbers including zero
$\mathcal{S}_i$	Set of grid indices k in the shooting interval $[\tilde{t}_i, \tilde{t}_{i+1}]$

S	Lyapunov function state space, $S \subset \mathcal{X}$
$T_\Omega(z)$	Tangent cone at z
$\mathbb{U}$	Restricted control space, $\mathbb{U} \subset \mathcal{U}$
$\mathcal{U}$	Control space, $\mathcal{X} := \mathbb{R}^q$
$\mathcal{V}$	Set of vertices in the hypergraph
$\mathbb{X}_{\text{feas}}$	Feasible set in the TOMPC approach
$\mathbb{X}$	Restricted state space, $\mathbb{X} \subseteq \mathcal{X}$
$\mathbb{X}_{\text{lin}}$	Forward invariant region of the local controller (dual-mode)
$\mathbb{X}_\text{f}$	Terminal set, $\mathbb{X}_\text{f} \subseteq \mathbb{X}$
$\mathcal{X}$	State space, $\mathcal{X} := \mathbb{R}^p$
Y	Generic forward invariant set
$\tilde{\mathcal{Z}}$	Set of nominal optimization parameters whose vertex is not fixed
$\mathcal{Z}$	Set of nominal optimization parameters z_v
$\mathcal{Z}_r^g$	Set of optimization parameters that directly affect g_r
$\mathcal{Z}_s^h$	Set of optimization parameters that directly affect h_s
$\mathcal{Z}_l^J$	Set of optimization parameters that directly affect J_l

Function Spaces

$\mathcal{K}$	Comparison functions (stability analysis)
$\mathcal{K}_\infty$	Comparison functions (stability analysis)
$\mathcal{KL}$	Comparison functions for (stability analysis)
$\mathcal{L}$	Comparison functions (stability analysis)
$L^\infty([t_0, t_\text{f}], \mathcal{U})$	Lebesgue integrable control functions on interval $[t_0, t_\text{f}]$
$L^\infty(\mathbb{R}, \mathcal{U})$	Lebesgue integrable control functions on $\mathbb{R}$
$\mathcal{U}_k$	Control space on grid interval $[t_k, t_{k+1}]$
$\mathscr{U}^N(t_\text{f})$	Control space of N piecewise polynomials up to time t_f

Special Operators

$\lceil \cdot \rceil$	Ceiling operator
$\lvert Z \rvert$	Cardinality of set Z (only for sets)
$\lvert z \rvert$	Absolute value of scalar z
$\lfloor \cdot \rfloor$	Flooring operator
$\mathrm{D}_z f(z_0, \cdot)$	Jacobian of vector field $f(z, \cdot)$ with respect to z and evaluated at $(z_0, \cdot)$
$\mathrm{D} f(z_0)$	Jacobian of vector field $f(z)$ with respect to z and evaluated at z_0
$\lVert \cdot \rVert$	Arbitrary norm
$\lVert \cdot \rVert_1$	ℓ_1-norm (absolute value norm)
$\lVert \cdot \rVert_2$	ℓ_2-norm (Euclidean norm)
$\lVert \cdot \rVert_Q$	Weighted ℓ_2-norm, for example $\lVert z \rVert_Q = \sqrt{z^\intercal Q z}$
$\max(z_0, z_1)$	Componentwise maximum of vectors z_0 and z_1
$\min(z_0, z_1)$	Componentwise minimum of vectors z_0 and z_1
$\nabla_z J(z_0, \cdot)$	Gradient of scalar function $J(z, \cdot)$ with respect to z and evaluated at $(z_0, \cdot)$
$\nabla J(z)$	Gradient of scalar function $J(z)$ with respect to z and evaluated at z_0
$\nabla^2_{zz} J(z_0, \cdot)$	Hessian of scalar function $J(z, \cdot)$ with respect to z and evaluated at $(z_0, \cdot)$
$\nabla^2 J(z_0)$	Hessian of scalar function $J(z)$ with respect to z and evaluated at z_0

Abbreviations and Acronyms

AD	**A**utomatic **D**ifferentiation
BFGS	**B**royden-**F**letcher-**G**oldfarb-**S**hanno algorithm
IPOPT	**I**nterior **P**oint **Opt**imizer
KKT	**K**arush-**K**uhn-**T**ucker
LICQ	**L**inear **I**ndependence **C**onstraint **Q**ualification
LQR	**L**inear **Q**uadratic **R**egulator
MFCQ	**M**angasarian-**F**romovitz **C**onstraint **Q**ualification
MPC	**M**odel **P**redictive **C**ontrol
NRMSE	**N**ormalized **R**oot **M**ean **S**quare **E**rror
nz	**n**on-**z**eros
OSQP	**O**perator **S**plitting **Q**uadratic **P**rogram
PID	**P**roportional-**I**ntegral-**D**ifferential
RTI	**R**eal-**T**ime **I**teration Scheme
SQP	**S**equential **Q**uadratic **P**rogramming
w.r.t.	**w**ith **r**espect **t**o

1 Introduction

1.1. Motivation

In a number of industries, minimizing time is essential to increasing the productivity of automation solutions. To give a few examples, gantry cranes lift and transport a large number of containers in ports to meet the increasing demand for import and export goods. In addition to sophisticated logistics, faster crane control is crucial to increase productivity. In the field of warehouse robotics, mobile robots are also expected to navigate as fast as possible while avoiding obstacles. The productivity in the area of automated assembly at automobile manufacturers correlates strongly with the execution speed of their robotic manipulators. Racing is also dedicated to minimizing lap times as a central objective. In most of today's industrial applications, however, linear control concepts such as PID or robust H_∞-control [Zho+96] as well as linear quadratic regulators (LQR) [KS72] are used due to their simple realization and implementation. The minimization of time is only achieved by an aggressive tuning of the control parameters. These conventional controllers cannot explicitly incorporate constraints on control respectively state variables such as limited crane swing, obstacle avoidance or bounded robot respectively car velocities without conservatism and further restrictions [Sch+97]. Practical control systems are therefore not operated at

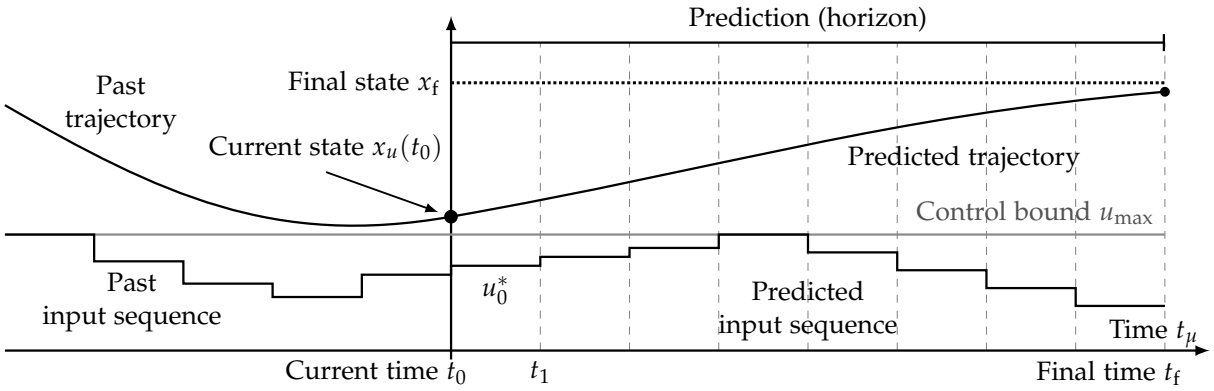


Figure 1.1.: Receding horizon principle in MPC.

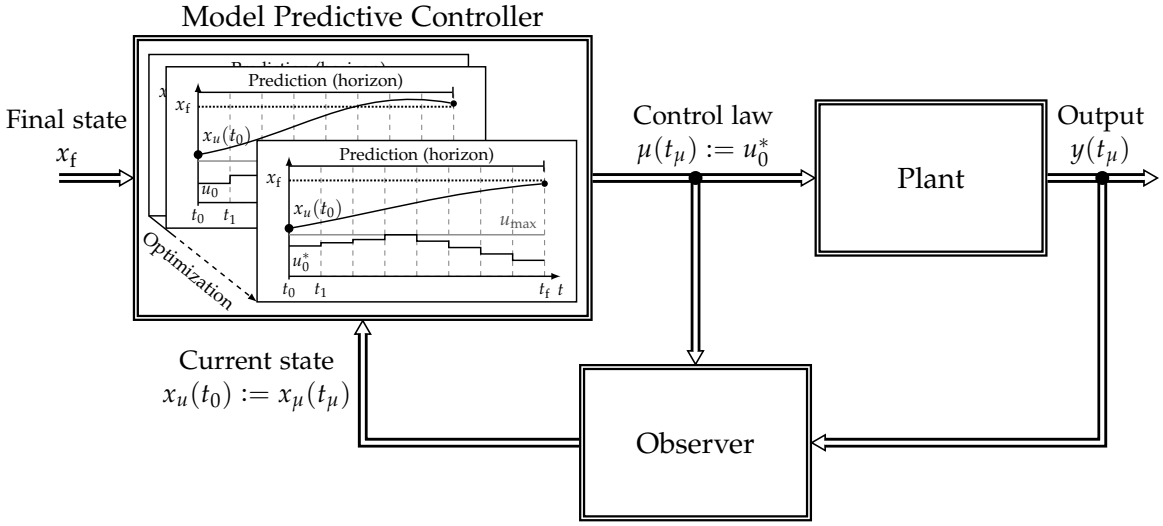


Figure 1.2.: Typical point-to-point MPC architecture. Subscript μ indicates the context to the closed-loop evolution wheres subscript u refers to the prediction.

their potential performance limits in order to achieve actual time-optimality.
In recent decades, model predictive control (MPC) became an established and comprehensive framework for controlling nonlinear systems under appropriate performance criteria and explicit adherence to constraints. In each sampling interval, the model predictive controller employs a (nonlinear) model of the dynamic system to predict and optimize the future evolution of the system within a specified temporal horizon as illustrated in Figure 1.1. The first portion of the optimized control sequence is commanded to the plant before the optimization process is repeated. Consequently, the closed-loop architecture as depicted in figure 1.2 consists of the following primary ingredients:

- Knowledge of the full state of the plant which is either directly measurable or estimated by a state observer
- Numerical simulation of the system trajectories based on an internal model
- Constrained optimization algorithm

Even if the basic design of a model predictive controller is rather intuitive due to the simple definition of performance criteria and constraints in comparison to, for example, a PID controller, it's nowadays application in practice is still limited to slow processes [PH14]. Especially the underlying numerical simulations and (iterative) constrained optimization are usually difficult to solve for generic nonlinear respectively nonconvex problem formulations. Also, its online realization is computationally intensive. Once these problems are adequately solved, *MPC has the potential to replace PID, H_∞ and LQR controllers in industrial applications* to maximize performance while maintaining increasing system complexity. To this end, recent researches focus on developing numerically efficient MPC realizations to enable their utilization also for faster systems such as mechatronic systems. However, these realizations mainly consider MPC with quadratic cost terms, in particular, the minimization of the state error

and control effort, which limits its application to areas that require true time-optimality. The already excellent and predictable potential of MPC motivates the dedicated expansion to realize time-optimal control tasks in future industrial applications.

1.2. Contribution and Outline

This thesis contributes to the following two central research questions:

1. How to design time-optimal controllers for nonlinear systems under the explicit consideration of constraints, particularly in the framework of MPC?

2. How to realize time-optimal MPC in a computationally efficient way?

The first research question is addressed by introducing direct transcription methods that first discretize and then optimize the subordinate optimal control problems. In contrast to conventional methods, the underlying discretization grid is explicitly treated as temporally variable during optimization. This principle procedure makes it possible to replace the entire transition time by local time information which is defined only between two successive discrete states. Accordingly, minimizing the local time information leads to the overall minimum transition time. The resulting optimal control problems retain their inherent sparse structure, even if time is explicitly regarded as an optimization parameter. Many iterative optimization algorithms already take sparse algebra into account, but the calculation of first- and second-order derivatives in each optimization step is still very demanding. A hypergraph representation of the optimal control problems inherently captures their sparse structure and enables efficient structure-depending finite difference computations. The hypergraph reconfigures itself online with little computational effort, especially for different problem dimensions, which later proves to be decisive for the proposed time-optimal control approaches.
The following list provides a brief overview of the chapters contained in this thesis including their central contribution:

Chapter 2: This chapter summarizes the current state of the art of time-optimal MPC and related work done in the context of optimal control as well as MPC in general. A particular focus lies on optimal control methods with free final time and variable switching points from optimal (feedforward) control.

Chapter 3: Preliminaries and key definitions which are necessary for the following chapters are summarized. In addition to the classification of different control tasks for time-optimal control, the mathematical foundation for sampled closed-loop systems is provided. Also, the general continuous-time time-optimal control problem is defined which serves as an essential basis for all direct transcription methods presented in the following chapters. The chapter also presents two benchmark systems and a real experimental system which are referred to in numerous examples throughout the thesis.

Chapter 4: This chapter presents a global uniform discretization grid with a *single temporal optimization parameter*. Several direct transcription methods are defined, in particular multiple shooting, Hermite-Simpson collocation and collocation based on finite differences. This chapter also highlights some effects that can occur when collocation with higher-order control representations is imposed.

Chapter 5: The previously defined global uniform discretization grid is now integrated with state feedback. First, various challenges and issues for its direct closed-loop application are discussed. Then, a shrinking horizon grid adaptation scheme is presented for which (practical) asymptotic stability results are derived under reasonable assumptions. The grid adaptation strategy is further extended to maintain a desired temporal resolution. As known from general time-optimal control, the control sequence is affected by chattering while stabilizing the system at the steady state. To remedy this, the controller based on variable discretization grids is either embedded in a dual-mode formulation or supplemented by a quadratic form cost function to achieve smooth stabilization.

Chapter 6: This chapter introduces two further variable grid representations based on *multiple temporal optimization parameters*. The first approach enforces uniformity by additional equality constraints. Even though the solution is identical to the global uniform grid, its underlying sparsity structure is different as it is pointed out later in Chapter 7. The second approach omits additional uniformity constraints and tries to achieve a quasi-uniform grid in terms of a least-squares cost function. Furthermore, the chapter provides several examples and remarks regarding grid adaptation and closed-loop integration.

Chapter 7: This chapter presents the hypergraph representation to exploit sparsity for derivative computations as mentioned above. Originally, the hypergraph is borrowed from the context of robotics and this chapter extends it to nonlinear programs arising in MPC. The applicability is not limited to time-optimal control problems. The procedure seamlessly extends to general optimal control and MPC problems.

Chapter 8: This chapter provides extensive benchmark results and a comparative analysis of the previously defined variable grid approaches. Furthermore, the concepts are compared to related and recent methods in the literature for both the open-loop and the closed-loop performance.

Chapter 9: Based on a modified multiple shooting approach, this chapter proposes a non-uniform grid which is well suited for control systems that exhibit the bang-singular-bang property. Practical stability results are provided under milder conditions than for the uniform grid. Furthermore, a proposed grid adaptation scheme seeks for the ideal number of switching points online.

Chapter 10: This chapter summarizes the central results, provides concluding remarks for the developed concepts and suggests possible directions for future works.

In addition, the appendix contains extensions to the proposed approaches and further supplementary material.
The following list summarizes the most important contributions and highlights of this thesis:

1. Time-optimal control via direct transcription and variable discretization grids:
   - Global uniform grid,
   - Local uniform grid,
   - Quasi-uniform grid,
   - Non-uniform grid for bang-singular-bang control.
2. Analysis and comparison of several numerical schemes to approximate the system dynamics in terms of time-optimal control
   - Multiple shooting,
   - Collocation via finite differences,
   - Collocation via quadrature (Hermite-Simpson).
3. Feedback control:
   - Theoretical results on stability and recursive feasibility,
   - Practically motivated extensions and realizations.
4. Structural sparsity exploitation and online optimization:
   - Hypergraph formulation for sparse derivative computations,
   - C++ software framework for fast MPC and direct optimal control,
   - Specialized sequential quadratic programming algorithm.

2
Related Work

Model predictive controllers solve an optimal control problem at each sampling interval. Consequently, their development, application, and success are closely related to advances and achievements in the field of optimal control and optimization in general. In particular, numerical approaches are preferred as they are suitable for more generic and complex applications. This chapter gives a brief overview of the current state of the art in time-optimal MPC and the most important areas that are decisive for its development. These include feedforward optimal control, optimal control with free final time, model predictive control (especially in the context of continuous-time, nonlinear systems and real-time optimization).

2.1. Optimal Control Methods

The main objective of optimal control is to find a control policy that operates or controls a system under certain optimality criteria and constraints. While for certain problem formulations analytic solutions exist to form a closed-loop control law, the more general goal is to find an open-loop control policy (numerically). The following overview mainly focuses on continuous-time optimal control formulations as they form the basis for methods presented in this thesis. Continuous-time optimal control problems are expressed in terms of cost and constraint functionals. Their solution is a control trajectory and thus a function of time. Methods for solving such problems are categorized into three main types: dynamic programming, indirect and direct methods (refer to Figure 2.1). For analytic solutions, especially for the important class of unconstrained infinite-horizon optimal control of linear systems with quadratic form cost functions, commonly known as *Linear Quadratic Regulator* (LQR), the reader is referred to [Kal60; Wil71].

2.1.1. Dynamic Programming

Dynamic programming is based on the principle of optimality developed by Bellmann [Bel57]. The principle states that "an optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions" [Bel54].

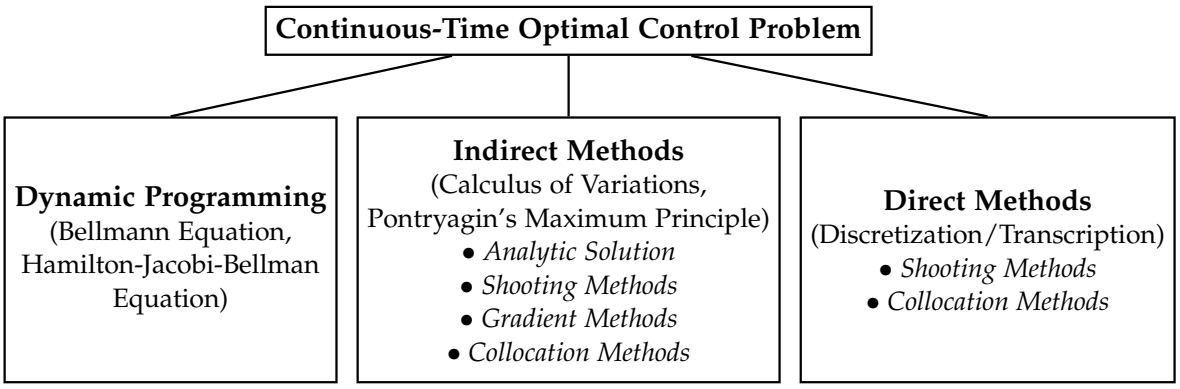


Figure 2.1.: Overview of different methods to solve continuous-time optimal control problems.

Consequently, the optimal control problem is divided into several subproblems. For discrete-time systems this fact is mathematically formulated by the Bellmann equation. In the case of continuous-time systems, a partial differential equation which is referred to as Hamilton-Jacobi-Bellmann equation has to be solved. A collection of established dynamic programming methods is summarized in [Ber95]. Dynamic programming methods usually require the sampling of the entire time, state and control space which guarantees to find the globally optimal solution, but on the other hand makes them intractable for larger system dimensions. This is often referred to as curse of dimensionality [Pow11]. Approximate dynamic programming methods try to relax the curse of dimensionality [Pow11]. This can be achieved, for example, by applying function approximations to reduce the grid size [Grü97], or by iteratively examining samples in the grid close to the actual solution [Luu10].

2.1.2. Indirect Methods

Indirect methods transform the continuous-time optimal control problem into a set of ordinary differential equations and boundary conditions using calculus of variations. The solution is then obtained by solving the resulting boundary value problem (first-order necessary conditions), which consists of state and adjoint variables. Pontryagin's maximum principle extends the necessary condition for optimal control problems with constraints [Pon87]. Numerically, the boundary value problem is solved with shooting or collocation methods, similar to the direct methods. Therefore indirect methods are often referred to as *first optimize and then discretize* methods. However, they solve simultaneously for the continuous state and adjoint variables. The latter complicates its initialization with a meaningful initial guess, since an intuitive physical interpretation is missing. In addition, the domain of convergence is usually smaller compared to dynamic programming and direct methods [Bet98]. On the other hand, indirect methods are still frequently used due to their high accuracy and fast convergence close to the optimal solution compared to direct methods [Pas12].

2.1.3. Direct Methods

Direct methods respectively direct optimal control or direct transcription methods *first discretize* the original problem representation *and then optimize* the resulting ap-

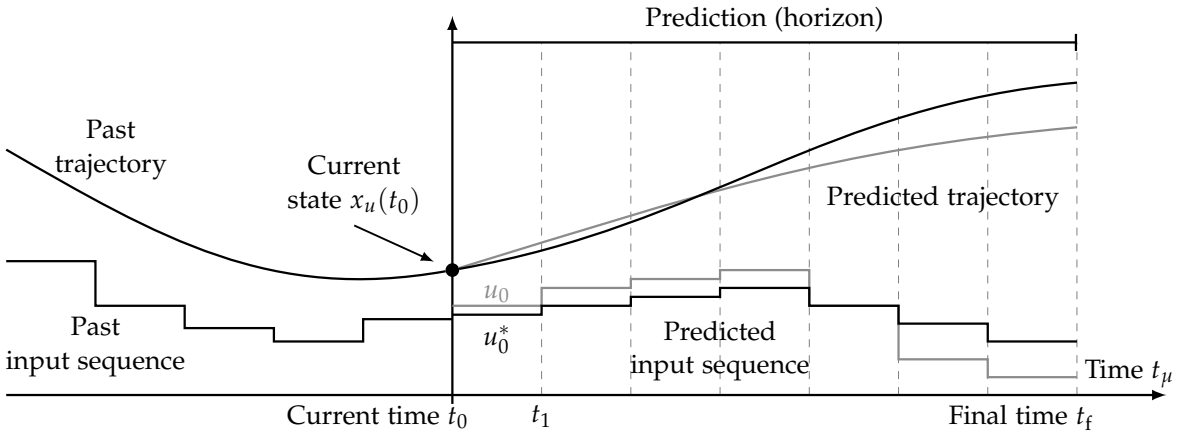


Figure 2.2.: Illustration of the single shooting method: Example initialization (gray) and converged solution (black). The first control is denoted by u_0 and u_0^* its optimal value.

proximation. By discretization of the continuous-time optimal control problem it is transformed into a (non-)linear program, which is subject to a finite set of optimization parameters, especially states and controls at discrete timestamps. Consequently, cost and constraint functionals are replaced by ordinary functions. A wide range of established constrained optimization algorithms exists to solve nonlinear programs numerically [NW06].

Direct methods are further categorized into sequential and simultaneous approaches. A well known sequential method for continuous-time problems is single shooting. Single shooting defines the control trajectory in terms of piecewise constant controls subject to optimization. The corresponding state trajectory is then obtained by solving the initial value problem numerically with respect to a given initial state, all controls and the system dynamics equations [Bet98]. Consequently, an iterative optimization algorithm simulates the entire state trajectory at least once in each iteration to evaluate cost functions and constraints before computing a new update of the control sequence. This procedure is repeated until convergence (refer to Figure 2.2). The major drawback of this approach is its limited numerical stability and usually low convergence. The intermediate results of the optimization are very sensitive to changes at the beginning of the control sequence that cause significant nonlinear changes at the end [Bet98].

On the other hand, simultaneous approaches discretize and optimize both states and controls. For example, multiple shooting discretizes the state trajectory on the same or coarser grid than the controls [BP84]. These introduced state variables are explicitly considered as optimization parameters called shooting nodes and the resulting grid partitions are called shooting intervals (refer to Figure 2.3). Optimization algorithms solve an isolated initial value problem for each shooting interval with the shooting node as the starting point. The nonlinear program is extended by additional deflection constraints that enforce connectivity among shooting intervals. Although the number of optimization parameters to be determined is larger than for single shooting, convergence is usually faster [Bet98]. Furthermore, the optimization problem has a sparse structure which is exploited in [Lei+03].

Direct collocation constitutes another well established simultaneous approach which

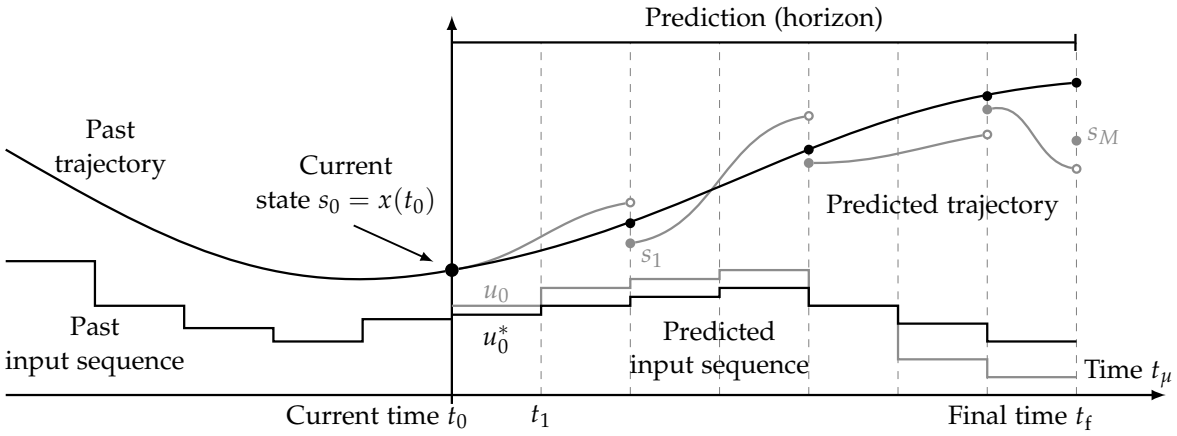


Figure 2.3.: Illustration of the multiple shooting method: Example initialization (gray) and converged solution (black). Shooting nodes are denoted as $s_0, s_1, \ldots, s_M$.

discretizes both the state and control trajectory. The basic idea is to approximate the system dynamics and cost functions using a predefined set of basis respectively kernel functions. The system dynamics equations are fulfilled at the discrete time instants, which is enforced by adding a set of algebraic equality constraints to the nonlinear program [Tsa+75]. Established candidates for basis functions are finite difference approximations and quadrature rules, which usually lead to piecewise linear, quadratic or cubic polynomials (splines) for the states and control trajectory [Str93; Kel17]. An example with a cubic Hermite spline as state and piecewise linear spline as control trajectory is shown in Figure 2.4. Direct collocation usually requires a larger number of optimization parameters in comparison to multiple shooting. Note, numerical integration with step size control can be easily integrated in multiple shooting, but collocation requires to adapt the whole temporal grid or the basis functions itself to achieve higher accuracies [Sag06]. However, whereas multiple shooting is usually preferred for optimal control problems with relatively simple controls, collocation methods achieve higher accuracies for tasks in which more complex control trajectories are necessary. For the sake of completeness, two specializations of direct collocation have become established in recent decades: Orthogonal collocation in which basis functions vary along the temporal grid (for instance polynomials of different orders) [Bie84]; and pseudospectral optimal control methods which define a single but higher-order polynomial for the complete grid. Pseudospectral methods converge with a spectral (exponential) rate which is faster than any polynomial rate [Hes+07; RK12].

2.1.4. Free Final Time, Switching Points and Time-Optimal Control

The majority of optimal control problems are solved according to a fixed temporal grid such that only controls and states are optimized. On the other hand, there are many applications in which the temporal grid or the final time must also be optimized. For example, the time required for guiding a system to another state is unknown in advance or the optimal control problem could be infeasible for the specified final time. The switching points of the control sequence (which coincide with the grid points)

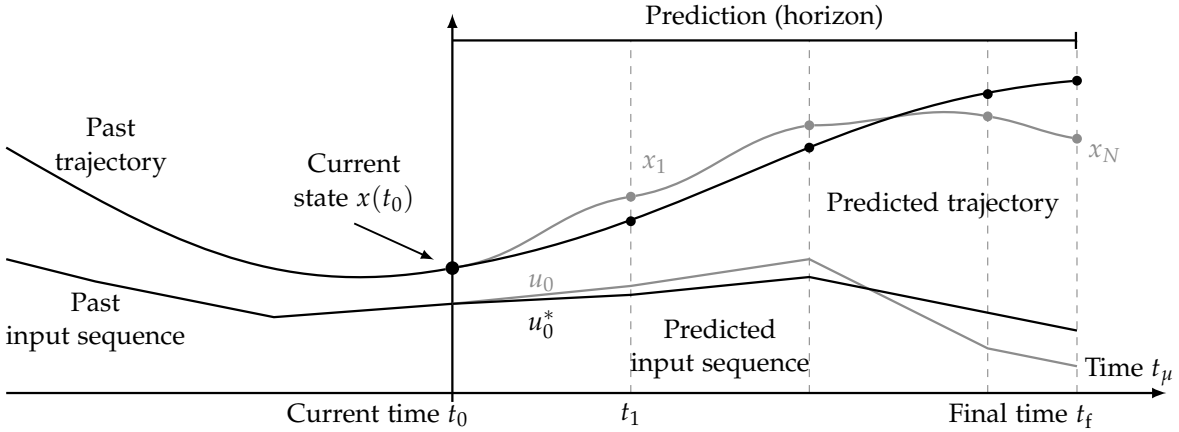


Figure 2.4.: Illustration of a collocation method: Initialization (gray) and converged solution (black). A cubic spline is selected for the state trajectory and a linear spline for the control. States at grid points are denoted by $x_0, x_1, \ldots, x_N$.

also do not correspond to those of admissible solutions.

In time-optimal control, the total transition time respectively the final time itself is subject to optimization compulsively. Many plants are controlled either at control or state limits. Thus many practical time-optimal control problems consist of either bang-bang, bang-singular-bang or a small finite set of piecewise constant controls. With such problems, the number of effective switching points is often significantly smaller than the ordinary temporal resolution of the a priori defined discretization grid. Feldbaum's well-known theorem states that the time-optimal control sequence of an unconstrained linear system with state dimension p and only real eigenvalues exhibits at most p switches [Pon87]. Sussmann provides conditions for nonlinear control-affine systems with bounded controls such that the resulting time-optimal control is bang-bang [Sus79]. Furthermore, he proves that nonlinear two-dimensional systems with a single affine control are of bang-singular-bang type [Sus87b; Sus87a]. Even though proving the bang-singular-bang property for nonlinear systems is difficult or intractable in general, many practical applications and experiments reveal bang-singular-bang control sequences.

A numerical method for time-optimal control of linear systems with constraints is presented in [Fat68] and [TK71]. The approach partitions the complete time interval t_f into N uniform subintervals of length t_f/N and represents the control trajectory as piecewise constant with respect to the subintervals. Also, constraints are imposed only at the grid points of the subintervals. The so-called set of attainability describes all initial states from which admissible control functions produce admissible trajectories. With this convex set, an auxiliary linear program is defined which is solved for example using the simplex method [TK71].

In order to deal with the free final time and hence nonlinear time-optimal control problems, a time-scaling respectively time transformation is presented and applied in [QD73; Jen+91; Teo+91]. The control trajectory is parameterized as a sequence of piecewise constant controls. An overview of different canonical forms of optimal control problems including the time transformation approach is presented in [II02].

Maurer and Oberle provide second-order sufficient conditions for problems with a free final time based on the proposed time transformation [MO02]. Refer to Appendix C.1.1 for the mathematical formulation of the approach.
Teo et al. present an optimal control formulation in which the temporal grid and its individual grid partition lengths are also subject to optimization and hence includes variable switching points [Teo+91]. The authors propose two alternative parameterizations: Either the actual controls are kept constant and just the switching points are determined or both controls and the switching points are subject to optimization. The authors provide non-smooth gradients of the cost functionals and constraints which are derived based on the results in [Has76] and which depend on first-order necessary conditions (Hamiltonian). Due to the discontinuous gradients (required for the proposed gradient-based indirect method) and the discontinuity of the system dynamics equations at the switching points, the approach was "never implemented for a practical problem" [Teo+99].
Later in [Teo+99] the so-called control parameterization enhancing transform for constrained optimal control problems is presented. Hereby, the original grid and switching times are mapped on to a uniformly spaced grid in a new time scale similar to the previously mentioned time transformation approach but with individual scalar parameters for the different grid partitions. The grid is now fixed with respect to the new time scale (usually every grid partition is of length 1) and the system dynamics equation is transformed similarly. Hence standard numerical optimal control algorithms can be utilized. This method is applied to time-optimal control problems in [Lee+97] which involves the transformation and minimizes the scalar time mapping function. In [Lee+99] the control parameterization enhancing transform is applied to discrete-valued control problems and in [Li+06] to switched systems. Further applications are summarized in [Reh+99]. Related work in [Vos10] formulates a direct method for bang-bang control and provides first- and second-order conditions as well as first- and second-order variational derivatives of the state trajectory with respect to the switching times.
An approximate time-optimal solution to an optimal control problem tailored for flexible structures with linear dynamics is provided in [Alb02]. The approach utilizes an indirect method to solve the optimal control problem. The infinite time-derivative at the switching times is replaced by a finite magnitude. For single-input nonlinear systems, [Kay03] approximates the time-optimal control problem in the arc time space (duration of arcs). The authors propose a dedicated approach to seek feasible but not necessarily optimal switching points [KN96]. With an application to the lap-time minimization of race cars, an optimal control formulation with different dynamics models is presented in [KS10; KS12].
Kashiri et al. propose an iterative indirect method to solve time-optimal control problems [Kas+11]. As for indirect methods in general, the method heavily relies on a proper initialization for which the authors develop a dedicated strategy.

2.2. Model Predictive Control

The development of MPC concepts began in the 1960s. State space models became popular for the controller design, in particular the LQR [Kal60; Wil71] and its extension by a Kalman filter for state estimation which is referred to as linear quadratic Gaussian regulator [Kal60; Ath71]. However, even if the solution is obtained analytically, the approach is restricted to quadratic cost functions and is only applicable to unconstrained linear systems.
The receding horizon concept enables constrained real-time optimization during feedback control. Lee and Markus formulate the fundamental concept of MPC:

> "*One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement. The procedure is then repeated.*" [LM67]

Early practical MPC realizations are defined in terms of impulse response models [Ric+67; Ric+77] and step response models (dynamic matrix control) [CR80; QB03].
Albeit not directly related to MPC but to predictive control in general, another early concept called internal model control requires a model of the plant dynamics during runtime in order to realize a pure feedforward control scheme in the absence of model mismatch and disturbances. In practice, an additional feedback part compensates for those errors [GM82; Mor83].
Since the late 1990s and early 2000s the majority of research in MPC considers state space models either in discrete time or as sampled-data systems. For an overview of influential contributions in MPC at that time, the reader is referred to [QB97; Hel+98; MH99; QB03; MS04]. The fundamental stability results of MPC are summarized in [May+00; GP17; Raw+17]. Especially the quasi-infinite horizon scheme guarantees stability for nonlinear systems under certain conditions [CA98; FA03; HL02]. Stability results on considering variable final times generally in the context of MPC are provided in [MM93; May95].
To apply MPC to fast systems, such as mechatronic systems, the interest in efficient numerical realizations has grown considerably in the last two decades. The majority of recent approaches to continuous-time dynamics focus on direct methods that solve the underlying optimal control problem using a nonlinear program with finite parameters as described in Section 2.1.3. Diehl et al. propose multiple shooting to solve continuous-time system dynamics in MPC [Die+02]. Multiple shooting usually achieves a better convergence rate due to its sparse albeit larger problem structure. The real-time iteration (RTI) scheme reduces the computational effort in each sampling instance by merely applying a single warm-started sequential quadratic programming (SQP) step at each sampling interval [Die+05; Gro+16]. Kouzoupis et al. combine the RTI scheme with different first-order methods in order to analyze its application to embedded nonlinear MPC [Kou+15a]. To handle stiff systems, implicit integrators are preferred over explicit ones. Recently, lifted integrators based on the inexact Newton method

reduce the overall computational cost [Qui+15a; Qui+15b]. Zanelli et al. modify the RTI scheme to significantly reduce the number of variables subject to optimization by applying a backward Riccati sweep to a subset of the horizon [Zan+17]. Other approaches are based on early-terminating interior point methods that also exploit the sparsity of the problem structure in order to allow an efficient computation [WB08; WB10; Ric13; Pak+13]. Simultaneous methods and warm-starting techniques are key to efficiency. Warm-starting in the context of real-time optimization assumes that the (updated) solution of the previous sampling interval provides a proper and reasonable initialization for the optimization in the current step. Depending on the warm-start strategy, the initialization is improved a priori, for example, by shifting the temporal grid. For quadratic programming, the sparse structure of the nonlinear program is exploited by condensing techniques, for example in [Jer+11; Kou+15b; Fri+16]. Another way to increase efficiency is to remove inactive and obsolete constraints online, as proposed for linear systems in [JM13; Jos+15; Jos+17].

An efficient method based on projected gradients is presented in [GK12] within a real-time capable MPC scheme. Regarding linear systems, the approach by Zeilinger et al. guarantees stability and feasibility under hard real-time conditions [Zei+14]. Tailored gradient methods are recently applied to systems governed by partial differential equations [Rhe+14]. Saturation functions transform the underlying optimal control problem into an unconstrained one. A two-stage transformation technique with interior penalties is applied to nonlinear MPC in [KG16] to solve an unconstrained auxiliary MPC problem using an efficient gradient method. Nielsen et al. focus on the parallelization of the Newton step arising in both active-set and interior-point solvers by exploiting the sparse structure of the nonlinear program as well [NA15].

Move-blocking MPC methods reduce the number of control actions while keeping the horizon lengths relatively large. Either of these methods keep the control constant beyond the so-called control horizon or multiple controls in between the horizon are kept constant [Cag+04; Cag+07; GI10]. Move-blocking with a single degree of freedom in control and a tailored optimization strategy is considered in [Mak+18b].

Most of the above methods have in common that they compute at least first-order and often second-order derivatives of the cost function respectively constraints. If closed-form analytic derivatives are not available, they are computed automatically in numeric form. Often, the numeric computation relies on finite differences, usually central differences, to achieve the desired precision. On the other hand, automatic differentiation (AD) recently emerged as a popular and ubiquitous approach for computing exact derivatives as in the ACADO toolkit [Hou+11b]. AD inherently retains the sparse structure of the optimal control problem and avoids the numerical evaluation of structured zero elements in first- and second-order derivatives. CasADi [And13] constitutes a mature and efficient open-source AD framework frequently reported in the MPC literature as the preferred tool in the realm of optimal control. The framework of AD provides an elegant and simple way to formulate the optimal control problem by merely formulating the mathematical expressions of the nonlinear program while preserving its sparse structure.

Automatic code generation strategies became popular during the last years in order to generate tailored and efficient C code [Vuk+13]. Code generation plays an impor-

tant role in order to realize MPC on embedded systems [Kuf+15]. The previously mentioned AD frameworks feature code generation capabilities.
Explicit MPC solves the general parameterized optimization problem offline. The optimal control action for the current state and target is extracted from a precomputed lookup table. The explicit approach requires the a priori discretization of the operational space. A well-known bottleneck is that the memory requirements grow exponentially with the dimension of the state space, thus limiting the approach to low-order dynamics. Selected methods for linear systems are presented in [Kou+11; MK11] and for nonlinear systems in [Joh04; Pin+13; Tri+16].

2.3. Time-Optimal Model Predictive Control

The previously mentioned MPC approaches mainly consider cost functionals that depend on the state and control trajectory, especially a quadratic form cost. Thus, grids for the discretization of continuous-time dynamics are usually fixed during runtime.
The literature mentions time-optimal MPC research rarely. Nevertheless, some contributions and applications rely on the basic time transformation as discussed in Section 2.1.4. Theoretical stability results for controllers considering the time transformation are intractable and are hence not yet available in the literature (see also [Ver+17]). Zhao et al. provide a time-optimal MPC scheme for the control of a spherical robot based on the time transformation [Zha+04]. A hybrid cost function that also considers quadratic form cost achieves stabilization. The transformed time is bounded from below close to the target state such that only the quadratic form cost becomes active. Verschueren et al. compute time-optimal motions along a Cartesian path for robotic manipulators [Ver+16b]. Time transformation is applied to the underlying time-optimal control problem to map states and controls onto a fixed integration grid. A nonlinear MPC method for time-optimal point-to-point transitions which does not rely on time transformation is presented in [Van+11b; Van+11a]. The method called TOMPC for time-optimal MPC minimizes the settling time in a two-layer optimization routine. The settling time is defined in terms of the horizon length N. The outer loop incrementally reduces the horizon of the control sequence N until the inner loop nonlinear program with a standard quadratic form cost fails to generate a feasible solution for the allocated time horizon. Since the cost function minimizes the distance of discrete states to the final state, the solution with the shortest feasible horizon is quasi time-optimal. Due to the lower bound on the time horizon, the algorithm behaves like a conventional MPC in the vicinity of the final state and therefore guarantees stability. The computation time strongly depends on the initial estimate of the settling time, as it determines the number of iterations in the outer loop time horizon reduction. Refer to Appendix C.1.2 for a mathematically more detailed description of TOMPC. The work of [PP14] presents the analysis and design of time-optimal feedback control with variable horizon lengths N.
An alternative approach that follows a reference path in minimum time is presented in [Lam12]. Time-optimality is nearly achieved in case of long time horizons. A time-op-

timal approach for linear systems is presented in [Bes+09].
An approach that considers ℓ_1-norm cost functions for linear systems is presented in [Hom16]. For general nonlinear systems, Verschueren et al. proposes a stabilizing time-optimal MPC approach based on a weighted ℓ_1-norm cost [Ver+17]. The approach considers discrete-time and sampled-data models and guides the system towards a target state (equilibrium) in minimum-time and stabilizes it there. It is required that the horizon length N is sufficiently large such that the target state is reachable within N time steps. The single-stage optimization, as well as the milder assumptions on N, are superior in comparison to TOMPC. Since the ℓ_1-norm is non-smooth, in every practical implementation it is replaced by a smooth representation consisting of additional slack variables. A more detailed description is provided in Appendix C.1.3.
In applications such as race car automatic control, tailored MPC methods minimize the lap time [KS10; TC10; Ver+14; Ver+16a]. Explicit MPC is extended to the offline computation of time-optimal tasks for linear time-invariant and piecewise affine systems [Gri+05]. An approximation based on Voronoi diagrams for nonlinear systems is provided in [Rai+12].
Table 2.1 provides a brief categorization of the related point-to-point (and stabilizing) methods, also in comparison to the variable discretization grids.

Table 2.1.: Properties of minimum-time point-to-point MPC methods. Time transformation is considered without hybrid costs. N^* denotes the minimum grid size / horizon length.

	Time Transformation	TOMPC	ℓ_1-Norm Approach	Variable Discretization Grids
Type of Cost	Total time	Quadratic form	Weighted ℓ_1-norm	Time interval(s)
Temporal Grid	Fixed (but scaled dynamics)	Fixed (but adapted)	Fixed	Variable
Horizon	Shrinking	Shrinking	Receding	Shrinking
System Model	Cont.-Time	Discr./Cont.-Time	Discr./Cont.-Time	Cont.-Time
Stability Results	No	Yes	Yes	Yes with grid adaptation (practical)
Additional Opt. Parameters	Time t_f	Horizon length N	Slack variables ($N\cdot$ state dim.)	Time interval(s) (either 1 or N)
Additional Constraints	No	No	Yes (for smoothness)	No
Additional Design Parameters	No	No	Yes (weight)	Yes (lower bound on time)
Grid Size $N < N^*$	Suboptimal	Infeasible	Infeasible	Suboptimal

3
Fundamentals

The notation throughout this thesis generally follows established conventions, parameter names and symbols as used in the majority of MPC literature, as demanded in [May14]. These are mainly based on the book [GP17] and its predecessor edition.

3.1. Dynamic System

A continuous-time, nonlinear, time-invariant system with state trajectory $x : \mathbb{R} \mapsto \mathcal{X}$ and control trajectory $u : \mathbb{R} \mapsto \mathcal{U}$ is defined by the finite dimensional ordinary differential equation:

$$\dot{x}(t) = f\big(x(t), u(t)\big). \tag{3.1.1}$$

Throughout this thesis, state space $\mathcal{X}$ and control space $\mathcal{U}$ are assumed as $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{U} = \mathbb{R}^q$ with state vector dimension $p \in \mathbb{N}$ and control vector dimension $q \in \mathbb{N}$ respectively. Depending on the actually used direct optimal control strategy and optimization algorithm, more general metric spaces can be considered. Function $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$ defines a nonlinear mapping of the state and control trajectory, $x(t)$ and $u(t)$ respectively, to the state velocity $\dot{x}(t)$ embedded in $\mathcal{X}$. Depending on the actual application, limits on the states or controls might be enforced by the controller. The restricted state space is defined as $\mathbb{X} \subseteq \mathcal{X}$. Accordingly, the restricted control space is denoted as $\mathbb{U} \subset \mathcal{U}$. States and controls are called admissible if $x(t) \in \mathbb{X}$ respectively $u(t) \in \mathbb{U}$ holds.
For many control applications, the solution $x(t)$ to (3.1.1) contained in an open time interval $I \subseteq \mathbb{R}$ with initial value $x(t_\mathrm{s}) = x_\mathrm{s}$ and $t_\mathrm{s} \in I$ is of particular interest. Carathéodory's existence theorem addresses conditions for the existence and uniqueness of the solution [Hal80]. The following assumption states simplified conditions which are sufficient for the scope of this thesis:

Assumption 3.1.1 (Lipschitz Continuity)**.** The mapping $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$ is continuous and Lipschitz in its first argument (see Appendix A.1 for a definition of Lipschitz continuity). Furthermore, the control $u : \mathbb{R} \mapsto \mathcal{U}$ is supposed to be locally Lebesgue integrable respectively piecewise continuous on $t \in I$.

Since the system is time-invariant, the initial time t_s is fixed to $t_\mathrm{s} = 0$ without loss of

generality. The solution with initial value $x(t_s = 0) = x_s$ and $t \in I$ is then given by:

$$x(t) := \varphi\big(t, x_s, u(t)\big) = x_s + \int_{t_s=0}^{t} f\big(x(\tau), u(\tau)\big)\mathrm{d}\tau. \tag{3.1.2}$$

Hereby, notation $\varphi\big(t, x_s, u(t)\big)$ explicitly expresses the dependence on the initial state x_s and control trajectory $u(t)$.
MPC requires a system model (3.1.1) during runtime. Therefore, it is essential to distinguish between the mathematical description of the system dynamics, state and control trajectories for both the prediction, that is mainly the underlying optimal control problem, and the closed-loop behavior. If not stated otherwise, it is assumed that the prediction model equals the actual plant dynamics and hence no model mismatch occurs. Whenever state trajectories or control trajectories are presented without any direct relation to either the prediction or the closed-loop context, $x(t)$ and $u(t)$ are utilized as before.

3.2. Feedback Control

3.2.1. Classification of Feedback Control Tasks

Control tasks, especially in the context of MPC, are mainly categorized into four common types:

- *Point-to-point motion:* Hereby, a controller guides the system (3.1.1) from an initial state $x_s \in \mathbb{X}$ to a predefined terminal state $x_f \in \mathbb{X}$.
- *Set-point stabilization:* While for point-to-point motions the controller does not necessarily need to stabilize the system at x_f (preferably a steady state), a stabilization task explicitly does either asymptotically or exponentially under the influence of disturbances (it is assumed that the initial plant state x_s is already close to x_f for local stabilization).
- *Path following / contouring control:* In contrast to point-to-point motions, the controller guides the system alongside a geometrical reference path (without time information), for example by simultaneously minimizing the transition time or the contouring error.
- *Reference tracking:* The reference trajectory is time-parameterized and hence the controller needs to minimize both the spatial and temporal deviation to the reference.

Note, point-to-point control, set-point stabilization and reference tracking are all seamlessly addressed with a conventional stabilizing quadratic form MPC formulation. A time-varying reference trajectory is directly incorporated to compute the state and control deviations along the horizon. For a constant reference, point-to-point motions and set-point stabilization follow immediately. Instead, path following usually requires a parameterized path model embedded in the optimal control problem.

In the context of time-optimal control, the majority of control applications are based on point-to-point motions or path following in minimum time. Although stabilizing control is achievable, the controller reacts on small errors with chattering. As a result, mechanical actuators can be stressed and their durability is likely to be reduced. By definition, reference tracking excludes minimum-time objectives, since the time information is strictly contained in the reference trajectory.

3.2.2. Continuous-Time Time-Optimal Control Problem

This section introduces the general time-optimal control formulation for point-to-point motions which is afterwards integrated with state feedback in terms of MPC.

It is important to distinguish between the solution of the optimal control problem and the closed-loop evolution and to reflect that distinction in notation. Whenever the optimal control problem context must be preserved, the state trajectory is denoted as $x_u(t)$. Subscript u emphasizes its direct relation to the trajectory $u(t)$ obtained from the optimal control problem. For ease of notation, subscripts are omitted for the control trajectory and time variable. Note, that in closed-loop descriptions the plant input is already expressed by the control law (3.2.6). Time instances in closed-loop are indexed by n, time instances in prediction by k.

The general continuous-time optimal control problem with $t \in [t_0, t_\mathrm{f}]$ is defined by:

$$\min_{u(t)} \left[V_\mathrm{f}\big(x_u(t_\mathrm{f})\big) + \int_{t_0=0}^{t_\mathrm{f}} \ell\big(x_u(t), u(t)\big)\,\mathrm{d}t \right] \tag{3.2.1}$$

subject to

$$x_u(t_0 = 0) = x_\mu(t_\mu), \quad x_u(t) \in \mathbb{X}, \quad u(t) \in \mathbb{U}, \quad x_u(t_\mathrm{f}) \in \mathbb{X}_\mathrm{f},$$
$$\dot{x}_u(t) = f\big(x_u(t), u(t)\big).$$

Without loss of generality, the initial time is set to $t_0 = 0$. The state trajectory $x_u(t)$ follows from the solution to a boundary value problem defined by the system dynamics $\dot{x}_u(t) = f\big(x_u(t), u(t)\big)$, initial state $x_u(t_0)$, terminal constraint $x_u(t_\mathrm{f}) \in \mathbb{X}_\mathrm{f}$ and state $x_u(t) \in \mathbb{X}$ respectively control constraints $u(t) \in \mathbb{U}$. The initial state $x_u(t_0)$ is either directly measured or estimated from a state observer. The integrand $\ell : \mathcal{X} \times \mathcal{U} \mapsto \mathbb{R}$ denotes the running cost and $V_\mathrm{f} : \mathcal{X} \mapsto \mathbb{R}$ defines the terminal cost. Terminal cost $V_\mathrm{f}(\cdot)$ and terminal constraint $x_u(t_\mathrm{f}) \in \mathbb{X}_\mathrm{f}$ are beneficial to enforce stability [CA98; May+00].

For time-optimal control tasks the final time t_f is variable and subject to optimization such that the running cost simplifies to $\ell(\cdot) := 1$ and the terminal cost to $V_\mathrm{f}(\cdot) := 0$. The system is required to reach a desired terminal state $x_\mathrm{f} \in \mathbb{X}$ in minimum time such that the terminal region becomes a single state $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$. However, some applications relax the terminal condition just to fix a subset of state components at time t_f. Therefore, the more general notion $x_u(t_\mathrm{f}) \in \mathbb{X}_\mathrm{f}$ is kept in the remainder. The resulting

continuous-time time-optimal control problem is defined by:

$$\min_{u(t),t_\mathrm{f}} \; t_\mathrm{f} \tag{3.2.2}$$

subject to

$$x_u(t_0 = 0) = x_\mu(t_\mu), \quad x_u(t) \in \mathbb{X}, \quad u(t) \in \mathbb{U}, \quad x_u(t_\mathrm{f}) \in \mathbb{X}_\mathrm{f},$$
$$\dot{x}_u(t) = f\big(x_u(t), u(t)\big).$$

At first glance, the optimal control problem seems trivial as the objective function is linear and unbounded, but on the other hand t_f implicitly depends on the solution to the boundary value problem as defined by the constraints. Set $\mathbb{U}$ is considered compact to exclude trivial solutions with infinite control values. Note that the terminal set $\mathbb{X}_\mathrm{f}$ does not immediately ensures stability, as shown later.
Some further properties and assumptions are necessary for a meaningful formulation of the control task.

Definition 3.2.1 (Admissibility)**.** For initial value $x_\mathrm{s} \in \mathbb{X}$ and $t \in [t_0, t_\mathrm{f}]$, the control trajectory $u(t)$ and the corresponding trajectory $x_u(t)$ with $x_u(t_0) = x_\mathrm{s}$ are *admissible for x_s up to time t_f*, if

$$u(t) \in \mathbb{U}, x_u(t) \in \mathbb{X} \text{ for all } t \in [t_0, t_\mathrm{f}) \text{ and } x_u(t_\mathrm{f}) \in \mathbb{X}_\mathrm{f} \tag{3.2.3}$$

holds.

Local optimization solvers often assume that $\mathbb{X}$ and $\mathbb{U}$ are compact and convex (for example box constraints). The optimal control problem (3.2.2) is referred to as feasible if the resulting optimal state and control trajectories are admissible according to Definition 3.2.1. Closely related to feasibility is the notion of viability which implies feasibility. The following assumption extends the notion of a viable set $\mathbb{X}$ from [Grü02] to incorporate $\mathbb{X}_\mathrm{f}$ and the free final time t_f:

Assumption 3.2.1 (Viability)**.** The tuple $(\mathbb{X}, \mathbb{X}_\mathrm{f})$ is called *viable* if for each $x_\mathrm{s} \in \mathbb{X}$ there exists $t_\mathrm{f} \geq t_0$ and $u(t) \in \mathbb{U}$ such that $\varphi\big(t - t_0, x_\mathrm{s}, u(t)\big) \in \mathbb{X}$ and $\varphi\big(t_\mathrm{f} - t_0, x_\mathrm{s}, u(t)\big) \in \mathbb{X}_\mathrm{f}$ hold for all $t \in [t_0, t_\mathrm{f}]$.

Viability is often also called weak or controlled forward invariance and provides a means to define controllability in the presence of state and control constraints.
Pontryagin's maximum principle formulates a general necessary condition for optimality of (3.2.2) [Pon87]. Further necessary and sufficient conditions are given by the Hamilton-Jacobi-Bellman equation [Ber95]. [Lei81; Cla83] discuss the special treatment of terminal state constraints. Standard direct and indirect methods have difficulties in handling the variable final time t_f. Furthermore, the formulation does not exhibit any sparse structure, since t_f depends on the complete control trajectory $u(t)$ for $t \in [t_0, t_\mathrm{f}]$. The most common approach is to apply a time transformation in order to substitute the variable time interval $[t_0, t_\mathrm{f}]$ by the unit interval. Appendix C.1.1 provides a mathematical definition. The transformed problem is then solved with conventional direct and indirect methods. In contrast, the following chapters formulate direct transcription

methods that are based on variable discretization grids and do not explicitly rely on a time transformation. Direct methods assume that the numerical approximation of the system dynamics is chosen sufficiently precisely so that necessary and sufficient conditions of standard parameter optimization are applicable in favor of Pontryagin's maximum principle or the Hamilton-Jacobi-Bellman equation. Appendix A.4 summarizes these optimality conditions.

Error Analysis Any numerical solution to (3.2.1) approximates the system dynamics equation $\dot{x}_u(t) = f\big(x_u(t), u(t)\big)$. Particularly, there are two sources of numerical errors: Errors due to the choice of the transcription method and errors due to the solution to the nonlinear program [Kel17]. In order to quantify and compare both errors at once, a reference trajectory $x_{\mathrm{ref}}(t)$ is assumed to be available. The reference is either obtained from already approved approaches or by solving the original problem close to machine precision. Let $u^*(t)$ denote the solution to (3.2.1). The star emphasizes that the solution is optimal with respect to the specified transcription method and solver accuracy. Rather than referring to the approximated state trajectory $x_u(t)$, the actual open-loop state trajectory $\hat{x}_u(t)$ is then obtained by:

$$\hat{x}_u(t) = \varphi\big(t - t_0, x(t_0), u^*(t)\big) \quad \text{with} \quad t \in [t_0, t_\mathrm{f}]. \tag{3.2.4}$$

Hereby, the initial value problem is solved by a Runge-Kutta method with step sizes close to machine precision. The integral error of the ℓ_2-norm with respect to the reference $x_{\mathrm{ref}}(t)$ is evaluated by:

$$e_{\hat{x}}(t) = \int_{t_0}^{t} \|x_{\mathrm{ref}}(\tau) - \hat{x}_u(\tau)\|_2 \,\mathrm{d}\tau. \tag{3.2.5}$$

Throughout this thesis, (3.2.5) is evaluated with respect to a fine grid based on the available sampled data points of $x_{\mathrm{ref}}(t)$. Extrapolation with zero-order hold is applied in case one of the trajectories reaches its final state, since the desired target state usually constitutes a steady state. Note, $e_{\hat{x}}(t_\mathrm{f})$ defines an essential performance index for the benchmarks.

3.2.3. Nominal Closed-Loop System and Stability Definitions

In the context of MPC, continuous-time closed-loop control systems are generally defined in terms of sampled feedback control laws [MS04; GP17].
Let $t_{\mu,0} < t_{\mu,1} < \ldots < t_{\mu,n} < \ldots < \infty$ define the sampling instances with $n \in \mathbb{N}_0$ and $t_{\mu,n} \in \mathbb{R}_0^+$. Hereby, subscript μ indicates that the context belongs to the evolution of the closed-loop system. In every interval $[t_{\mu,n}, t_{\mu,n+1})$, optimal control problem (3.2.1) or any particular approximation provides an admissible control trajectory $u^*(t; t_{\mu,n})$ on time domain $t \in [0, t_\mathrm{f}]$. For clarity, the notation $u^*(t; t_{\mu,n})$ explicitly includes $t_{\mu,n}$ to indicate its relation to the closed-loop time instance $t_{\mu,n}$. The generic sampled control law $\mu : \mathcal{X} \mapsto \mathcal{U}$ is defined implicitly by:

$$\mu\big(x_\mu(t_\mu)\big) := u^*(t - t_{\mu,n}; t_{\mu,n}) \quad \text{for} \quad t_\mu \in [t_{\mu,n}, t_{\mu,n+1}). \tag{3.2.6}$$

Remark 3.2.1. The proper definition requires $t_\mathrm{f} \geq t_{\mu,n+1} - t_{\mu,n}$. Otherwise, system (3.1.1) reaches t_f and a proper remedy is to maintain $f(x_\mathrm{f}, u_\mathrm{f}) = 0$ with some $u_\mathrm{f} \in \mathbb{U}$ by defining $u^*(t - t_{\mu,n}; t_{\mu,n}) := u_\mathrm{f}$ for $t \in (t_\mathrm{f}, t_{\mu,n+1} - t_{\mu,n})$.

Considering the plant dynamics (3.1.1), the resulting closed-loop system with initial state $x_\mathrm{s} \in \mathcal{X}$ at time $t_{\mu,0}$ is defined by:

$$\dot{x}_\mu(t_\mu) = g_\mathrm{f}\big(x_\mu(t_\mu)\big) = f\Big(x_\mu(t_\mu), \mu\big(x_\mu(t_\mu)\big)\Big), \quad x_\mu(t_{\mu,0}) = x_\mathrm{s}. \tag{3.2.7}$$

According to (3.1.2), the state trajectory of the closed-loop system is obtained by solving the initial value problem:

$$x_\mu(t_\mu) := \varphi_\mu\big(t_\mu, t_{\mu,0}, x_\mathrm{s}\big) := \varphi\Big(t_\mu - t_{\mu,0}, x_\mathrm{s}, \mu\big(x_\mu(t_\mu)\big)\Big). \tag{3.2.8}$$

Remark 3.2.2. All theoretical investigations assume that the solution to the optimal control problems is available within zero computation time $\Delta t_\mathrm{cpu} = 0\,\mathrm{s}$. Obviously, the computational burden is large such that this assumption does not hold for real applications. In case $\Delta t_\mathrm{cpu} \ll \Delta t_{\mu,n}$, the overall performance is often well and quite similar to the ideal case. However, this does not hold in general. A possible way to take the computation time into account is to compensate the expected duration a priori at the beginning of each closed-loop sampling step by forward simulation [GP17].

The following definitions are crucial for the proposed time-optimal control formulations.

Definition 3.2.2 (Forward Invariance)**.** A set $Y \subseteq \mathcal{X}$ is called *forward invariant* for closed-loop system (3.2.7) if $\varphi_\mu\big(t_\mu, t_{\mu,0}, x_\mathrm{s}\big) \in Y$ holds for all $x_\mathrm{s} \in Y$ and all $t_\mu \geq t_{\mu,0}$.

Forward invariance maintains feasibility and stability for constrained state spaces $\mathbb{X} \subseteq \mathcal{X}$ and hence the case $Y = \mathbb{X}$ is of particular importance throughout this thesis. It is common to define stability properties for nonlinear systems by so-called comparison functions. These apply to both continuous-time and discrete-time systems and either take a stationary reference x_f or a time-dependent reference trajectory into account. However, this thesis focuses on stationary references x_f.

Definition 3.2.3 (Comparison Functions)**.** Comparison functions belong to either of these classes:

$$\begin{aligned}
\mathcal{K} &:= \{\alpha : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+ \mid \alpha \text{ is continuous and strictly increasing with } \alpha(0) = 0\},\\
\mathcal{K}_\infty &:= \{\alpha : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+ \mid \alpha \in \mathcal{K}, \alpha \text{ is unbounded}\},\\
\mathcal{L} &:= \{\delta : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+ \mid \delta \text{ is continuous and strictly decreasing with } \lim_{t\to\infty} \delta(t) = 0\},\\
\mathcal{KL} &:= \{\beta : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+ \mid \beta \text{ is continuous}, \beta(\cdot, t) \in \mathcal{K}, \beta(r, \cdot) \in \mathcal{L}\}.
\end{aligned}$$

Furthermore, for defining a proper region for local stability properties, consider the following ball with steady state x_f as origin and $\eta \in \mathbb{R}_0^+$:

$$\mathcal{B}_\eta(x_\mathrm{f}) := \{x \in \mathcal{X} \mid \|x - x_\mathrm{f}\| < \eta\} \tag{3.2.9}$$

in which the norm $\|x - x_\mathrm{f}\|$ denotes an arbitrary distance metric relating x and x_f. Utilizing these ingredients, local and global asymptotic stability can be defined as follows:

Definition 3.2.4 (Asymptotic Stability)**.** Let $x_\mathrm{f} \in \mathcal{X}$ be a steady state for (3.2.7) so that $g_\mathrm{f}(x_\mathrm{f}) = 0$ holds. Then x_f is *locally asymptotically stable* if there exist $\eta > 0$ and a function $\beta \in \mathcal{KL}$ such that the inequality

$$\|\varphi_\mu(t_{\mu,n}, t_{\mu,0}, x_\mathrm{s}) - x_\mathrm{f}\| \leq \beta(\|x_\mathrm{s} - x_\mathrm{f}\|, n) \tag{3.2.10}$$

holds for all $x_\mathrm{s} \in \mathcal{B}_\eta(x_\mathrm{f})$ and all $n \in \mathbb{N}_0$. Furthermore, x_f is *asymptotically stable on a forward invariant set* Y with $x_\mathrm{f} \in Y$ if (3.2.10) holds for all $x_\mathrm{s} \in Y$. In case $Y = \mathcal{X}$, x_f is called *globally asymptotically stable*.

Note, the definition considers only the sampled grid points $t_{\mu,n}$. Consequently, asymptotic stability implies that the smaller the initial distance from x_s to x_f is, the smaller the distance becomes for all future n (stability) and that the system approaches x_f as $n \to \infty$ (attraction). In practice, it is sometimes useful to relax the conditions of asymptotic stability, for example due to modeling errors or perturbations [GP17]:

Definition 3.2.5 (P-Practically Asymptotic Stability)**.** Let Y be a forward invariant set and let $P \subset Y$ be a subset of Y. Then a point $x_\mathrm{f} \in Y$ is *P-practically asymptotically stable on Y* if there exists $\beta \in \mathcal{KL}$ such that (3.2.10) holds for all $x_\mathrm{s} \in Y$ and all $n \in \mathbb{N}_0$ with $\varphi_\mu(t_{\mu,n}, t_{\mu,0}, x_\mathrm{s}) \notin P$.

The set P is often defined in terms of a ball (3.2.9). Especially in the context of point-to-point motions, the objective is to converge to a region around x_f given some tolerances. As an example, consider an autonomous vehicle that should reach a desired position and orientation in a parking lot. Hereby, it is crucial that forward invariance holds for the constrained state space of the controlled vehicle and hence it does not collide with any obstacle while converging towards the desired set point. The parking maneuver is successfully completed once the vehicle pose is within a certain tolerance of the nominal pose. Practical stability also plays a vital role for the results later in this thesis.

Remark 3.2.3. Note, the previous definitions are defined for grid points $t_{\mu,n}, n \in \mathbb{N}_0$. Obviously, it is required that the continuous-time solution of the initial value problem (3.2.8) does not deviate much in between two consecutive grid points such that

$$\|\varphi_\mu(t_\mu, t_{\mu,0}, x_\mathrm{s}) - x_\mathrm{f}\| \leq \beta(\|x_\mathrm{s} - x_\mathrm{f}\|, t_\mu) \tag{3.2.11}$$

holds for all $t_\mu \geq t_{\mu,0}$. Conditions for this so-called uniformly boundedness are summarized in [GP17] and are beyond the scope of this thesis. The work [NT04] provides a general overview of stability impacts concerning sampled-data systems.

3.3. Benchmark Systems

Two simulative benchmark systems and one experimental system are investigated for examples and in the comparative analysis of alternative methods. Even though the optimal and predictive control formulations consider nonlinear systems with an arbitrary number of controls and states, the evaluations and benchmarks in this thesis are designed for faster albeit smaller system definitions as they frequently occur in

mechatronics. While systems with large time constants are already considered for some decades, recent advances in real-time optimization enable much faster systems with sampling times in the range of 0.1 to 100 ms and state dimensions of 1 to 10 depending on nonlinearity and horizon length. Nonlinear systems with large horizon lengths (as required for time-optimal control) are computationally demanding such that much smaller dimensions are expected for sampling times of around 10 to 100 ms. The selected benchmark systems address specific properties like nonlinearities, one or two controls, a single equilibrium or infinite equilibria, partial or complete terminal equality constraints and the presence of bang-singular-bang controls depending on state bounds.

3.3.1. Van der Pol Oscillator

The Van der Pol oscillator constitutes a second-order dynamic system with nonlinear damping. Although the model has been originally developed for electrical circuits with triodes respectively vacuum tubes [Van26], its application now extends to other domains [GL12]. The literature commonly reports the Van der Pol oscillator as a common benchmark system for control applications or system analysis methods due to its properties. Depending on the external actuation force, the system exhibits deterministic chaotic behavior [CL47; GL12]. The oscillator is described by the following nonlinear ordinary differential equation with respect to the time-dependent function $y : \mathbb{R} \mapsto \mathbb{R}$:

$$\ddot{y}(t) - a_{\mathrm{vdp}}\big(1 - y(t)^2\big)\dot{y}(t) + y(t) = u(t) \tag{3.3.1}$$

with damping coefficient $a_{\mathrm{vdp}} \in \mathbb{R}_0^+$ and external actuation force $u : \mathbb{R} \mapsto \mathbb{R}$.
By defining the state vector $x(t) := \big(x_1(t), x_2(t)\big)^\mathsf{T}$, the nonlinear control-affine state space model is given by:

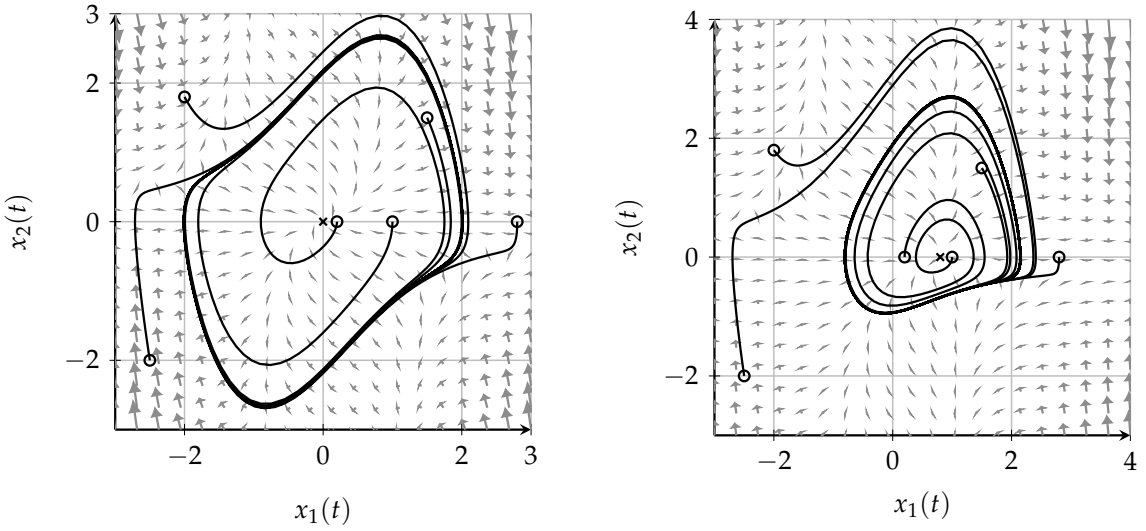
$$\begin{aligned} \dot{x}(t) &= f\big(x(t), u(t)\big) = \begin{pmatrix} x_2(t) \\ a_{\mathrm{vdp}}\big(1 - x_1(t)^2\big)x_2(t) - x_1(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t), \\ y(t) &= \begin{pmatrix} 1 & 0 \end{pmatrix} x(t). \end{aligned} \tag{3.3.2}$$

Note, for the Van der Pol oscillator the unrestricted state and control sets are $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{U} = \mathbb{R}$. For a given control reference $u_\mathrm{f} \in \mathcal{U}$, the system exhibits a unique steady state at $x_\mathrm{f} = (u_\mathrm{f}, 0)^\mathsf{T}$. A linearization of (3.3.2) at x_f and some arbitrary $u_\mathrm{f} \in \mathcal{U}$ is obtained using the Taylor series expansion:

$$\dot{x}_{\mathrm{lin}}(t) = Ax_{\mathrm{lin}}(t) + Bu_{\mathrm{lin}}(t) = \begin{pmatrix} 0 & 1 \\ -1 & a_{\mathrm{vdp}}(1 - u_\mathrm{f}^2) \end{pmatrix} x_{\mathrm{lin}}(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_{\mathrm{lin}}(t). \tag{3.3.3}$$

The state vector of the linear system is denoted by $x_{\mathrm{lin}}(t) = x(t) - x_\mathrm{f}$ and the control by $u_{\mathrm{lin}}(t) = u(t) - u_\mathrm{f}$ respectively. Furthermore, the state matrix A is determined by the Jacobian of the nonlinear system $f(\cdot)$ with respect to the states and evaluated at x_f and u_f that is $A = \mathrm{D}_x f(x_\mathrm{f}, u_\mathrm{f})$. Examination of the eigenvalues of A reveals the following behavior of the system near the steady state:

- A stable node if $a_{\mathrm{vdp}}(1 - u_\mathrm{f}^2) \leq -2$.



(a) Control reference $u_\mathrm{f} = 0.0$

(b) Control reference $u_\mathrm{f} = 0.8$

Figure 3.1.: Phase portraits of the Van der Pol oscillator for $a_\mathrm{vdp} = 1.0$ and $u(t) = u_\mathrm{f}$ with $u_\mathrm{f} \in \{0, 0.8\}$. Several individual solutions from different initial states (indicated by circles) are shown. The steady state $x_\mathrm{f} = (u_\mathrm{f}, 0)^\intercal$ is marked by a cross.

- A stable focus if $-2 < a_\mathrm{vdp}(1 - u_\mathrm{f}^2) < 0$.
- Harmonic motion if $a_\mathrm{vdp}(1 - u_\mathrm{f}^2) = 0$.
- An unstable focus if $0 < a_\mathrm{vdp}(1 - u_\mathrm{f}^2) < 2$.
- An unstable node if $a_\mathrm{vdp}(1 - u_\mathrm{f}^2) \geq 2$.

No exact analytical solutions to the nonlinear system are available. However, the oscillator has a unique and stable limit cycle for $a_\mathrm{vdp} \in \mathbb{R}_0^+$ and $u_\mathrm{f}^2 < 1$ [Jam74]. For the uncontrolled case this is easily verified by Liénard's theorem [Per91]. Consequently, the limit cycle exists in case the linearized system corresponds to an unstable focus respectively node. Figure 3.1 shows the phase portraits of the Van der Pol oscillator for two control reference values. Additionally, some explicit solutions from different initial states are presented. The initial states are chosen similarly for both settings and they are marked as circles. The (unstable) steady state is highlighted by a cross symbol. Obviously, the simulated solutions converge towards the limit cycle either if they start in the interior or in the exterior of the cycle. A larger value a_vdp increases the nonlinearity such that the sinusoidal wave deforms into a non-sinusoidal (sawtooth) wave.
A detailed control synthesis which discusses domains of controllability and switching curves for time-optimal bang-bang control of the Van der Pol oscillator based on Lyapunov theory and Prontryagin's maximum principle is presented in [Jam74]. Note, for the simulations and experiments in this thesis, controllability and reachability are ensured by obtaining feasible solutions from the nonlinear program solvers satisfying the first-order optimality conditions.

Whenever point-to-point motions for the Van der Pol oscillator are investigated in this thesis, the terminal region is set to the steady state x_f with the second component identical to zero such that $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$. The damping coefficient is set to $a_\mathrm{vdp} = 1.0$ and the constraint sets to $\mathbb{X} = \mathcal{X}$ and $\mathbb{U} = \{u \in \mathcal{U} \mid |u| \leq 1\}$ respectively. As a result, the limit cycle remains active and a potential steady state with $|u_\mathrm{f}| < 1$ represents an unstable focus that makes stabilization more difficult.

3.3.2. Simple Rocket System

The simple rocket constitutes a benchmark system which is for example utilized in [Hou+11a]. Consider Newton's second law of motion with inertial force F_inertial, thrust force F_thrust and drag force F_drag:

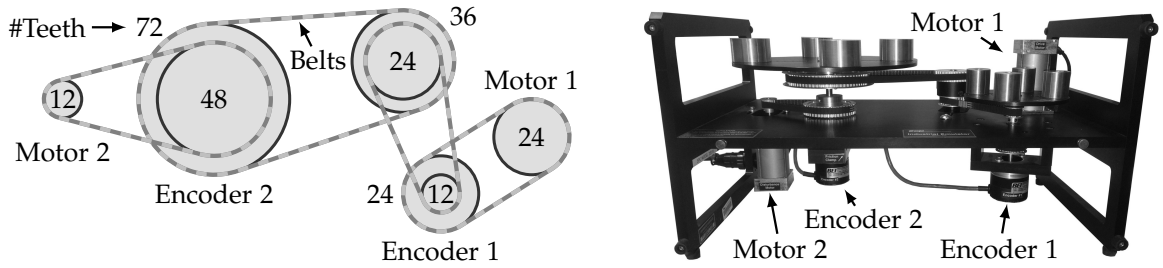
$$F_\mathrm{inertial} = F_\mathrm{thrust} - F_\mathrm{drag}, \tag{3.3.4}$$

$$m_\mathrm{r}(t)\dot{v}_\mathrm{r}(t) = u(t) - \frac{1}{2}c_\mathrm{drag}\dot{v}_\mathrm{r}(t)^2. \tag{3.3.5}$$

Hereby, $v_\mathrm{r} : \mathbb{R} \mapsto \mathbb{R}$ denotes the velocity of the rocket, $m_\mathrm{r} : \mathbb{R} \mapsto \mathbb{R}$ is the time-dependent mass and $F_\mathrm{drag} := u(t)$ with $u : \mathbb{R} \mapsto \mathbb{R}$ represents the input of the system. The constant c_drag is set to $c_\mathrm{drag} = 0.04$ according to [Hou+11a]. Note, the model neglects gravitational forces. Furthermore, the time-dependent mass is modeled such that it changes with respect to the square of the applied thrust: $\dot{m}(t) = -c_\mathrm{roc}u(t)^2$. In the remainder, the rate of change coefficient is set to $c_\mathrm{roc} = 0.01$. By further introducing the position of the rocket $s_\mathrm{r} : \mathbb{R} \mapsto \mathbb{R}$, the following state space model with state vector $x(t) := \big(s_\mathrm{r}(t), v_\mathrm{r}(t), m_\mathrm{r}(t)\big)^\mathsf{T}$ is defined:

$$\begin{aligned} \dot{x}(t) &= f\big(x(t), u(t)\big) = \begin{pmatrix} v_\mathrm{r}(t) \\ \frac{u(t) - 0.02\, v_\mathrm{r}(t)^2}{m_\mathrm{r}(t)} \\ -0.01\, u(t)^2 \end{pmatrix}, \\ y(t) &= \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} x(t). \end{aligned} \tag{3.3.6}$$

The unrestricted state and control sets are $\mathcal{X} = \mathbb{R}^3$ and $\mathcal{U} = \mathbb{R}$ respectively. Later on, when defining the restricted sets for the optimal control problems, the mass is assumed to be either positive or zero, that is $m_\mathrm{r}(t) \geq 0$. On the other hand, the thrust force is allowed to be applied either in the direction of flight or against the direction of flight, indicated by a positive or negative value $u(t)$. In any case the rocket cannot regain mass since $\dot{m}_\mathrm{r}(t) \leq 0$ holds even for $u(t) \leq 0$. It is straightforward to verify that system (3.3.6) can rest at any position $s_\mathrm{r}(t)$ if $u(t) = 0$ by setting $\dot{x}(t) = 0$ and hence exhibits an infinite number of steady states.
When defining a specific terminal state, for example, a particular target position $s_\mathrm{r,f}$ and target velocity $v_\mathrm{r,f} = 0$ (to ensure the final state is a steady state), the corresponding final mass is usually unknown in practice and choosing $m_\mathrm{r,f}$ arbitrarily might result in an infeasible problem. Therefore, the rocket system constitutes an ideal example for which the terminal constraint should not fix all components of the final state and hence $\mathbb{X}_\mathrm{f} = \{(s_\mathrm{r}, v_\mathrm{r}, m_\mathrm{r})^\mathsf{T} \in \mathbb{X} \mid s_\mathrm{r} = s_\mathrm{r,f}, v_\mathrm{r} = 0\}$. If not stated otherwise, constraint sets are set to $\mathbb{X} = \{(s_\mathrm{r}, v_\mathrm{r}, m_\mathrm{r})^\mathsf{T} \in \mathcal{X} \mid -0.5 \leq v_\mathrm{r} \leq 1.7, m_\mathrm{r} \geq 0\}$ and $\mathbb{U} = \{u \in \mathcal{U} \mid |u| \leq 1\}$ respectively.



(a) Schematic overview of the mechanical parts

(b) Picture of the system

Figure 3.2.: Experimental testbed *ECP Industrial Plant Emulator Model 220*.

3.3.3. Motion Control Plant ECP Model 220

The *ECP Industrial Plant Emulator Model 220* provides an experimental testbed [ECP]. The system consists of two load plates actuated by two brushless direct current drive motors. Transmission belts couple the load plates and motors according to Figure 3.2. Encoders are attached to the axes of the load plates which provide angular positions and whose associated velocities are estimated with a digital signal processor.
The experimental setup considers both motor currents as controllable plant input $u(t) := \big(u_1(t), u_2(t)\big)^\intercal$ with $u_1, u_2 : \mathbb{R} \mapsto \mathbb{R}$ and hence the setup serves as multiple input system in contrast to the other (simulative) benchmark systems which are single input systems. A mathematical model is defined and identified as described in Appendix F.1 using global nonlinear optimization. Therefore, the units are omitted in the following. Transmission belts are considered as stiff and hence only the angular position and velocity at encoder 2 are treated as (independent) state variable (the gear ratio is 4). The underlying model structure is motivated by the mechanical equilibrium taking the combined moment of inertia, viscous friction, Coulomb friction and the superposition of both actuation torques into account. To this end, the state vector is defined as $x(t) := \big(x_1(t), x_2(t)\big)^\intercal$ with $x_1 : \mathbb{R} \mapsto \mathbb{R}$ denoting the position and $x_2 : \mathbb{R} \mapsto \mathbb{R}$ the velocity respectively. The resulting state space is $\mathcal{X} = \mathbb{R}^2$, the control space $\mathcal{U} = \mathbb{R}$ and the system dynamics model is given as follows:

$$\dot{x}(t) = f\big(x(t), u(t)\big) = \begin{pmatrix} x_2(t) \\ -c_1 x_2(t) - c_2 \tanh\big(c_3 x_2(t)\big) + k_1 u_1(t) - k_2 u_2(t) \end{pmatrix} \tag{3.3.7}$$

with $k_1 = 34.51$, $k_2 = 34.13$, $c_1 = 1.46$, $c_2 = 2.53$ and $c_3 = 5$. The output equation is omitted as the full state is available for state feedback control. Note, $\tanh(\cdot)$ is chosen in favor of the actual sign function in order to account for Assumption 3.1.1 and the slope is set to $c_3 = 5$ to facilitate its later application for smooth online optimization. Furthermore, constraint sets are defined as $\mathbb{X} = \{(x_1, x_2)^\intercal \in \mathcal{X} \mid |x_2| \leq 5\}$ and $\mathbb{U} = \{u \in \mathcal{U} \mid |u| \leq 0.5\}$ respectively.

4

Global Uniform Grid for Time-Optimal Control

This chapter presents a direct approach to time-optimal control with a uniform discretization grid defined by *a single temporal parameter* subject to transformation. Parts of this chapter have been published in [Rös+14a; Rös+15c; Rös+17g].

4.1. Direct Transcription Methods

Direct methods discretize the control and state trajectories according to a specified grid. In contrast to conventional approaches, the discretization interval length itself becomes a decision parameter subject to optimization. Three direct method types are proposed for time-optimal control problems: multiple shooting, collocation via finite differences and Hermite-Simpson collocation.
The control trajectory $u(t)$ with $t \in [t_0, t_\mathrm{f}]$ is discretized along a grid with $N \in \mathbb{N}$ partitions of length $\Delta t \in \mathbb{R}_0^+$:

$$\begin{aligned} t_0 \leq t_0 + \Delta t \quad &= t_1, \\ t_1 \leq t_1 + \Delta t \quad &= t_2, \\ \vdots \quad & \\ t_{N-1} \leq t_{N-1} + \Delta t \quad &= t_N = t_\mathrm{f}. \end{aligned} \tag{4.1.1}$$

Note, the grid implies $t_\mathrm{f} = t_0 + N\Delta t$ and $k = 0, 1, \ldots, N$ denotes the index for individual grid points $t_k = t_0 + k\Delta t$.

4.1.1. Collocation via Finite Differences

Collocation methods discretize both the state and control trajectory on the same grid. Basis functions approximate the system dynamics equation $\dot{x}_u(t) = f\big(x_u(t), u(t)\big)$ and control trajectory $u(t)$ such that the system dynamics are satisfied at grid points t_k for $k = 0, 1, \ldots, N$. The optimal control literature contains few approaches which represent the system dynamics with finite differences. The majority of approaches refer to multiple shooting and collocation via quadrature.

The control trajectory $u(t)$ is assumed to be piecewise constant with respect to the grid partitions. In particular, it is

$$u(t) := u_k = \text{constant} \quad \text{for} \quad t \in [t_k, t_k + \Delta t] \quad \text{and} \quad k = 0, 1, \ldots, N-1. \tag{4.1.2}$$

The states at the grid points t_k are denoted as:

$$x_u(t_k) := x_k \quad \text{for} \quad k = 0, 1, \ldots, N. \tag{4.1.3}$$

The system dynamics equation, in particular $\dot{x}_u(t)$, is approximated on the k-th grid interval by a finite difference kernel $\phi_{\text{fd}}(x_k, x_{k+1}, u_k)$:

$$\dot{x}_u(t_k) \approx \frac{x_{k+1} - x_k}{\Delta t} = \phi_{\text{fd}}(x_k, x_{k+1}, u_k). \tag{4.1.4}$$

Recommended choices for the kernel are:

- Forward differences:
$$\phi_{\text{fd}}(x_k, x_{k+1}, u_k) := f(x_k, u_k). \tag{4.1.5}$$
- Midpoint differences:
$$\phi_{\text{fd}}(x_k, x_{k+1}, u_k) := f\Big(\frac{x_{k+1} + x_k}{2}, u_k\Big). \tag{4.1.6}$$
- Crank-Nicolson differences:
$$\phi_{\text{fd}}(x_k, x_{k+1}, u_k) := \frac{f(x_{k+1}, u_k) + f(x_k, u_k)}{2}. \tag{4.1.7}$$

Whereas forward differences are the most straightforward implementation, it is well known, that implicit kernels that in addition refer to the subsequent state $x_u(k+1)$ are not only more accurate but also numerically more stable. Hence Crank-Nicolson differences evaluate the system dynamics equation twice. In case of computationally complex function evaluations, midpoint differences exhibit a reasonable trade off between computation time and accuracy.
By considering $t_{\text{f}} = t_0 + N\Delta t$, (4.1.2), (4.1.3) and (4.1.4), the resulting nonlinear program is defined as:

$$\min_{\substack{u_0, u_1, \ldots, u_{N-1}, \\ x_0, x_1, \ldots, x_N, \\ \Delta t}} N\Delta t \tag{4.1.8}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_{\text{f}}, \\
&x_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad \Delta t \geq \Delta t_{\min}, \\
&\frac{x_{k+1} - x_k}{\Delta t} = \phi_{\text{fd}}(x_k, x_{k+1}, u_k), \\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

The lower bound on the discretization width $\Delta t_{\min} \in \mathbb{R}^+$ must be strictly positive.

4.1.2. Multiple Shooting

From a theoretical point of view, shooting methods retrieve the exact state trajectory $x_u(t)$ with respect to a discretized control sequence by solving the initial value problem (3.1.2) for the system dynamics equation on the interval $t \in [t_0, t_f]$. The key idea of multiple shooting is to partition the initial value problem on the interval $[t_0, t_f]$ into multiple initial value problems to be solved in isolation. Connectivity and compliance among individual solutions are enforced by additional equality constraints in the nonlinear program.
The control trajectory $u(t)$ is defined as piecewise constant according to (4.1.2). Multiple shooting discretizes the state trajectory either on the same or on a grid with sparser granulation:

$$t_0 = \tilde{t}_0 \leq \tilde{t}_1 \leq \tilde{t}_2 \leq \ldots \leq \tilde{t}_M = t_N. \tag{4.1.9}$$

Herby, $M \leq N$ must hold and for the sake of simplicity, it is assumed that intermediate state grid points $\tilde{t}_i$ for $i = 1, 2, \ldots, M-1$ coincide with control grid points t_k, in particular $\tilde{t}_i \in \{t_1, t_2, \ldots, t_{N-1}\}$. Consequently, the sets $\mathcal{S}_i = \{k \in \mathbb{N}_0 \mid \tilde{t}_i \leq t_k < \tilde{t}_{i+1}, k \leq N\}$ contain all indices k that correspond to shooting interval $[\tilde{t}_i, \tilde{t}_{i+1}]$.
The states at grid points $\tilde{t}_i$ are referred to as shooting nodes s_i:

$$x_u(\tilde{t}_i) := s_i \quad \text{for} \quad i = 0, 1, \ldots, M. \tag{4.1.10}$$

The original initial value problem on the interval $[t_0, t_f]$ is transformed into M shooting intervals. For every shooting interval $[\tilde{t}_i, \tilde{t}_{i+1}]$, $i = 0, 1, \ldots, M-1$, shooting node s_i constitutes the initial state for the underlying initial value problem and let $u(t + \tilde{t}_i) := \tilde{u}_i(t)$ with $t \in [0, \tilde{t}_{i+1} - \tilde{t}_i]$ define the corresponding part of the control trajectory. Hence, function $\tilde{u}_i(t)$ depends on $\tilde{m}_i$ consecutive constant controls u_k with $k \in \mathcal{S}_i$ and cardinality $\tilde{m}_i := |\mathcal{S}_i|$. Obviously, for $N = M$ it is $\tilde{u}_i(t) := u_k$ with $m_{s,i} = 1$ such that both grids are identical. With $\tilde{t}_{i+1} - \tilde{t}_i = \tilde{m}_i \Delta t$, individual solutions to the initial value problem $x_u(\tilde{t}_{i+1})$ are obtained by evaluating (3.1.2):

$$x_u(\tilde{t}_{i+1}) = \varphi\big(\tilde{m}_i \Delta t, \tilde{t}_i, s_i, \tilde{u}_i(t)\big) \quad \text{with} \quad t \in [\tilde{t}_i, \tilde{t}_{i+1}]. \tag{4.1.11}$$

Continuity among shooting intervals is ensured only if the so-called deflection constraints $s_{i+1} = x_u(\tilde{t}_{i+1})$ hold for all i. These constraints constitute the fundamental component in the following nonlinear program:

$$\min_{\substack{u_0, u_1, \ldots, u_{N-1}, \\ s_0, s_1, \ldots, s_M, \\ \Delta t}} N \Delta t \tag{4.1.12}$$

subject to

$$\begin{aligned}
& s_0 = x_\mu(t_\mu), \quad s_M \in \mathbb{X}_f, \quad \Delta t \geq \Delta t_{\min}, \\
& s_i \in \mathbb{X}, \quad u_k \in \mathbb{U} \text{ for all } k = 0, 1, \ldots, N-1, \\
& s_{i+1} = \varphi\big(\tilde{m}_i \Delta t, s_i, \tilde{u}_i(t)\big), \\
& \varphi\big(\varpi \Delta t, s_i, \tilde{u}_i(t)\big) \in \mathbb{X} \text{ for all } \varpi = 1, 2, \ldots, \tilde{m}_i - 1, \\
& i = 0, 1, \ldots, M-1.
\end{aligned}$$

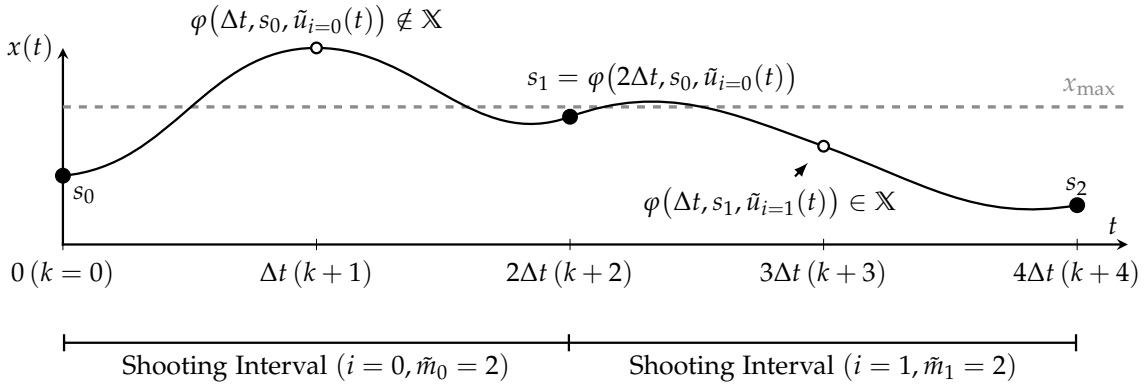


Figure 4.1.: Example state trajectory for multiple shooting with $N = 4$ and $M = 2$. The intermediate state at $k = 1$ is not subject to constraints. Deflection constraints are satisfied.

Notice the additional albeit optional state constraints that depend on the solution to the initial value problems. State constraints are not only imposed on the shooting grid but also for intermediate time instances of the control grid (4.1.1) which is especially crucial if $M \ll N$. Keeping those constraints also implies that the individual choice of M does not affect the optimal solution itself but only the problem structure. Figure 4.1 depicts an already converged example state trajectory for $N = 4$ and $M = 2$ and highlights the effect of ignoring state constraints for the intermediate state at $k + 1$. On the other hand, the second shooting interval shows that even if intermediate states are subject to constraints, the state trajectory violates the constraint. However, this is related to all direct methods and small violations are tolerated as they are controlled by a proper choice of N. For efficiency reasons, any practicable (single-threaded) implementation should solve the initial value problem for a each shooting interval once every time functions and constraints are evaluated such that intermediate states are cached.
The initial value problems $\varphi(\cdot)$ are solved by explicit integration schemes since initial intermediate states in each shooting interval are unknown and not subject to optimization. On the other hand, implicit integration schemes require either internal root finding strategies, for example, an iterative Newton-based algorithm, or they operate with intermediate states and possibly controls as additional optimization parameters. The latter becomes equivalent to direct collocation in which intermediate states (referred to as collocation points) are considered explicitly. A common case is discussed in the next Section 4.1.3. Appendix A.3 provides an overview of explicit integration schemes. In the remainder, multiple shooting is combined with either forward Euler or the explicit 5^{th}-order Runge-Kutta scheme.

4.1.3. Hermite-Simpson Collocation

Collocation via numerical quadrature approximates the integral form of the system dynamics with respect to grid (4.1.1)

$$\int_{t_k}^{t_k+\Delta t} \dot{x}_u(t)\,\mathrm{d}t = x_u(t_k+\Delta t) - x_u(t_k) = x_{k+1} - x_k = \int_{t_k}^{t_k+\Delta t} f\big(x_u(t), u(t)\big)\,\mathrm{d}t \quad (4.1.13)$$

and the control trajectory by basis functions. Quadrature approximates the integral based on multiple (future) knot points also known as implicit numerical integration schemes. Common basis functions are implicit Runge-Kutta functions, for example, Legendre polynomials, midpoint rule, trapezoidal rule or the Simpson rule.
In Hermite-Simpson collocation, the Simpson quadrature rule approximates the integrand, particularly the system dynamics equation $f\big(x_u(t), u(t)\big)$ for $t \in [t_k, t_{k+1}]$ by a quadratic polynomial [Kel17]:

$$\begin{aligned}
&\int_{t_k}^{t_k+\Delta t} f\big(x_u(t), u(t)\big)\,\mathrm{d}t \\
&\approx \phi_{\mathrm{hs}}(x_k, x_{k+0.5}, x_{k+1}, u_k, u_{k+0.5}, u_{k+1}, \Delta t) \\
&= \frac{1}{6}\Delta t\big(f(x_k, u_k) + 4f(x_{k+0.5}, u_{k+0.5}) + f(x_{k+1}, u_{k+1})\big).
\end{aligned} \tag{4.1.14}$$

State and controls at midpoints $t_{k+0.5} := 0.5(t_{k+1} + t_k)$ of the k-th grid partition are denoted as $x_{k+0.5} := x_u(t_{k+0.5})$ and $u_{k+0.5} := u(t_{k+0.5})$ respectively. States x_k and x_{k+1} as well as controls u_k and u_{k+1} coincide with grid points t_k but $x_{k+0.5}$ is not known in advance. Fortunately, $x_{k+0.5}$ is computed from a quadratic interpolant by evaluating the states and function values at grid points k and $k+1$:

$$x_{k+0.5} := h_{\mathrm{hs}}(x_k, x_{k+1}, u_k, u_{k+1}, \Delta t) = \frac{1}{2}(x_k + x_{k+1}) + \frac{\Delta t}{8}\big(f(x_k, u_k) - f(x_{k+1}, u_{k+1})\big). \tag{4.1.15}$$

The actual derivation is provided in [Kel17]. Equation (4.1.15) becomes a separate equality constraint to the nonlinear program which is referred to as uncompressed form. Otherwise $x_{k+0.5}$ is replaced directly in (4.1.14). The latter is denoted as compressed form. Choosing the compressed or uncompressed form has no influence on the actual solution but only the number of parameters to be optimized.
Note, the previous derivation does not consider any particular choice for the control trajectory $u(t)$ between grid points t_k and t_{k+1} but only the presence of $u_{k+0.5}$. Consider the following cases:

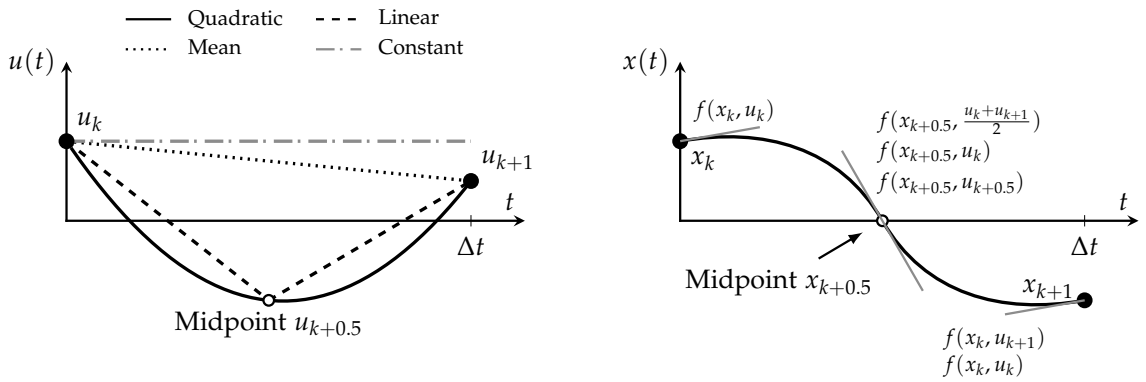
- The *quadratic control spline* $u(t)$ follows from quadratic polynomials for each grid partition $t \in [t_k, t_{k+1}]$:

$$\begin{aligned}
u(t) &:= u_k + \beta_1(t - t_k) + \beta_2(t - t_k)^2, \\
\beta_1 &= -\frac{1}{\Delta t}(3u_k - 4u_{k+0.5} + u_{k+1}), \\
\beta_2 &= \frac{2}{\Delta t^2}(u_k - 2u_{k+0.5} + u_{k+1}).
\end{aligned} \tag{4.1.16}$$

  Note, (4.1.16) is not subject to optimization, but constructs the control trajectory afterwards.

- The *linear control spline* is defined by linear segments between u_k and $u_{k+0.5}$ as well as $u_{k+0.5}$ and u_{k+1}.

- Omitting $u_{k+0.5}$ as additional optimization parameter and substituting the midpoint control by $u_{k+0.5} = 0.5\big(u_k + u_{k+1}\big)$ results in the *(linear) mean control spline*.



(a) Control representations

(b) Cubic Hermite state spline

Figure 4.2.: State and control representations of Hermite-Simpson collocation.

- The *piecewise constant control trajectory* omits both the midpoints $u_{k+0.5}$ and the final control u_N.

Figure 4.2a illustrates the different representations and their associated optimization parameters. Only the quadratic and linear spline consider $u_{k+0.5}$ as explicit optimization parameter.
For brevity, only the nonlinear program of the uncompressed form with dedicated $u_{k+0.5}$ as optimization parameters is presented here. Appendix C.2 details the other formulations.

$$\min_{\substack{u_0,u_{0.5},u_1,\ldots,u_N,\\ x_0,x_{0.5},x_1,\ldots,x_N,\\ \Delta t}} N\Delta t \tag{4.1.17}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_\mathrm{f}, \quad u_N \in \mathbb{U},\\
&x_k \in \mathbb{X}, \quad x_{k+0.5} \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad u_{k+0.5} \in \mathbb{U}, \quad \Delta t \geq \Delta t_\mathrm{min},\\
&x_{k+1} - x_k = \phi_\mathrm{hs}(x_k, x_{k+0.5}, x_{k+1}, u_k, u_{k+0.5}, u_{k+1}, \Delta t),\\
&x_{k+0.5} = h_\mathrm{hs}(x_k, x_{k+1}, u_k, u_{k+1}, \Delta t),\\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

Since the system dynamics equation is approximated with a quadratic polynomial, the corresponding state trajectory $x_u(t)$, in particular the integral of the system dynamics, is constructed by a cubic polynomial for $t \in [t_k, t_{k+1}]$:

$$\begin{aligned}
x_u(t) &:= x_k + \gamma_1(t - t_k) + \gamma_2(t - t_k)^2 + \gamma_3(t - t_k)^3, \qquad (4.1.18)\\
\gamma_1 &= f(x_k, u_k),\\
\gamma_2 &= -\frac{1}{2\Delta t}\big(3\gamma_1 - 4f(x_{k+0.5}, u_{k+0.5}) + f(x_{k+1}, u_{k+1})\big),\\
\gamma_3 &= \frac{2}{3\Delta t^2}\big(\gamma_1 - 2f(x_{k+0.5}, u_{k+0.5}) + f(x_{k+1}, u_{k+1})\big).
\end{aligned}$$

The whole state trajectory is referred to as cubic Hermite spline. Figure 4.2b shows an example of the cubic polynomial for a single interval Δt. At grid and midpoints, tangent lines emphasize the related values of the system model depending on the selected control representation. Note, the solution to the nonlinear program for both the quadratic and the linear control spline are identical as the actual interpolation is not part of (4.1.17). Due to reduced degrees of freedom, the mean and constant control spline result in different state trajectories even though it is not visualized in Figure 4.2b for better readability.

Remark 4.1.1. Multiple shooting with $M = N$ and both direct collocation types are fairly similar to each other under certain conditions. As discussed before, shooting and quadrature-based collocation methods are usually distinguished by means of explicit and implicit integration schemes. For piecewise constant controls, finite difference equations are transformed into a similar integral form as the following example illustrates for forward differences:

$$\frac{x_{k+1} - x_k}{\Delta t} = f(x_k, u_k) \tag{4.1.19}$$

$$\Leftrightarrow \qquad x_{k+1} = x_k + \Delta t f(x_k, u_k). \tag{4.1.20}$$

The equivalence of (4.1.19) and (4.1.20) holds for $\Delta t \neq 0$. Note, expression (4.1.20) is the forward Euler numerical integration scheme. A similar relation holds for midpoint differences and rectangle quadrature as well as for Crank-Nicolson differences and trapezoidal quadrature. However, as shown in Chapter 8, using the finite difference version with Δt in the denominator reveals different convergence rates in comparison to the integral form.

4.2. Solution to the Nonlinear Program

The proposed nonlinear programs (4.1.8), (4.1.12) and (4.1.17) approximate the continuous-time time-optimal control problem (3.2.2) with N partitions of length Δt each. The continuous-time control trajectory is replaced by either piecewise constant controls or by splines. To conclude, the grid size N is crucial for the accuracy and feasibility of the approximation:

- The number of grid partitions is equivalent to the degrees of freedom in terms of control interventions (number of different u_k).
- The system dynamics equation is approximated numerically on every grid partition. Hence a coarse grid in combination with less accurate integration respectively differentiation schemes can violate the sampling theorem such that fast dynamics are not captured adequately.
- Obviously, $N > 0$ must hold to define a proper nonlinear program.

Even though these observations apply to direct methods in general, the variable discretization grid and the fixed final state aggravate a proper initial guess of the grid

size N. The following viability assumption addresses the first point. According to Section 3.2.2, viability implies feasibility and needs to be assumed for any particular application:

Assumption 4.2.1 (Conditional Viability)**.** Viability according to Definition 3.2.1 is assumed to be ensured not only for arbitrary $u(t)$ with $t_0 \leq t \leq t_\mathrm{f}$, but also for the limited control function space given by the chosen transcription method and grid size N, in particular $u \in \mathscr{U}^N(t_\mathrm{f})$.

Appendix A.2 provides a formal definition of function space $\mathscr{U}^N(t_\mathrm{f})$. The following assumption is stronger and also addresses the system dynamics approximation. It is crucial for any closed-loop application:

Assumption 4.2.2 (Optimal Solution and System Dynamics Accuracy)**.** For a given $x_\mathrm{s} \in \mathbb{X}$ respectively $x_\mu(t_{\mu,0})$ and $\mathbb{X}_\mathrm{f} \neq \emptyset$, there exists a finite $N > 0$ for which nonlinear programs (4.1.8), (4.1.12) or (4.1.17) are feasible and their solution constitutes a unique minimizer such that necessary and sufficient conditions hold. Furthermore, the grid size N is assumed to be large enough such that the system dynamics error for the chosen transcription method is negligible. For theoretical purposes, it is further assumed that the plant and system dynamics model have zero model mismatch.

Necessary and sufficient optimality conditions are summarized in Appendix A.4. Nonlinear programs (4.1.8), (4.1.12) or (4.1.17) are solved by any suitable nonlinear program solver. In any practical implementation, compact and convex constraint sets $\mathbb{X}, \mathbb{U}$ and $\mathbb{X}_\mathrm{f}$ are replaced by algebraic equality and inequality constraint functions which is straightforward for simple box constraints. However, the nonlinear program definitions throughout this thesis retain those sets whenever the actual functions are not explicitly required due to the compactness and clarity of notion. All nonlinear programs are solved by either the well established nonlinear program solver IPOPT (*Interior Point Optimizer*, see Appendix E.2.1) or a dedicated sequential quadratic programming (SQP) implementation as summarized in Appendix E.2.2.

Remark 4.2.1. Numerical simulations indicate that $\Delta t_\mathrm{min} > 0$ is crucial for IPOPT and the SQP implementation. Even though the finite differences approach (4.1.8) is not defined for $\Delta t = 0$, setting $\Delta t_\mathrm{min} = 0$ still results in proper solutions. The underlying line-search inherently rejects steps resulting in $\Delta t = 0$ due to the growing and unbounded equality constraint. On the other hand, the multiple shooting nonlinear program is defined for $\Delta t = 0$ but the solvers occasionally get stuck at infeasible points. Nevertheless, choosing any small $\Delta t_\mathrm{min} > 0$ is accomplished easily. To this end, $\Delta t_\mathrm{min} > 0$ is included in the stability theorems in Section 5.1.

Remark 4.2.2. Many optimization algorithms operate with the Hessian of the Lagrangian [NW06] which is often required to be positive (semi-) definite in every solver iteration. Since this is not always the case, those solvers implement a so-called inertia correction in which based on some heuristics the Hessian is modified until it becomes positive definite [NW06; Bet10]. Whereas quadratic form cost functions are inherently positive definite, the Hessian of the Lagrangian is often positive definite as well or can easily be fixed. However, in the time-optimal case, the Hessian of the

cost function is zero. By assuming only box constraints as inequalities, the Hessian of the Lagrangian consists only of the scaled Hessian of the system dynamics equality constraint. Now consider for example a linear system $\dot{x}_u(t) = Ax_u(t) + Bu(t)$ with $A, B, x_u(t), u(t) \in \mathbb{R}$ and $N = 1$. In case of the forward differences approximation according to (4.1.12) and (4.1.5), the corresponding equality constraint is $h_{\text{fd}}(u_0, \Delta t) = (x_1 - x_0)/\Delta t - Ax_0 - Bu_0 = 0$. Parameters x_0 and x_1 are fixed for $N = 1$ and hence not subject to optimization. As a result, the Hessian of $h_{\text{fd}}(\cdot)$ is given by:

$$\nabla^2 h_{\text{fd}}(u_0, \Delta t) = \begin{pmatrix} 0 & 0 \\ 0 & 2\Delta t^{-3}(x_1 - x_0) \end{pmatrix}. \tag{4.2.1}$$

The corresponding eigenvalues are 0 and $2\Delta t^{-3}(x_1 - x_0)$ such that depending on $x_1 - x_0$ the Hessian is either positive or negative semidefinite (it is $\Delta t > 0$ by definition). As a second example, consider the very same system dynamics but with multiple shooting (4.1.12) (forward Euler) and $N = M = 1$. The equality constraint is $h_{\text{fe}}(u_0, \Delta t) = (x_1 - x_0) - \Delta t(Ax_0 + Bu_0) = 0$ with leads to the following Hessian:

$$\nabla^2 h_{\text{fe}}(u_0, \Delta t) = \begin{pmatrix} 0 & -B \\ -B & 0 \end{pmatrix}. \tag{4.2.2}$$

In this example, the eigenvalues are $\pm B$ such that the Hessian is indefinite for any $B \neq 0$. Generally, a common remedy is to utilize the Broyden-Fletcher–Goldfarb-Shanno (BFGS) algorithm with additional positive definiteness enforcement to approximate second-order derivatives. On the other hand, IPOPT handles the nonlinear programs well with explicit Hessian computation. The dedicated SQP implementation ensures a positive semidefinite Hessian by squaring Δt in the cost function and replaces the Hessian of the Lagrangian by the Hessian of the new squared cost function. Appendix E.2.2 details the implementation.

4.3. Examples

This section presents two examples concerning time-optimal control of the Van der Pol oscillator.

Example 4.3.1 (Open-loop Control of the Van der Pol Oscillator)**.** Consider the Van der Pol oscillator with the origin as initial state, $x_{\text{f}} = (0.8, 0)^\intercal$ and control bounds $\mathbb{U} = \{u \in \mathbb{R} \mid |u| \leq 1\}$. Figure 4.3 shows the open-loop solutions for several transcription methods with $N = 15$. As reference, the time-optimal trajectory is obtained by a two-point boundary value problem [AK04]. Time-optimality is accomplished by a time transformation and the overall boundary value problem is solved with Matlab's *bvp4c* algorithm.
Inspection of the control trajectory in Figure 4.3 reveals that the major deviations occur at the control switching point ± 1. The ideal switching point does not coincide exactly with the grid for $N = 15$ such that the transition is realized over two consecutive intervals. Notice, the Hermite-Simpson method with quadratic control splines based

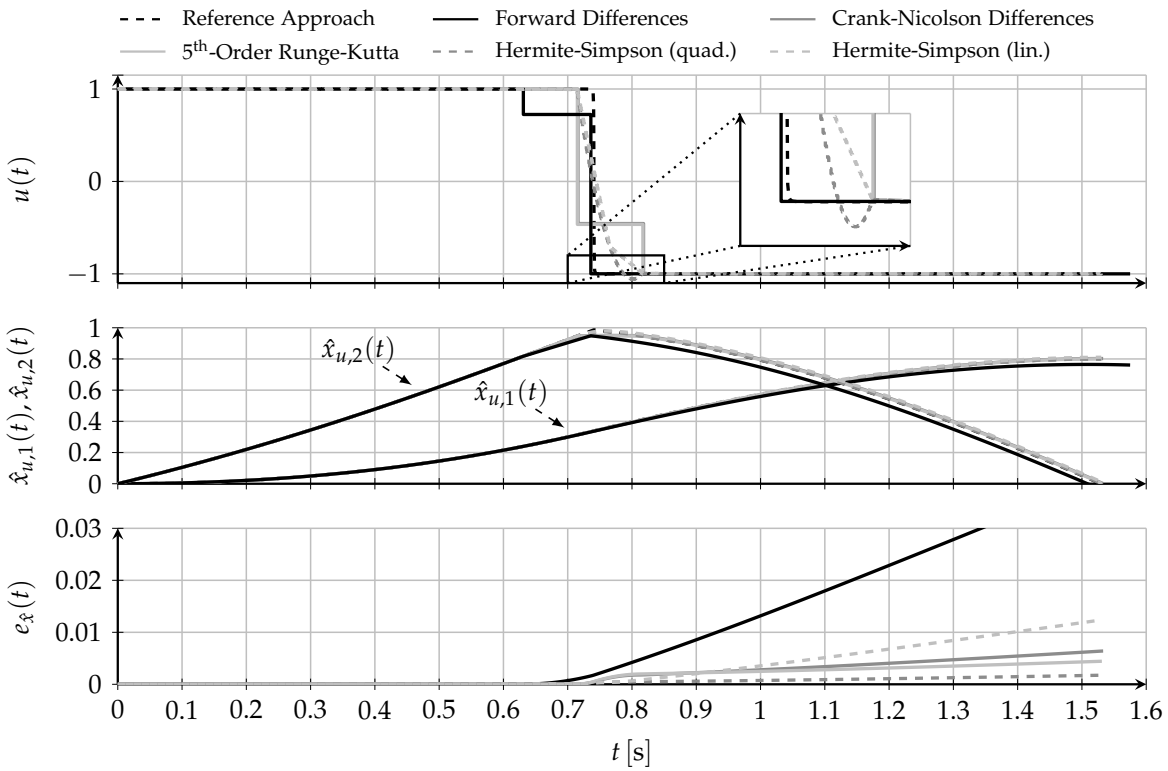


Figure 4.3.: Open-loop solutions for the Van der Pol oscillator with $x_\mathrm{f} = (0.8, 0)^\intercal$ and $N = 15$. Several direct transcription methods are compared. Multiple shooting is performed with the Runge-Kutta scheme. Hermite-Simpson collocation considers either linear or quadratic control splines. While the controls $u(t)$ are obtained from the nonlinear programs, the state trajectories are precisely simulated with $u(t)$. The bottom plot shows the error with respect to the reference solution.

on (4.1.16) violates the control bounds since constraints are only enforced at grid points and collocation midpoints.
In order to visualize and evaluate the transcription errors with respect to the reference solution, the actual open-loop trajectory $\hat{x}_u(t)$ is computed according to (3.2.4) as well as the evolution of the integral of the ℓ_2-norm error $e_{\hat{x}}(t)$ using (3.2.5). The error is zero for the first control arc since all methods share the same upper maximum control signal. However, beyond the switching point, the error evolution differs. Forward Euler exhibits the largest error which is also the case for forward differences and due to the coarse system discretization t_f is much larger. Due to the equivalence in the optimum according to (4.1.19) and (4.1.20) only forward Euler is presented. The implicit Hermite-Simpson method with quadratic control splines exhibits the smallest error as it matches the original switching point best but violates the lower control limit.

Example 4.3.2 (Hermite-Simpson Constraint Violation)**.** Consider the Van der Pol oscillator as before but with an additional bound on the second state, such that $\mathbb{X} = \{x \in \mathbb{R}^2 \mid |(0,1)x| \leq 0.7\}$ holds. Figure 4.4 shows the optimal state and control trajectories for the Hermite-Simpson method with quadratic control splines and varying grid sizes N. The additional state constraint results in a *non-bang-singular-bang*

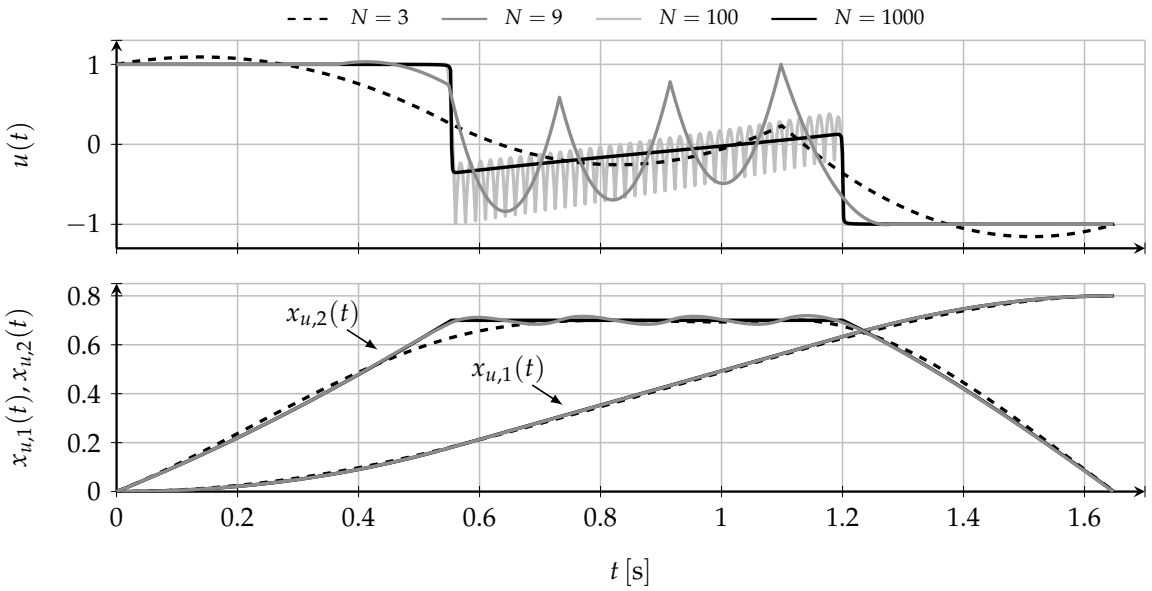


Figure 4.4.: Solutions obtained from Hermite-Simpson collocation with quadratic control splines for the Van der Pol oscillator with $x_\mathrm{f} = (0.8, 0)^\intercal$, an additional state constraint and varying N. The additional degree of control freedom at midpoints allows the cubic state trajectory to violate state constraints between the grid and midpoints to minimize time.

control type as the control is non-constant at the active bound. The solution for $N = 3$ already provides an accurate approximation of the state trajectory which is represented in terms of a cubic spline. On the other hand, unlike for quadratic form cost functions with smoother control trajectories, introducing $u_{k+0.5}$ for linear or quadratic control splines allows the time-optimal cubic state trajectory to violate the bound in between x_k and $x_{k+0.5}$ as well as in between $x_{k+0.5}$ and x_{k+1}. Obviously, this can similarly happen to quadratic form costs, but the penalty on the control usually avoids such constraint violations. Even though this is the optimal solution for the nonlinear program, this might be counter-intuitive at the design stage as commonly better results are expected for high-order control and state trajectories. For a large grid size, $N = 1000$, the solution becomes at least visibly oscillation-free. A possible remedy is to add further constraint evaluations to the nonlinear program. However, this results in higher computation times such that a more suitable way is to tackle the problem by reducing the degrees of freedom in the control, for example by using constant controls or by choosing $u_{k+0.5}$ as the mean control. In order to avoid repetitions, the corresponding example is provided in the chapter of the quasi-uniform grid. Note that the oscillation effect in minimum-time Hermite-Simpson collocation with $u_{k+0.5}$ as optimization parameter is not limited to the systems in this thesis. It is attributed to the cubic interpolation of the state trajectory and even occurs with double integrator systems.

5

Time-Optimal Model Predictive Control

This chapter integrates the previously defined direct optimal control problems (4.1.8), (4.1.12) or (4.1.17) with state feedback to form a time-optimal MPC scheme as depicted in Figure 1.2. Parts of this chapter have been published in [Rös+14a; Rös+15c; Rös+17g].

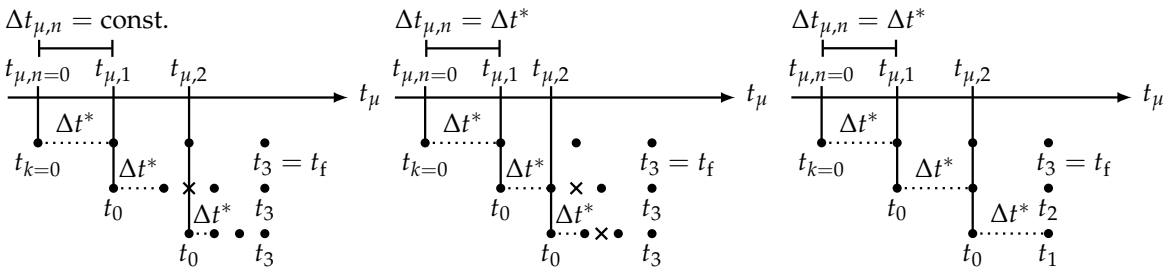
5.1. Shrinking-Horizon Closed-Loop Control

The nominal closed-loop control system with control law $\mu\big(x_\mu(t_\mu)\big)$, sampling grid $t_{\mu,n}, n \in \mathbb{N}_0$, and initial state $x_\mathrm{s} \in \mathbb{X}$ has been defined in Section 3.2.3. Let $u^*(t)$ denote the optimal admissible control trajectory from either (4.1.8), (4.1.12) or (4.1.17). Note, $u^*(t)$ consists of either piecewise constant or piecewise polynomial functions depending on the choice of the transcription method. The resulting predictive control law is now defined as follows:

$$\mu\big(x_\mu(t_\mu)\big) := u^*(t_\mu - t_{\mu,n}; t_{\mu,n}) \quad \text{for} \quad t_\mu \in [t_{\mu,n}, t_{\mu,n+1}). \tag{5.1.1}$$

For clarity, the notation $u^*(\cdot; t_{\mu,n})$ explicitly includes $t_{\mu,n}$ in order to indicate its relation to the closed-loop time instance $t_{\mu,n}$. Closed-loop sampling interval lengths $\Delta t_{\mu,n} = t_{\mu,n+1} - t_{\mu,n}$ are either set to a constant value for all $n \in \mathbb{N}_0$ as shown in Figure 5.1a (synchronous sampling) or they are inherited from the individual solutions Δt^* at time instances $t_{\mu,n}$ as depicted in Figure 5.1b (asynchronous sampling).
The remainder of this section addresses stability properties of the closed-loop system with (5.1.1) and steady state $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$. The nonlinear programs in Chapter 4 contain the final state as an equality condition similar to the optimal control problems with time transformation for which stability and recursive feasibility are often *assumed* to be valid. However, the control parameterization for direct transcription invalidates recursive feasibility guarantees. Even these time-optimal control formulations do not necessarily realize a proper stabilization at the steady state. In detail, the reasons are as follows:

- Since the grid is uniform and the grid size N as well as the final state x_f are fixed, Δt^* changes as the closed-loop system evolves. Correspondingly, grid points t_k of the very first nonlinear program do not coincide with the closed-loop sampling



(a) Synchronous sampling (b) Asynchronous sampling (c) Shrinking horizon

Figure 5.1.: Illustration of synchronous, asynchronous and shrinking horizon schemes. The dots correspond to the discretization grid of the prediction (horizon) at closed-loop sampling times $t_{\mu,n}$. The fixed sampling interval in (a) is set to Δt^* of the very first the optimal control problem. Crosses indicate a potential lack of recursive feasibility.

instances $t_{\mu,n}$ for both the synchronous and the asynchronous sampling scheme. Hence, Bellman's principle of optimality does not hold anymore. Figure 5.1a indicates that even if $\Delta t_{\mu,0} = \Delta t^*$ is inherited from the first nonlinear program, the synchronous scheme has a potential lack of recursive feasibility. The prediction at $t_{\mu,1}$ cannot realize a switch in control at t_2 of the previous solution (marked by a cross symbol). Figure 5.1b confirms this observation for the asynchronous sampling scheme. From a practical point of view, the principle of optimality is fulfilled in an approximative manner, especially for N large or $N \to \infty$.

- Consider the case $x_\mu(t_\mu) = x_\mathrm{f}$ for which the system is already at steady state x_f. For $\Delta t_\mathrm{min} = 0$, the optimal grid partition length is $\Delta t^* = 0$ and all system dynamic constraints are trivially satisfied. This in turn allows the optimizer to choose any $u(t) \in \mathbb{U}$ which is verified by $\varphi\big(0, x_\mathrm{f}, u(t)\big)$ in (3.1.2) (under-constrained problem). Alternatively, the special case $\Delta t^* = 0$ must be handled separately when evaluating the sampled control law. For $\Delta t_\mathrm{min} > 0$, $u(t)$ can also be arbitrary as long as the boundary conditions $x_\mu(t_\mu) = x_\mathrm{f}$ are met. Note, intermediate states are not attracted by x_f as in quadratic form MPC which often results in oscillations around the steady state and chattering in the control law while loosing feasibility and asymptotic stability guarantees.

Even though these issues might be acceptable in some applications, the closed-loop system exhibits an additional technical drawback: For $\Delta t_\mathrm{min} \approx 0$, a fixed grid size N implies $\Delta t \to \Delta t_\mathrm{min} \approx 0$ as the state of the system converges towards x_f (compare Figure 5.1a and 5.1b). This often results in numerical issues at the solving stage: The ill-conditioned optimization problem leads to significantly reduced convergence speeds. An example at the end of this section demonstrates this effect.
A first and straightforward approach to tackle those issues is given by the following adaptive shrinking horizon control scheme. The key is to reduce the grid size N while the control system evolves. The overall procedure is described in Algorithm 5.1. The set of all optimization parameters is denoted by $\mathcal{P}_\mathrm{global}$ with $\mathcal{P}_\mathrm{global} := \{u_0, u_1, \ldots, u_{N-1}, x_0, x_1, \ldots, x_N, \Delta t\}$ for finite difference collocation and multiple shooting ($M = N$). In case of uncompressed Hermite-Simpson collocation with quadratic control splines, the

Algorithm 5.1.: Time-optimal shrinking horizon control.

1: **procedure** SOLVEOCP($\mathcal{P}_{\text{global}}, x_\mu(t_{\mu,n}), x_\text{f}, N, N_{\min}$)
2: Initialize or update optimization parameters $\mathcal{P}_{\text{global}}$ ▷ Warm-starting
3: Solve nonlinear program w.r.t. $\mathcal{P}_{\text{global}}$ ▷ see (4.1.8), (4.1.12) or (4.1.17)
4: **if** $N > N_{\min}$ **then**
5: $N \leftarrow N - 1$ ▷ New N only applies at next invocation
6: **return** $\mathcal{P}_{\text{global}}, N$ ▷ Note, the optimized control sequence is included

set is defined as $\mathcal{P}_{\text{global}} := \{u_0, u_{0.5}, u_1, \ldots, u_N, x_0, x_{0.5}, x_1, \ldots, x_N, \Delta t\}$. The inputs of Algorithm 5.1 are an empty or non-empty set $\mathcal{P}_{\text{global}}$, the initial state $x_\text{s} = x_\mu(t_{\mu,n})$, the desired final state x_f, the initial grid size $N = N_{\text{init}}$ and a minimum grid size $N_{\min}$. First assume $N_{\min} = 1$ as the importance of limiting the grid size is discussed later. Note, in case not all components of x_f are fixed (as for the rocket system), these components of x_f serve as initial guess that is later refined by the solver.

Initialization and Update of Optimization Parameters

An initialization of state and control trajectories is necessary for any derivative based nonlinear program solver. In the very first invocation of Algorithm 5.1 the set of optimization parameters $\mathcal{P}_{\text{global}}$ is empty. For the examples and applications in the scope of this thesis, a rather simple initialization strategy is utilized which can be applied to a wide range of small- and midscale optimal control problems. The state trajectory is approximated by linear interpolation between start state $x_\mu(t_{\mu,n})$ and final state x_f with $N = N_{\text{init}}$ grid partitions, in particular

$$x_k = x_\mu(t_{\mu,n}) + \frac{k}{N_{\text{init}}}\big(x_\text{f} - x_\mu(t_{\mu,n})\big) \text{ for } k = 0, 1, \ldots, N_{\text{init}}. \tag{5.1.2}$$

The control sequence is initialized with zero controls $u_k = 0$ for $k = 0, 1, \ldots, N_{\text{init}}$. In case of Hermite-Simpson collocation, initialization applies to $k = 0, 0.5, 1, \ldots, N_{\text{init}}$ respectively. The time resolution is initialized with some expected $\Delta t > \Delta t_{\min}$. Obviously, this initialization strategy does not guarantee global convergence in the general case, since solving nonlinear programs numerically leads to a local minimizer with respect to the initialization $\mathcal{P}_{\text{global}}$.
Grid size N changes in subsequent invocations of Algorithm 5.1 according to line 5. A straightforward implementation erases $\mathcal{P}_{\text{global}}$ and reinvokes (5.1.2) according to the new grid size N. However, warm-starting the optimization problem is crucial for performance reasons [WB08]. Instead of reinitializing $\mathcal{P}_{\text{global}}$ in every sampling interval from scratch, the following update strategy applies:

- In case N remains unchanged, for example if $N = N_{\min}$ is already reached, the previous set is updated by only setting $x_0 = x_\mu(t_{\mu,n})$.
- Otherwise, Algorithm 5.1 reduces the grid size only by one and hence initializing $x_1 = x_\mu(t_{\mu,n})$ and erasing both x_0 and u_0 is sufficient. Notice, without model mismatch, this warm-start already defines the optimal solution as Figure 5.1c demonstrates (principle of optimality).

Grid Size Reduction and Asymptotic Stability Results

For details regarding the solution of the nonlinear programs in line 3 refer to Section 4.2. Afterwards, the grid size N reduces by one in any subsequent invocation. Algorithm 5.1 is invoked at every time instance $t_{\mu,n}$ with $x_\mu(t_{\mu,n})$ as new initial state for the underlying optimal control problem. This in turn ensures recursive feasibility at least until $N_{\min} = 1$ is reached (see Figure 5.1c).
In addition, by choosing piecewise constant controls as control parameterization, the following initial result on asymptotic stability holds:

Proposition 5.1.1. *Consider* (4.1.8), (4.1.12) *with forward/backward Euler or Hermite-Simpson collocation with piecewise constant controls. Furthermore define* $\Delta t_{\min}$ *such that* $0 < \Delta t_{\min} < \Delta t^*$ *is ensured (*Δt^* *is the optimal time interval from the very first optimal control problem). Let* $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$ *represent a steady state such that there exists* $u \in \mathbb{U}$ *with* $f(x_\mathrm{f}, u) = 0$ *and assume that Assumption 3.1.1, constraint qualifications, second-order necessary conditions and for any initial* $x_\mu(t_{\mu,0}) = x_\mathrm{s} \in \mathbb{X}$ *Assumption 4.2.2 hold. The closed-loop feedback is realized with Algorithm 5.1 and* $N_{\min} = 1$ *at sampling times inherited from the open-loop solution, in particular* $t_{\mu,n+1} - t_{\mu,n} = \Delta t^*$*. Then, the closed-loop system is asymptotically stable on* $\mathbb{X}$*.*

The result inherently addresses recursive feasibility and hence ensures forward invariance on $\mathbb{X}$ according to Definition 3.2.4. Appendix B.2 provides the proof and related lemmas. Note, this result already takes $\Delta t_{\min}$ into account according to Remark 4.2.1 and condition $\Delta t_{\min} < \Delta t^*$ is easily satisfied by a proper choice of N.

Example 5.1.1 (Closed-loop Control)**.** This example demonstrates the observations in closed-loop control. The control task consists of guiding the Van der Pol oscillator from the origin to $x_\mathrm{f} = (0.8, 0)^\intercal$ with control bounds $\mathbb{U} = \{u \in \mathbb{R} \mid |u| \leq 1\}$. Figure 5.2 shows the evolution of the controls, the states and the corresponding computation time Δt_cpu. Nonlinear programs obtained from multiple shooting with $M = N$ and $N_\mathrm{init} = 50$ are solved with IPOPT and warm-start is disabled. The closed-loop sampling times are inherited from the open-loop solutions. First consider a case without grid reduction and $\Delta t_{\min} = 0.0001\,\mathrm{s}$. The control trajectory exhibits proper bang-bang characteristics until x_f is reached at $t_\mu \approx 1.56\,\mathrm{s}$. Afterward, the controller does not choose the correct control to keep the system entirely at the steady state which is $\mu(\cdot) = 0.8$ according to Section 3.3.1. Also, the second state $x_{\mu,2}(t_\mu)$ oscillates around zero. The required computation time indicates that IPOPT's convergence rate drops significantly due to the ill-conditioned problem. The case $\Delta t_{\min} = 0.001\,\mathrm{s}$ performs better but close to x_f the trajectory is rather smooth instead of bang-bang. Obviously, the controller already reached $\Delta t^* = \Delta t_{\min}$ and is able to stabilize the system at x_f. Note, the transition close to x_f is just a random albeit feasible solution to the boundary value problem subject to optimization (the cost function already reached a constant value). Setting $\Delta t_{\min} = 0.04\,\mathrm{s}$ enforces an active bound $\Delta t^* = \Delta t_{\min}$ from the very beginning, since $N_\mathrm{init} \cdot 0.04\,\mathrm{s} \geq t_\mathrm{f}$ with $t_\mathrm{f} \approx 1.56\,\mathrm{s}$. The solution is neither time-optimal nor does the system reach the equilibrium x_f for $t_\mu \to \infty$. In comparison, the closed-loop system with grid reduction (shrinking horizon) according to Algorithm 5.1 and Proposition 5.1.1 is able to stabilize the system at x_f while simultaneously decreasing the computation time.

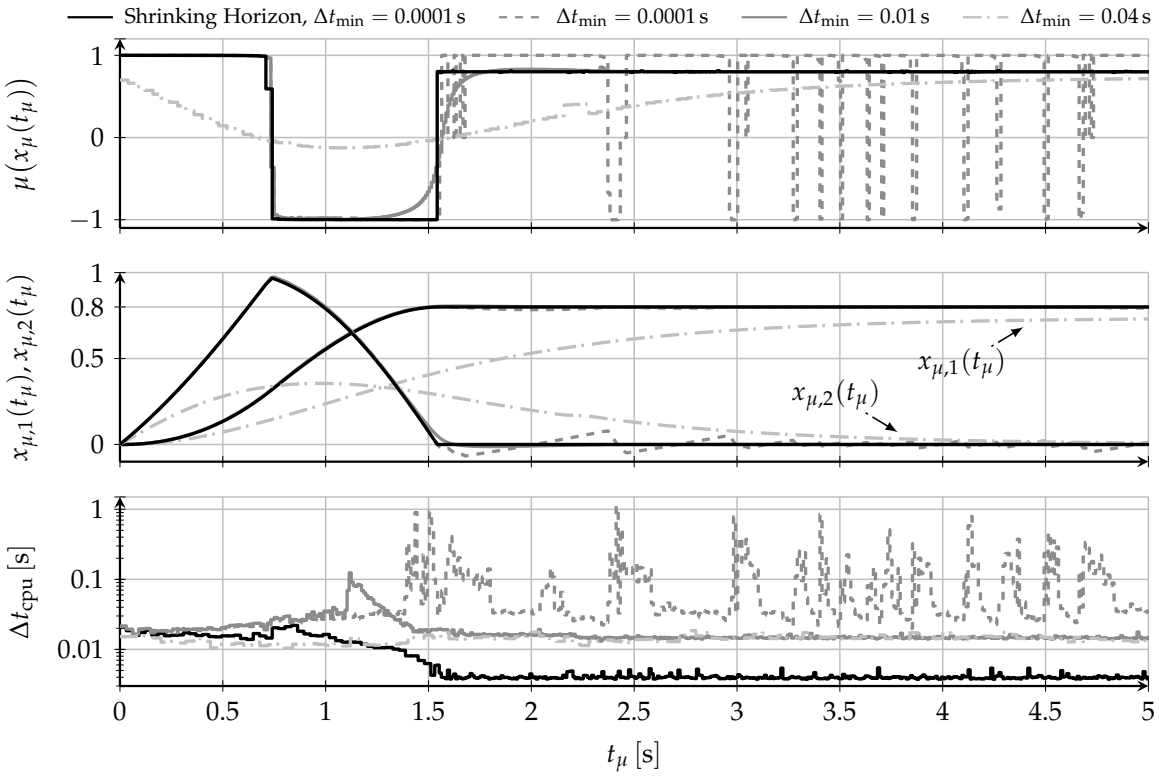


Figure 5.2.: Comparison of closed-loop control realizations for multiple shooting with $M = N$ for the Van der Pol Oscillator. Three approaches in gray perform without grid adaptation but different lower bounds on Δt. The shrinking horizon grid adaptation stabilizes the system properly with $\mu(\cdot) = 0.8$ while maintaining time-optimality and less computational burden.

Proposition 5.1.1 is a rather theoretical result. If the first solution is feasible (which relates to viability), all subsequent solutions are feasible as well for $N \to 1$ and for maintaining the system at steady state x_f. In practice, however, any small disturbance results in a potential loss of viability for small grid sizes. The terminal equality constraint requires the system to perfectly reach x_f with $N_{\min} = 1$ piecewise constant controls. But it is well known that systems often require multiple switches in control to reach x_f, even for initial states close to x_f. For unconstrained linear systems, [Kai80; Van+11b] suggest to choose $N_{\min} \geq p/q$ with state dimension p and control dimension q respectively. Note that this condition is also related to Feldbaum's theorem for single input systems with real eigenvalues which states $N_{\min} \geq p$ [Pon87].
Choosing $N_{\min} > 1$ in Algorithm 5.1 no longer confirms asymptotic stability. As long as $N > N_{\min}$ lasts, the closed-loop evolution coincides with Figure 5.1c and leads to sampling intervals of $\Delta t_{\mu,n} = \Delta t^*$, but once $N = N_{\min}$ is reached, the controller takes another step with Δt^* and then behaves like the asynchronous scheme in Figure 5.1b. Consequently, recursive feasibility and thus asymptotic stability are not guaranteed.

Relaxation to Practical Asymptotic Stability

The previous discussion elucidates that a minimum grid size $N_{\min} > 1$ is crucial to maintain viability. A proper value for $N_{\min}$ depends on the system and constraint sets,

however, some $N_{\min} \geq p$ is a good starting point for simulations and experiments. It is particularly important to still provide stability and feasibility guarantees even for $N_{\min} > 1$. In the following, the stability results are relaxed to ensure convergence at least up to a certain region containing x_f and more general control parameterizations. The practical applications then decide if this particular region is small enough to, for example, realize a proper point-to-point motion. Alternatively, it facilitates the systematic design of a dual-mode controller that restores asymptotic stability.
The following sets play a crucial role for the derivation of practical stability:

Definition 5.1.1 (Controllability Region)**.** Let $\mathscr{U}^N(t_c)$ define the space of all control functions up to time t_c with $u \in \mathscr{U}^N(t_c)$ according to the direct transcription methods, in particular N piecewise polynomial functions (see Appendix A.2 for a formal description). The set which contains all states $\tilde{x} \in P_c(x_\mathrm{f}, N, t_c)$ from that the final state $x_\mathrm{f} \in \mathbb{X}$ is reachable within a transition time of t_c is defined by:

$$P_c(x_\mathrm{f}, N, t_c) := \big\{\tilde{x} \in \mathbb{X} \mid \exists u \in \mathscr{U}^N(t_c),\ \varphi\big(t_c, \tilde{x}, u\big) = x_\mathrm{f},\ \varphi\big(t, \tilde{x}, u\big) \in \mathbb{X},\ u(t) \in \mathbb{U},\ 0 \leq t \leq t_c\big\}. \tag{5.1.3}$$

Note, this set relates to viability on $\big(P_c(x_\mathrm{f}, N, t_c), \mathbb{X}_\mathrm{f}\big)$ up to time t_c. It is further equivalent to the reachable set from x_f in time t_c with N steps and the backward respectively reverse-time system $\dot{x}(t) = -f\big(x(t), u(t)\big)$ [BM15]. Determining $P_c(x_\mathrm{f}, N, t_c)$ analytically is usually difficult and common numerical methods to obtain reachable or controllable sets can be applied [BM15]. For instance simulations can be performed for low-dimensional systems, or an auxiliary optimal control problem can be solved which maximizes the target set (reachable set) according to the backward dynamics [Hor15]. The stability analysis takes $\Delta t_{\min}$ according to Remark 4.2.1 into account. Accordingly, the optimal interval length reduces until $\Delta t^* = \Delta t_{\min}$ is reached and remains constant afterwards which affects closed-loop convergence:

Definition 5.1.2 (Constant Cost Region)**.** For any $\Delta t_{\min} > 0$ and $N \geq 1$, the optimal cost function value $N\Delta t^*$ remains constant in a neighborhood of x_f which is:

$$P_{\Delta t_{\min}}(x_\mathrm{f}, N) := P_c(x_\mathrm{f}, N, N\Delta t_{\min}). \tag{5.1.4}$$

The following relaxed (practical) stability result according to Definition 3.2.5 holds:

Theorem 5.1.1. *Consider nonlinear programs* (4.1.8), (4.1.12) *or* (4.1.17). *Let* $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$ *represent a steady state such that there exists* $u \in \mathbb{U}$ *with* $f\big(x_\mathrm{f}, u\big) = 0$ *and assume that Assumption 3.1.1 and for any initial* $x_\mu(t_{\mu,0}) = x_\mathrm{s} \in \mathbb{X}$ *Assumption 4.2.2 hold. Furthermore, choose* $N_{\min} \geq 1$ *and* $\Delta t_{\min}$ *such that* $0 \leq \Delta t_{\min} < \Delta t^*$ *holds at time instance* $t_{\mu,0}$. *The closed-loop feedback is realized with Algorithm 5.1 at sampling times inherited from the open-loop solution, in particular* $t_{\mu,n+1} - t_{\mu,n} = \Delta t^*$. *Then, the closed-loop system is P-practically asymptotically stable on* $\mathbb{X}$ *with* $P = P_c\big(x_\mathrm{f}, N_{\min}, (N_{\min} - 1)\Delta t^*\big)$.

Condition $\Delta t_{\min} < \Delta t^*$ ensures that (5.1.4) is not active before the systems enters P and applies for proper choices of $\Delta t_{\min}$ and N. Note, the closed-loop system only looses recursive feasibility guarantees in P and then asymptotic stability in $P_{\Delta t_{\min}}(x_\mathrm{f}, N_{\min})$. Refer to Appendix B.2 for the proofs. Practical implications of these results are that by

Algorithm 5.2.: Global uniform grid optimal control with grid adaptation.

1: **procedure** SOLVEOCP($\mathcal{P}_{\text{global}}, x_\mu(t_{\mu,n}), x_\text{f}, \Delta t_\text{ref}, \Delta t_\epsilon, I_\text{adapt}, N, N_\text{min}$)
2: Initialize or update optimization parameters $\mathcal{P}_{\text{global}}$ ▷ Warm-starting
3: **for** $i = 1, 2, \ldots, I_\text{adapt}$ **do**
4: **if** $i > 1$ **or** warm-start **then**
5: Adapt grid $\mathcal{P}_{\text{global}}$ w.r.t. Δt_ref and Δt_ϵ ▷ See (5.2.1) and (5.2.2)
6: Solve nonlinear program w.r.t. $\mathcal{P}_{\text{global}}$ ▷ see (4.1.8), (4.1.12) or (4.1.17)
7: **if** $|\Delta t - \Delta t_\text{ref}| \leq \Delta t_\epsilon$ **and** $N > N_\text{min}$ **then**
8: $N \leftarrow N - 1$ ▷ New N only applies at next invocation
9: **return** $\mathcal{P}_{\text{global}}, N$ ▷ Note, the optimized control sequence is included

increasing the initial N, which in turn reduces Δt^*, or by reducing N_min, the size of region P is reduced. In contrast, certain choices of N and N_min are further implicitly bounded by Δt_min, the dynamics accuracy, viability and the computational resources required, thus limiting the endless decrease of P.

5.2. Grid Size Adaptation

The previous descriptions and results clearly show that reducing the grid while converging towards x_f is advantageous. On the other hand, reducing N in a blinded respectively uncontrolled way like in Algorithm 5.1 can result in loosing viability as soon as disturbances, changes in the reference x_f or dynamically changing constraints occur. For the latter, consider an autonomous vehicle navigation scenario as an example, in which a dynamic obstacle penetrates the original trajectory requiring an expansion of the trajectory.
In the following, an enhanced scheme which adapts the grid size N dependent on the intermediate Δt^* and a specified desired temporal resolution $\Delta t_\text{ref} \in \mathbb{R}^+$ as reference is introduced. This way, the control engineer can specify a practicable and reasonable trade off between the system dynamics accuracy, the number of controls and the computation time. The basic idea is to solve the nonlinear program repeatedly and to update the grid before each subsequent iteration when $|\Delta t - \Delta t_\text{ref}| > \Delta t_\epsilon$ is fulfilled. Hereby, the reference $\Delta t_\text{ref} > 0$ is usually related to the inherent sampling time and $\Delta t_\epsilon = 0$ must hold in order to match the desired resolution exactly. However, to avoid oscillations in terms of inserting and removing grid partitions, a small hysteresis $\Delta t_\epsilon > 0$ should be imposed. The overall procedure is described in Algorithm 5.2. Let $i = 1, 2, \ldots, I_\text{adapt}$ denote the number of outer loop iterations.
The initialization and update of optimization parameters in line 2 follows the strategy as described in Section 5.1 and the initial grid partition length is initialized with $\Delta t = \Delta t_\text{ref}$. The grid is either adapted for $i > 1$ or if the set $\mathcal{P}_{\text{global}}$ is warm-stared from the previous iteration. In the following, variables are augmented by superscript $\{i\}$ whenever their relation to a specific iteration i is of particular importance, for example $\Delta t^{\{i\}}$, $N^{\{i\}}$ or $\mathcal{P}_{\text{global}}^{\{i\}}$. The optimized parameter set in every iteration, which is the

result of line 6, is denoted as $\mathcal{P}_{\text{global}}^{\{i\}}$. Thus $\mathcal{P}_{\text{global}}^{\{i+1\}} = \mathcal{P}_{\text{global}}^{\{i\}}$ is implicitly fulfilled in every new iteration $i = 1, 2, \ldots, I_{\text{adapt}} - 1$.

A rather aggressive adaptation strategy is to estimate a proper $N^{\{i+1\}}$ based on the optimized time interval $\Delta t^{*\{i\}}$ of iteration i:

$$N^{\{i+1\}} = \begin{cases} \min\left(\max\left(\frac{\Delta t^{*\{i\}}}{\Delta t_{\text{ref}}} N^{\{i\}}, N_{\min}\right), N_{\max}\right) & |\Delta t^{*\{i\}} - \Delta t_{\text{ref}}| \geq \Delta t_\epsilon \\ N^{\{i\}} & \text{otherwise} \end{cases}. \tag{5.2.1}$$

The upper bound $N_{\max}$ provides a safeguard to limit the maximum computation time if desired. In case $\Delta t^{*\{i\}} \gg \Delta t_{\text{ref}}$, the continuous-time system dynamics are already well approximated after invoking line 6, hence uniformly thinning out multiple grid partitions at once does not change the optimal trajectories significantly. On the other hand, if $\Delta t^{*\{i\}} \ll \Delta t_{\text{ref}}$ the system dynamics are not yet approximated adequately such that the optimized trajectory might differ substantially from the desired one. For instance, fast dynamics related to small time constants might not be considered at stage i due to a coarse grid. After inserting $N^{\{i+1\}} - N^{\{i\}}$ at once, the nonlinear program solver needs to recover from this major structure change. Especially in the case of early terminated solving in line 6, $\Delta t^{*\{i+1\}}$ can be a poor intermediate result whenever the optimization at iteration i is computationally expensive, for example if $N^{\{i+1\}} - N^{\{i\}} \gg 0$.

A linear search strategy with a hysteresis Δt_ϵ implements similar to the shrinking horizon case, in which the grid size reduces only by one in each iteration, a more careful grid adaptation:

$$N^{\{i+1\}} = \begin{cases} N^{\{i\}} + 1 & \Delta t^{*\{i\}} > \Delta t_{\text{ref}} + \Delta t_\epsilon \wedge N < N_{\max} \\ N^{\{i\}} - 1 & \Delta t^{*\{i\}} < \Delta t_{\text{ref}} - \Delta t_\epsilon \wedge N > N_{\min} \\ N^{\{i\}} & \text{otherwise} \end{cases}. \tag{5.2.2}$$

Whenever the grid is requested to change, in particular if $N^{\{i+1\}} \neq N^{\{i\}}$ in (5.2.2) or (5.2.1), the implementation resamples the grid linearly by interpolating each state and control sequence with respect to time. As a consequence, the updated time interval is $\Delta t^{\{i+1\}} = \Delta t^{*\{i\}} N^{\{i\}} / N^{\{i+1\}}$.

The nonlinear programs are solved as described before. Algorithm 5.2 includes the feedforward action $N^{\{i+1\}} = N^{\{i\}} - 1$ in line 8 of Algorithm 5.1. Otherwise, $I_{\text{adapt}} = 1$ leads to a delayed grid reduction as the actual change of Δt^* is available after the solution to the nonlinear program.

Corollary 5.2.1. *Consider nonlinear programs* (4.1.8), (4.1.12) *or* (4.1.17). *Let* $\mathbb{X}_{\text{f}} = \{x_{\text{f}}\}$ *represent a steady state such that there exists* $u \in \mathbb{U}$ *with* $f(x_{\text{f}}, u) = 0$ *and assume that Assumption 3.1.1 and for any initial* $x_\mu(t_{\mu,0}) = x_{\text{s}} \in \mathbb{X}$ *Assumption 4.2.2 hold. Furthermore, choose* $N_{\min} > 0$ *and* $\Delta t_{\min}$ *such that* $0 \leq \Delta t_{\min} < \Delta t^*$ *holds at time instance* $t_{\mu,0}$. *The closed-loop feedback is realized with Algorithm 5.2 at sampling times inherited from the open-loop solution.* N_{init} *is chosen such that* $|\Delta t_{\text{ref}} - \Delta t^*| \leq \Delta t_\epsilon$ *holds for the very first sampling time* $t_{\mu,0}$. *Then, the closed-loop system is P-practically asymptotically stable on* $\mathbb{X}$ *with* $P = P_c(x_{\text{f}}, N_{\min}, (N_{\min} - 1)\Delta t^*)$.

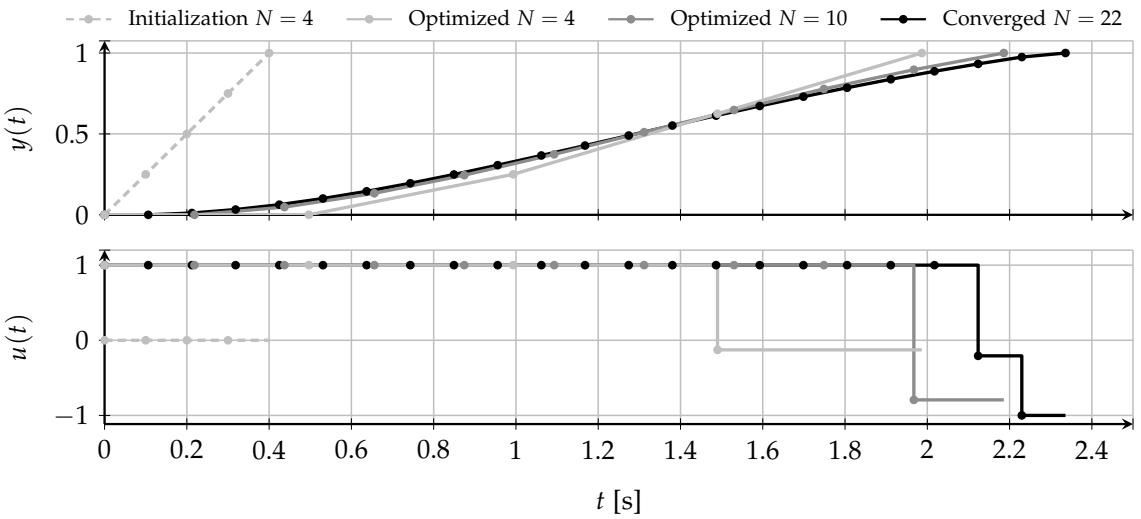


Figure 5.3.: Example of the enhanced grid adaptation for the second-order system (5.2.3) with a straight line initialization in state space and zero controls.

Appendix B.2 provides the proof. In the case N_{init} cannot be chosen appropriately to impose $\Delta t_{\text{ref}} = \Delta t^*$, I_{adapt} must be set sufficiently large to find a suitable N in the first invocation of Algorithm 5.2.
For practical reasons, especially with limited computational resources, it is often advantageous to terminate the optimization prematurely, for example by specifying a maximum number of solver iterations or by limiting the computation time. The outer optimization loop with I_{adapt} iterations calls the optimization several times and adapts the grid simultaneously. Therefore it is often unnecessary to require high-precision intermediate solutions in line 6. Consequently, the total computation time of Algorithm 5.2 does not differ significantly from that of a one-step optimization without grid adaptation if I_{adapt} and premature terminate conditions are selected accordingly. This does not apply to other approaches with two-stage optimization like in [Van+11a], where the solver cannot generate a trajectory for too coarse grids.

Example 5.2.1 (Grid Adaptation)**.** Figure 5.3 shows an example of the grid adaptation scheme for an input-constrained second-order system with $u(t) \in [-1, 1]$, $y(t) \in \mathbb{R}$ and state vector $x(t) = \big(x_1(t), x_2(t)\big)^\mathsf{T} \in \mathbb{R}^2$:

$$\begin{aligned} \dot{x}(t) &= \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t), \\ y(t) &= \begin{pmatrix} 1 & 0 \end{pmatrix} x(t). \end{aligned} \tag{5.2.3}$$

The control task is specified with initial state $(0, 0)^\mathsf{T}$ and final state $x_\mathrm{f} = (1, 0)^\mathsf{T}$. The desired temporal resolution is set to $\Delta t_{\text{ref}} = 0.1$. For brevity, only the evolution of $y(t)$ and $u(t)$ is depicted in Figure 5.3. The state and control sequences are initialized with $N_{\text{init}} = 4$ in Algorithm 5.2 which are also depicted as dashed lines in Figure 5.3. Intermediate results of the optimization for $N = 4$ and $N = 10$ are shown, as well as the final result. It is noticeable that the optimization result with $N = 4$ already provides a reasonable approximation of $N = 22$ which then increases by adding more grid partitions.

Example 5.2.2 (Closed-Loop Control with Grid Adaptation)**.** Consider the Van der Pol oscillator as in Example 5.1.1. At $t_\mu = 2.5\,\mathrm{s}$ the final state switches from $x_\mathrm{f} = (0.8, 0)^\intercal$ to $x_\mathrm{f} = (-0.5, 0)^\intercal$. Forward differences are utilized as transcription method with $\Delta t_\mathrm{min} = 1 \cdot 10^{-7}\,\mathrm{s}$. The grid adaptation scheme (Algorithm 5.2) is configured with $N_\mathrm{min} = 2$, $N_\mathrm{max} = 100$ and $\Delta t_\epsilon = 0.1\Delta t_\mathrm{ref}$. The closed-loop sampling time is inherited from the optimal control problem and is further bounded to the interval $[0.01\,\mathrm{s}, 0.1\,\mathrm{s}]$. Figure 5.4 shows the control, state and grid size evolutions for three configurations. For $\Delta t_\mathrm{ref} = 0.1\,\mathrm{s}$ (with grid adaptation according to (5.2.2)) the initial grid size $N_\mathrm{init} = 15$ is already chosen properly as N does not increase. Hereby, I_adapt is set to $I_\mathrm{adapt} = 10$ and as soon as x_f changes, the grid size increases to keep Δt_ref. For the other two cases only one iteration $I_\mathrm{adapt} = 1$ is performed ($\mathcal{P}_\mathrm{global}$ is still warm-started). In addition, Δt_ref is reduced to $\Delta t_\mathrm{ref} = 0.01$, and the initial grid size is set to $N_\mathrm{init} = 50$. Gaussian noise with zero mean and a standard deviation of 0.01 is added as measurement noise in both states (the plant simulation model is not affected). Both the linear grid adaptation (5.2.2) and the aggressive strategy (5.2.1) are shown. Obviously, the aggressive strategy results in a strong oscillating grid size N. Both the low choice of $I_\mathrm{adapt} = 1$ and the additional noise lead to a large N upon arrival at the steady state. Similar to Example 5.1.1, the state trajectories oscillate around x_f (especially $x_{\mu,2}(t_\mu)$). Also, the amount of chattering due to noise is high.

Example 5.2.2 shows that the grid size adaptation can be successfully integrated into the closed-loop system to keep the desired grid resolution Δt_ref. On the other hand, the results on stabilizing at steady state x_f might not be satisfactory for certain applications. This is due either to the effects of the nonlinear program and its numerical solution, or to the chattering present in any time-optimal control system with disturbances. Thus, if stabilization is desired, it is recommended to switch to a suitable stabilizing control scheme near x_f. This can be realized by smooth switching either as a result of a hybrid cost function or explicitly as in dual-mode control.

5.3. Smooth Stabilizing Control

The previous examples show that time-optimal control leads to chattering while stabilizing the system at the desired steady state. In practice, often chattering drastically limits the lifetime of mechanical components and hence a smooth stabilizing control is generally preferred. Two approaches are presented below but the exposition is kept brief since established state feedback control and conventional MPC concepts apply.

5.3.1. Dual-Mode Control

Dual-mode MPC is the predecessor to quasi-infinite horizon MPC concerning stability enforcement [May+00]. The key idea is to control the system to a terminal region $\mathbb{X}_\mathrm{f}$ and then switch to an external local stabilizing controller [MM93; Chi+96].
Let x_f denote the steady state at which the system should be stabilized and u_f the corresponding control such that $f(x_\mathrm{f}, u_\mathrm{f}) = 0$ holds. Assume that the linearized system $\dot{x}_\mathrm{lin}(t) = Ax_\mathrm{lin}(t) + Bu_\mathrm{lin}(t)$ with $A := \mathrm{D}_x f(x_\mathrm{f}, u_\mathrm{f})$ and $B := \mathrm{D}_u f(x_\mathrm{f}, u_\mathrm{f})$ is stabilizable,

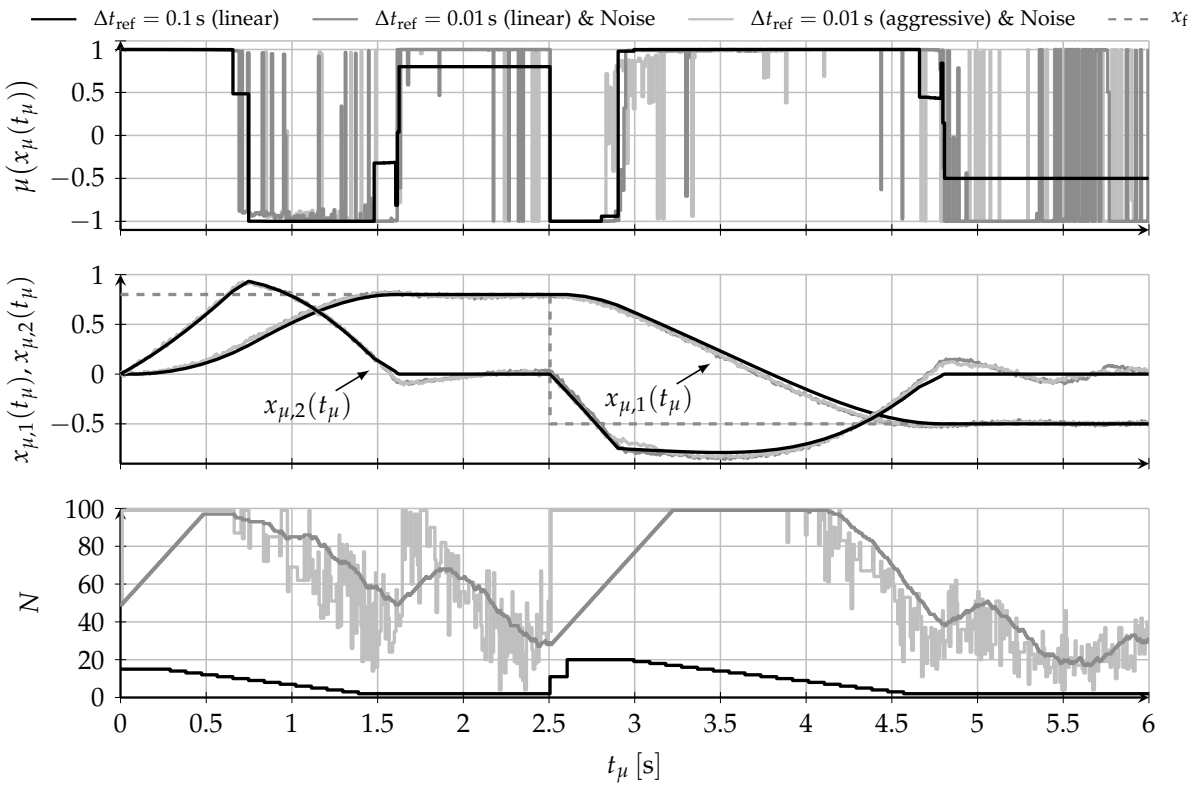


Figure 5.4.: Closed-loop control of the Van der Pol oscillator with grid size adaptation and varying final state x_{f}. The first realization (black) stabilizes the system although only practical stability holds. Notice the peak in control before reaching x_{f}. Two realizations with smaller Δt_{ref} include measurement noise to emphasize chattering in control and grid adaptation.

in particular the dynamics of all uncontrollable states are stable. Then, a linear state feedback $K \in \mathbb{R}^{q \times p}$ can be determined such that $A - BK$ is asymptotically stable. However, since the original plant is nonlinear and constraints are present, the region of operation is limited to some set $\mathbb{X}_{\text{lin}} \subseteq \mathbb{X}$. The set $\mathbb{X}_{\text{lin}}$ is determined such that the local feedback law $\mu_{\text{lin}}\big(x_\mu(t_\mu)\big) = K(x_{\text{f}} - x_\mu(t_\mu)) + u_{\text{f}}$ is admissible with $\mu_{\text{lin}}\big(x_\mu\big) \in \mathbb{U}$ for all $x_\mu \in \mathbb{X}_{\text{lin}}$. Furthermore, the closed-loop system must be rendered forward invariant, in particular $\varphi\big(t_\mu, x_{\text{s}}, \mu\big(x_\mu(t_\mu)\big)\big) \in \mathbb{X}_{\text{lin}}$ for all $x_{\text{s}} \in \mathbb{X}_{\text{lin}}$ and $t_\mu \geq 0$. These requirements on $\mathbb{X}_{\text{lin}}$ ensure feasibility. However, to also ensure asymptotic stability, $\mathbb{X}_{\text{lin}}$ is further limited to provide a sufficient decrease of a suited Lyapunov function for the nonlinear closed-loop system. For a profound description refer to [CS82; MM93].

Remark 5.3.1. The linear feedback controller is usually obtained by the well known linear quadratic regulator (LQR) design. In practice, it is often sufficient to determine $\mathbb{X}_{\text{lin}}$ by discretizing the state space in the vicinity of x_{f} and by performing closed-loop simulations with the linear controller and the nonlinear system.

Combining the control law from the linear controller design above with (5.1.1) results in the following dual-mode control law:

$$\mu_{\text{dual}}\big(x_\mu(t_\mu)\big) = \begin{cases} \mu\big(x_\mu(t_\mu)\big) & \text{for } x_\mu(t_\mu) \notin \mathbb{X}_{\text{lin}} \\ \mu_{\text{lin}}\big(x_\mu(t_\mu)\big) & \text{for } x_\mu(t_\mu) \in \mathbb{X}_{\text{lin}} \end{cases}. \tag{5.3.1}$$

In order to ensure feasibility of the combined control law, $\mathbb{X}_\mathrm{f} \subseteq \mathbb{X}_\mathrm{lin}$ must hold, since otherwise the time-optimal controller would not reach $\mathbb{X}_\mathrm{lin}$. In fact, the theoretical investigations in Section 5.1 considered $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$ as stated in Theorem 5.1.1. Accordingly, even $P_c\big(x_\mathrm{f}, N_\mathrm{min}, (N_\mathrm{min}-1)\Delta t^*\big) \subseteq \mathbb{X}_\mathrm{lin}$ must hold to ensure a proper convergence to $\mathbb{X}_\mathrm{lin}$:

Corollary 5.3.1. *Consider the closed-loop control system and the assumptions according to Theorem 5.1.1 or Corollary 5.2.1 with design parameters N_min and Δt_min. The resulting control law is denoted by $\mu(x_\mu)$. Furthermore, consider a control law $\mu_\mathrm{lin}(x_\mu)$ which ensures asymptotic stability and forward invariance for the nonlinear system for all states $x_\mu \in \mathbb{X}_\mathrm{lin}$. Then, the closed-loop control system with composite control law* (5.3.1) *and steady state x_f is asymptotically stable on $\mathbb{X}$ if $P_c\big(x_\mathrm{f}, N_\mathrm{min}, (N_\mathrm{min}-1)\Delta t^*\big) \subseteq \mathbb{X}_\mathrm{lin}$ holds.*

The proof is straightforward and follows immediately from the previous results and observations (refer to Appendix B.2).

5.3.2. A Hybrid Cost Function for Smooth Stabilizing Control

In case the region $\mathbb{X}_\mathrm{lin}$ for the dual-mode controller cannot be determined in advance, for example, if the desired set point x_f varies frequently or is time-varying, a hybrid cost function can be specified. Even though not in the context of time-optimal control, [MM93; May95] provide results for conventional quadratic form cost functions with a variable final time t_f. In the context of time-optimal control, [Zha+04] combines the minimum-time and the quadratic form cost functions to allow for smooth stabilizing control. Hereby, it is crucial to define a lower bound $t_\mathrm{f} \geq t_\mathrm{min}$ to force the controller to behave like a conventional model predictive controller with stability enforcing terminal conditions close to a steady state x_f. The continuous-time optimal control problem is defined as follows:

$$\min_{u(t),t_\mathrm{f}} \Big[t_\mathrm{f} + \int_{t_0=0}^{t_\mathrm{f}} \ell\big(x_u(t), u(t)\big)\,\mathrm{d}t \Big] \tag{5.3.2}$$

subject to

$$x_u(t_0 = 0) = x_\mu(t_\mu), \quad x_u(t) \in \mathbb{X}, \quad u(t) \in \mathbb{U}, \quad x_u(t_\mathrm{f}) \in \mathbb{X}_\mathrm{f}, \quad t_\mathrm{f} \geq t_\mathrm{min},$$
$$\dot{x}_u(t) = f\big(x_u(t), u(t)\big).$$

Hereby, the running costs are defined as $\ell\big(x_u(t), u(t)\big) := \big(x_u(t) - x_\mathrm{f}\big)^\mathsf{T} Q \big(x_u(t) - x_\mathrm{f}\big) + + \big(u(t) - u_\mathrm{f}\big)^\mathsf{T} R \big(u(t) - u_\mathrm{f}\big)$ with positive semi definite weighting matrices $Q \in \mathbb{R}^{p\times p}$ and $R \in \mathbb{R}^{q\times q}$. Constant u_f is defined as in the previous section. In order solve (5.3.2), [Zha+04] applies the time transformation and multiple shooting.
In the context of variable discretization grids, direct transcription can be applied as introduced in Chapter 4. The inherent lower bound on the grid partition Δt_min already implies a lower bound on t_f. Note, the integral of the running cost in (5.3.2) must be considered appropriately in direct transcription. For many simple problems, the Riemann sum is sufficient, especially if Δt_min is small. In general, it is preferred to solve the integral with the same accuracy as the system dynamics equation. To this end, the integral is approximated by the (left) Riemann sum, the midpoint or the trapezoidal

rule in case of the finite difference collocation methods. In the case of multiple shooting it is often favorable to include the cost function directly into the initial value problems of the system dynamics [GP17]. For Hermite-Simpson collocation, the Simpson rule can be applied immediately as intermediate states $x_{k+0.5}$ and controls $u_{k+0.5}$ are available. Notice that the hybrid cost approach does not ensure recursive feasibility: as soon as $P_\Delta t_{\min}(\cdot)$ is reached, the controller behaves like a standard stabilizing model predictive controller. However, recursive feasibility is only guaranteed up to $P_c(\cdot)$ and theoretical findings above show that the gap between reaching both sets is non-empty. Although it is often negligible in practice, this theoretical result is particularly interesting.

Example 5.3.1 (Closed-Loop Control of the *ECP Plant Emulator 220*)**.** Consider the dynamic model of the *ECP plant emulator 220* according to (3.3.7) for which a hybrid, a dual-mode and a standalone time-optimal controller are now realized. The time-optimal controller is configured with $\Delta t_{\text{ref}} = 0.01$, $N_{\text{init}} = 50$, $\Delta t_{\min} = 0.001$, $N_{\min} = 2$ and $\mathbb{X}_\text{f} = \{x_\text{f}\}$. The LQR for dual-mode operation is obtained by linearizing (3.3.7) at $x_\text{f} = (x_{\text{f},1}, x_{\text{f},2})^\intercal \in \mathbb{R}^2$ and $u_\text{f} \in \mathbb{R}^2$:

$$\dot{x}_{\text{lin}}(t) = \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 12.63\tanh(5x_{\text{f},2})^2 - 14.1 \end{pmatrix}}_{\approx A} x_{\text{lin}}(t) + \underbrace{\begin{pmatrix} 0 & 0 \\ 34.51 & -34.13 \end{pmatrix}}_{\approx B} u_{\text{lin}}(t). \tag{5.3.3}$$

Since any steady state of (3.3.7) requires $x_2(t) = 0$ and $u(t) = (0,0)^\intercal$, the remaining variable $x_{\text{f},2}$ is set to 0 and the resulting linear system consists of constant matrices A and B. This is beneficial in case the target position $x_{\text{f},1}$ is subject to change as in Section 8.3. For this particular example, consider $x_\text{f} = (0,0)^\intercal$ as steady state of interest. The linear system is controllable and hence stabilizable since (A, AB) has full row rank. Weight matrices for the LQR design are set to $Q = \text{diag}(1, 0.1)$ and $R = \text{diag}(1,1)$ (both positive definite). The resulting feedback matrix is

$$K = \begin{pmatrix} 0.711 & 0.131 \\ -0.703 & -0.130 \end{pmatrix}. \tag{5.3.4}$$

The region of attraction $\mathbb{X}_{\text{lin}}$ is obtained by performing closed-loop simulations with the nonlinear system (3.3.7) and feedback $\mu_{\text{lin}}(x_\mu(t_\mu)) = -Kx_\mu(t_\mu)$ according to a predefined grid of resolution 0.2. Restricting the region to the largest inscribed ellipse results in:

$$\mathbb{X}_{\text{lin}} = \{x \in \mathbb{R}^2 \mid x^\intercal \begin{pmatrix} 2.2 & 0.38 \\ 0.38 & 0.11 \end{pmatrix} x \leq 1\}. \tag{5.3.5}$$

The control task consists of reaching $x_\text{f} = (0,0)^\intercal$ from initial states $(-7,0)^\intercal$, $(5,0)^\intercal$ and $(-7,0)^\intercal$ in minimum time. The first initial state corresponds to the longest transition and its optimal initial time interval is $\Delta t^* \approx 0.03\,\text{s}$ for $N = 50$. In order to determine set $P_c\big(x_\text{f}, N_{\min}, (N_{\min} - 1)\Delta t^*\big)$, the reverse-time system is simulated with $N_{\min} = 2$ controls and the overestimated transition time $(N_{\text{init}} - 1)\Delta t^* \leq 0.04\,\text{s}$. Since the system has the bang-bang property, only combinations of maximum control values need to be tested to obtain the reachable states with maximum distance to x_f in the given time, in particular $u = (0.5, -0.5)^\intercal$ respectively $u = (-0.5, 0.5)^\intercal$. The four combinations result

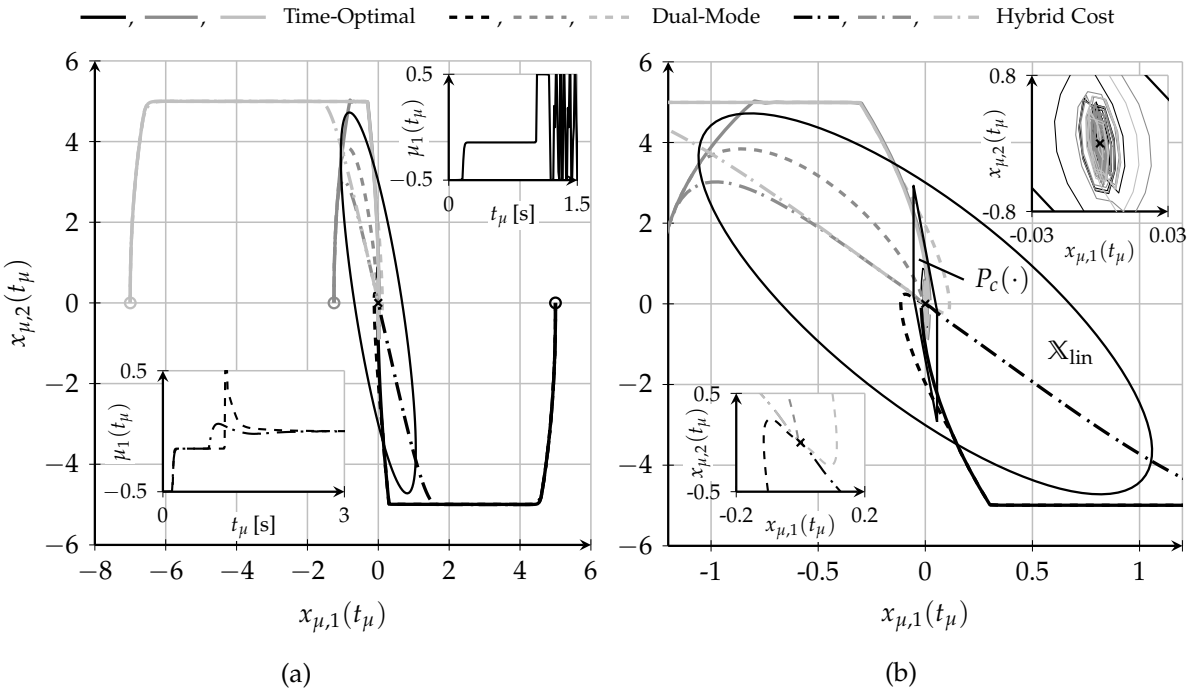


Figure 5.5.: Closed-loop results for the *ECP Model 220* in simulation. Three individual solutions from different initial states (indicated by circles) are shown and the steady state $x_\mathrm{f} = (0, 0)^\intercal$ is marked by a cross. The overall evolution of the state trajectories in the phase plane and the corresponding control trajectories for the first input are depicted in (a) whereas figure (b) shows magnified views for the vicinity of x_f.

in the following reachable states $(5 \cdot 10^{-2}, -2.93)^\intercal$, $(5 \cdot 10^{-2}, -0.18)^\intercal$, $(-5 \cdot 10^{-2}, 2.93)^\intercal$ and $(-5 \cdot 10^{-2}, 0.18)^\intercal$. Figure 5.5b shows both regions for practical stability and the LQR. Notice that the convex hull approximation of $P_c(\cdot)$ is in the interior of $\mathbb{X}_{\text{lin}}$. The hybrid cost optimal control problem (5.3.2) is defined according to the actual time-optimal controller with additional weight matrices Q and R as defined for the LQR above. Furthermore, the integral of the running cost is approximated by the Riemann sum. Figure 5.5 shows the simulated closed-loop trajectories in the phase plane and the corresponding controls. Note, the same numerical integration scheme (forward Euler) is applied for prediction and closed-loop simulation. The dual-mode controller switches to different trajectories as soon as $\mathbb{X}_{\text{lin}}$ is reached and the corresponding control is chatter-free. The hybrid controller also realizes a smooth transition as a compromise between the minimum-time and the quadratic form cost. Further tuning of the weights Q and R leads to more conservative respectively aggressive transitions for both the dual-mode and the hybrid cost approach.

6

Local Uniform and Quasi-Uniform Grid Representations

This chapter introduces a direct approach to time-optimal control which differs from Chapter 4 in that the underlying grid consists of *individual local time intervals* instead of relying on a global uniform time interval Δt. Nevertheless, the individual time intervals are required to maintain a uniform grid at least approximately in order to preserve the accuracy of the nonlinear program using different numerical integration or differentiation schemes and to enforce constraints at the grid points. Even though the numerical results are expected similar to Chapter 4, the optimal control problem with local time intervals shows a different sparsity pattern than with a single global time interval. Sparsity patterns are important when the nonlinear programs are solved and derivatives of cost and constraint functions with respect to the optimization parameters have to be calculated. It is often crucial for efficiency to a priori exclude all calculations that lead to structured zero values. Parts of this chapter have been published in [Rös+16a; Rös+17g].

6.1. Uniformity Enforcement by Equality Constraints

The continuous-time time-optimal control problem (3.2.2) is now transformed into a nonlinear program via direct transcription. Similar to Chapter 4, particular realizations in this thesis are collocation based on finite differences, multiple shooting and Hermite-Simpson collocation. However, the $k = 0, 1, \ldots, N-1$ grid partitions are now of individual lengths $\Delta t_k \in \mathbb{R}_0^+$:

$$\begin{aligned} t_0 \leq t_0 + \Delta t_0 \quad &= t_1, \\ t_1 \leq t_1 + \Delta t_1 \quad &= t_2, \\ \vdots \quad & \\ t_{N-1} \leq t_{N-1} + \Delta t_{N-1} \quad &= t_N = t_\mathrm{f}. \end{aligned} \tag{6.1.1}$$

Time instances at grid points t_k with $k = 0, 1, \ldots, N$ are obtained by $t_k = t_0 + \sum_{\bar{k}=0}^{k-1} \Delta t_{\bar{k}}$. Consequently, the final time t_f is:

$$t_\mathrm{f} = t_0 + \sum_{k=0}^{N-1} \Delta t_k. \tag{6.1.2}$$

The previous nonlinear programs (4.1.8), (4.1.12) and (4.1.17) are rewritten with cost function (6.1.2) and additional equality constraints enforcing $\Delta t_k = \Delta t_{k+1}$ for $k = 0, 1, \ldots, N-2$. For example, the nonlinear program of the finite difference collocation method (4.1.8) is then given as follows:

$$\min_{\substack{u_0, u_1, \ldots, u_{N-1}, \\ x_0, x_1, \ldots, x_N, \\ \Delta t_0, \Delta t_1, \ldots, \Delta t_{N-1}}} \sum_{k=0}^{N-1} \Delta t_k \tag{6.1.3}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N = x_\mathrm{f}, \\
&x_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad \Delta t_0 \geq \Delta t_{\min}, \quad \Delta t_k = \Delta t_{k+1}, \\
&\frac{x_{k+1} - x_k}{\Delta t_k} = \phi_\mathrm{fd}(x_k, x_{k+1}, u_k), \\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

Solving the nonlinear program proceeds as described in Section 4.2. The other nonlinear programs are omitted as the conversion from the global to the local grid is analogue.
Note, if $\Delta t_k = \Delta t_{k+1}$ is not enforced, the resulting nonlinear program is underdetermined as the total time t_f can be partitioned in different ways into the individual summands Δt_k. While this ambiguity is generally undesired, Chapter 9 exploits this for bang-bang control systems. However, it is still possible to omit the $N-1$ equalities $\Delta t_k = \Delta t_{k+1}$ and still resolve the ambiguity by further modifying the above nonlinear program as described below.

6.2. Approximation to the Uniform Grid

Another formulation approximates the uniform grid using the cost function $\sum_{k=0}^{N-1} \Delta t_k^2$ and the result of the following proposition:

Proposition 6.2.1. *Let* (6.1.2) *relate the total time t_f with individual time intervals Δt_k for $k = 0, 1, \ldots, N-1$. Then, $\Delta t_k = t_\mathrm{f}/N$ are the minimizers of $\sum_{k=0}^{N-1} \Delta t_k^2$ for all k. The resulting grid is uniform.*

Proof. A possible way to derive $\Delta t_k = t_\mathrm{f}/N$ from scratch is to solve $\min \sum_{k=0}^{N-1} \Delta t_k^2$ analytically subject to (6.1.2) using the method of Lagrange multipliers. However, for brevity, the Cauchy-Schwarz inequality is utilized here to proof Proposition 6.2.1:

$$\left(\sum_{k=0}^{N-1} \Delta t_k \xi_k \right)^2 \leq \left(\sum_{k=0}^{N-1} \Delta t_k^2 \right) \left(\sum_{k=0}^{N-1} \xi_k^2 \right). \tag{6.2.1}$$

By setting $\xi_k = 1$ and hence $\sum_{k=0}^{N-1} \xi_k^2 = N$ as well as applying (6.1.2), the following lower bound on the cost $\sum_{k=0}^{N-1} \Delta t_k^2$ can be specified:

$$\frac{t_\mathrm{f}^2}{N} \leq \sum_{k=0}^{N-1} \Delta t_k^2. \tag{6.2.2}$$

In order to proof that $\Delta t_k = t_\mathrm{f}/N$ are the actual minimizers, the corresponding cost must lie on bound (6.2.2):

$$\sum_{k=0}^{N-1} \Delta t_k^2 = \sum_{k=0}^{N-1} \frac{t_\mathrm{f}^2}{N^2} = N\frac{t_\mathrm{f}^2}{N^2} = \frac{t_\mathrm{f}^2}{N} \geq \frac{t_\mathrm{f}^2}{N}. \tag{6.2.3}$$

□

Hence, squaring the individual time intervals results in a uniform grid and no additional equality constraints enforcing $\Delta t_k = \Delta t_{k+1}$ are necessary. Obviously, Proposition 6.2.1 ignores the fact that additional constraints are present. Depending on the system dynamics and constraint formulation the optimal solution could deviate from a uniform grid if it further reduces the overall cost. Even though (quasi-)uniformity does not hold for the general case, it is assumed that constraint sets are rather simple (for example box constraints).
Again, consider the finite difference formulation from (4.1.8) and the local grid (6.1.1) as an example. The nonlinear program with quasi-uniform grid is then defined as:

$$\min_{\substack{u_0,u_1,\ldots,u_{N-1},\\ x_0,x_1,\ldots,x_N,\\ \Delta t_0,\Delta t_1,\ldots,\Delta t_{N-1}}} N\sum_{k=0}^{N-1} \Delta t_k^2 \tag{6.2.4}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N = x_\mathrm{f},\\
&x_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad \Delta t_k \geq \Delta t_\mathrm{min},\\
&\frac{x_{k+1} - x_k}{\Delta t_k} = \phi_\mathrm{fd}(x_k, x_{k+1}, u_k),\\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

Remark 6.2.1. Note, the cost function here is chosen as $N\sum_{k=0}^{N-1} \Delta t_k^2$ instead of $\sum_{k=0}^{N-1} \Delta t_k^2$. This does not change the minimizer at all but allows for a fairer comparison with the other grids: solvers terminate with respect to some desired convergence thresholds. The costs of the global and local uniform grids equal t_f. For the unmodified quasi-uniform grid, the optimal cost function value is t_f^2/N as derived in the proof of Proposition 6.2.1. Multiplication by N results in costs $Nt_\mathrm{f}^2/N = t_\mathrm{f}^2$ and thus independence from N. Especially for $t_\mathrm{f} \approx 1$, the convergence thresholds can be set globally and computation times are more comparable, even for varying N.

Similar as before, the conversion of all other global uniform grid transcription methods to the quasi-uniform grid is straightforward according to the changes discussed here.

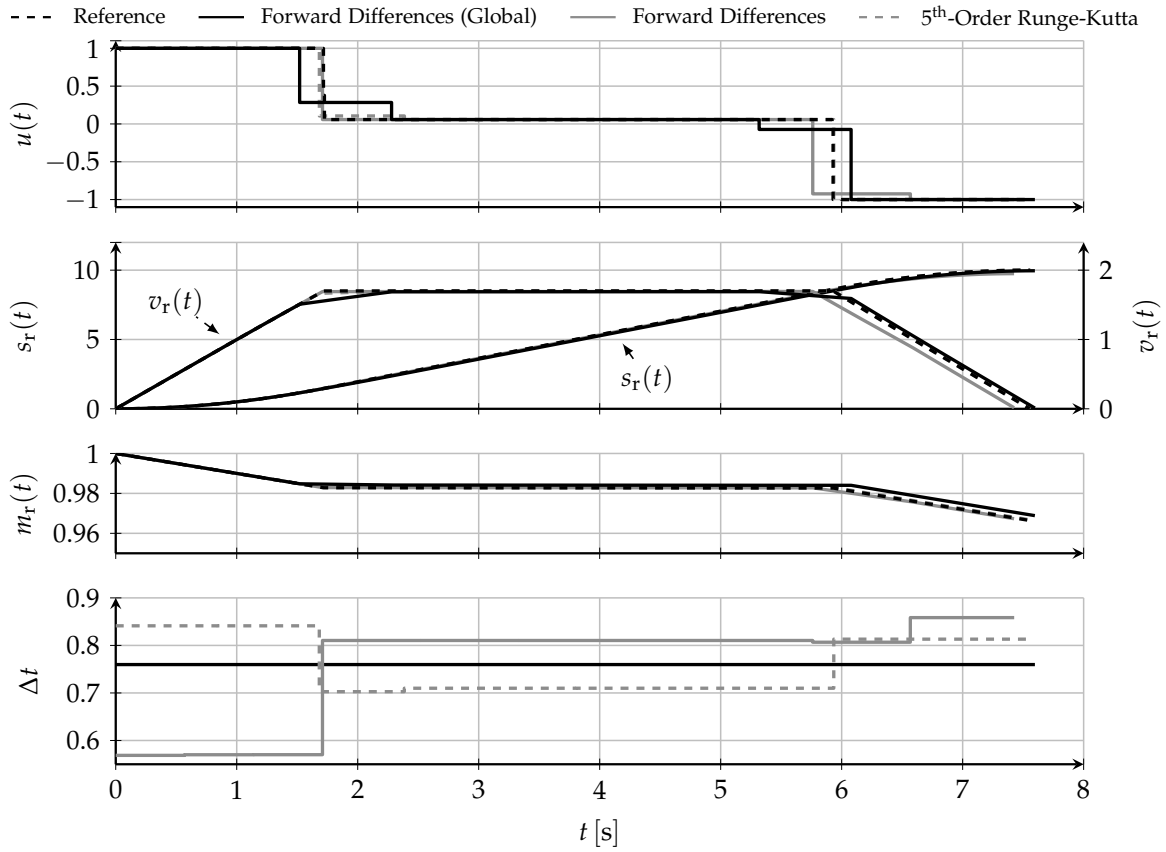


Figure 6.1.: Open-loop solutions for the rocket system with $s_{\mathrm{r,f}} = 10$ and $N = 10$. Multiple shooting is performed with the 5th-order Runge-Kutta scheme. The reference is obtained by the global uniform grid and $N = 1000$.

Example 6.2.1 (Open-loop Control of the Rocket System)**.** In this chapter, the rocket system is utilized for examples and demonstrations. Constraint sets $\mathbb{U}, \mathbb{X}$ and $\mathbb{X}_{\mathrm{f}}$ are defined according to Section 3.3.2. State $(0, 0, 1)^\intercal$ serves as initial state and the final rocket position is set to $s_{\mathrm{r,f}} = 10$. For initialization purposes, the final mass is set to 0.5, in particular it is $x_{\mathrm{f}} = (10, 0, 0.5)^\intercal$. Figure 6.1 shows the open-loop solutions for the quasi-uniform grid with collocation via forward differences as well as the 5th-order Runge-Kutta method. The grid size is set to $N = 10$. The global uniform grid with $N = 1000$ and convergence thresholds close to machine precision serves as reference. Furthermore, Figure 6.1 depicts the forward differences solution with a global uniform grid and $N = 10$.
The examination of single Δt_k shows that the resulting grid is not completely uniform. Interestingly, the switching points in the control trajectory are better approximated with respect to the reference than with respect to the uniform grid (the Runge-Kutta method even matches the second switching point). The solver is able to further reduce the total costs by slightly deviating from the uniform grid. Corresponding integral errors are shown in Figure 6.3 in combination with Hermite-Simpson types from Example 6.2.2. The results also show that the 5th-order Runge-Kutta method does not further improve the accuracy of the rocket dynamics as achieved with Hermite-Simpson collocation and constant controls.

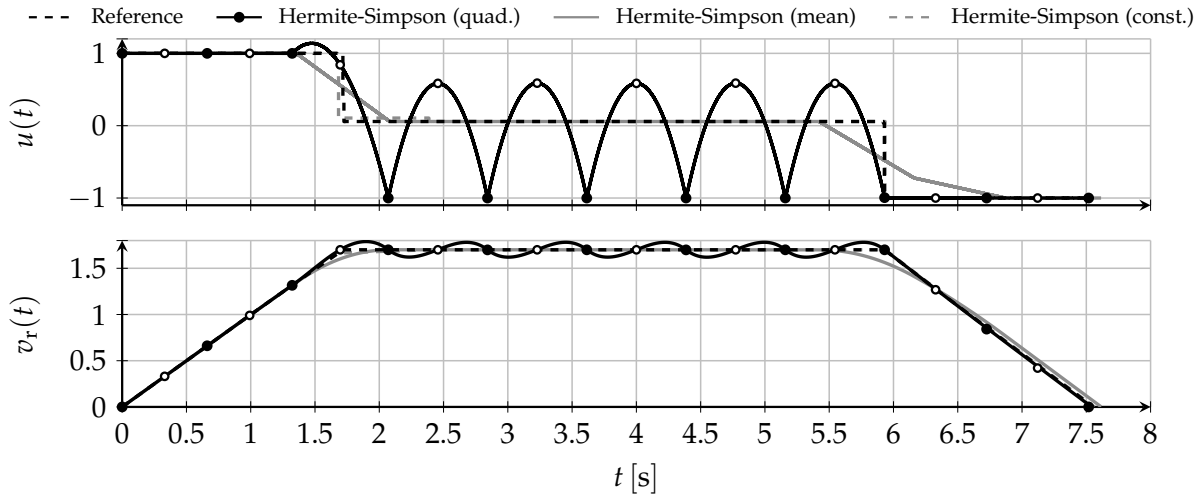


Figure 6.2.: Hermite-Simpson open-loop solutions for the rocket system with $s_{\mathrm{r,f}} = 10$ and $N = 10$. Similar to Figure 4.4, the additional degree of control freedom for the quadratic (and linear) spline representations violates state constraints. In contrast, the constant and mean (linear) representation adhere them. Filled circles represent grid points and open circles correspond to midpoints for Hermite-Simpson collocation with quadratic control splines.

Example 6.2.2. Consider the same control task and reference as in the previous example. In this case, several Hermite-Simpson collocation methods are compared. But unlike the analysis in Example 4.3.2 for the global uniform grid, different options for the control spline are taken into account for $N = 10$. Figure 6.2 shows the control trajectory $u(t)$ and the velocity $v(t)$ for three Hermite-Simpson realizations as the behavior at the active velocity bound is most conspicuous. Similar to the global uniform grid case, the quadratic control spline is not suitable for the task because it violates the velocity bound between the grid points. The linear control spline is omitted since the solution is similar but with intermediate $u(k), u(k+0.5), u(k+1)$ linearly connected. On the other hand, both the mean and the piecewise constant control variant are well suited, since they do not violate any constraints. Figure 6.3 reveals the least integral error for the constant control variant which is expected as the reference control is bang-bang as well. To conclude, depending on the actual application and the expected control characteristics, either the constant control or the mean variant in favor of linear or quadratic splines are more suitable.

6.3. Closed-Loop Control and Grid Adaptation

Adapting the grid with local time intervals is as important as for the global uniform case (refer to Section 5.2). Stability and recursive feasibility results apply similarly. The local uniform grid leads to identical solutions. For the quasi-uniform grid, inheriting sampling interval lengths from the open-loop solution already maintains recursive feasibility. The proofs for Lyapunov stability are similar by replacing $N\Delta t$ by the sum of N individual transitions of length Δt_k according to (6.1.2).

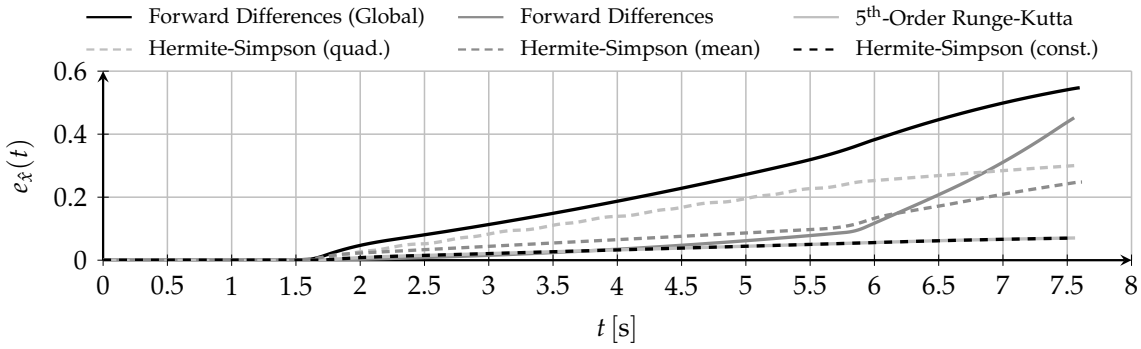


Figure 6.3.: Open-loop error related to the rocket system of Examples 6.2.1 and 6.2.2.

Algorithm 6.1.: Local/quasi-uniform grid optimal control with grid adaptation.

1: **procedure** SOLVEOCP($\mathcal{P}_{\text{local}}, x_\mu(t_{\mu,n}), x_f, \Delta t_{\text{ref}}, \Delta t_\epsilon, I_{\text{adapt}}, N$)
2: Initialize or update optimization parameters $\mathcal{P}_{\text{local}}$ ▷ Warm-starting
3: **for** $i = 1, 2, \ldots, I_{\text{adapt}}$ **do**
4: **if** $i > 1$ **or** warm-start **then**
5: **for** $k = 0, 1, \ldots, N-1$ **do**
6: **if** $\Delta t_k > \Delta t_{\text{ref}} + \Delta t_\epsilon$ **and** $N < N_{\max}$ **then**
7: Insert new grid partition by linear interpolation
8: **break**
9: **else if** $\Delta t_k < \Delta t_{\text{ref}} - \Delta t_\epsilon$ **and** $N > N_{\min}$ **then**
10: Remove grid partition
11: **break**
12: Solve nonlinear program w.r.t. $\mathcal{P}_{\text{local}}$ ▷ for example (6.2.4)
13: **if** $|\Delta t_k - \Delta t_{\text{ref}}| \leq \Delta t_\epsilon$ for all k **and** $N > N_{\min}$ **then**
14: $N \leftarrow N - 1$ ▷ New N only applies at next invocation
15: **return** $\mathcal{P}_{\text{local}}, N$ ▷ Note, the optimized control sequence is included

The previous strategy in Section 5.2 resamples the complete grid by linear interpolation. Although it is possible to apply a similar strategy, the following is easier to implement for local grids.

Algorithm 6.1 does not resample the entire state and control trajectories linearly, but only evaluates the first interval violating $|\Delta t_k - \Delta t_{\text{ref}}| > \Delta t_\epsilon$. The algorithm operates as follows. For finite difference collocation and multiple shooting with $M = N$ the set of all optimization parameters is denoted by $\mathcal{P}_{\text{local}} := \{u_0, u_1, \ldots, u_{N-1}, x_0, x_1, \ldots, x_N, \Delta t_0, \Delta t_1, \ldots, \Delta t_{N-1}\}$. On the other hand, the set for Hermite-Simpson collocation is $\mathcal{P}_{\text{local}} := \{u_0, u_{0.5}, u_1, \ldots, u_N, x_0, x_1, \ldots, x_N, \Delta t_0, \Delta t_1, \ldots, \Delta t_{N-1}\}$. The initialization step in line 2 initializes all time intervals with the reference value, in particular $\Delta t_k = \Delta t_{\text{ref}}$. The actual adaptation begins in line 5 when an index k is found that does not match the previous condition. If the resolution Δt_k is too low, a new grid partition respectively a state, control and time interval is inserted into the interval $[t_k, t_{k+1}]$ by linear interpolation (line 7). If the resolution is too high, the state, control and time interval are removed from $[t_k, t_{k+1}]$ in line 10.

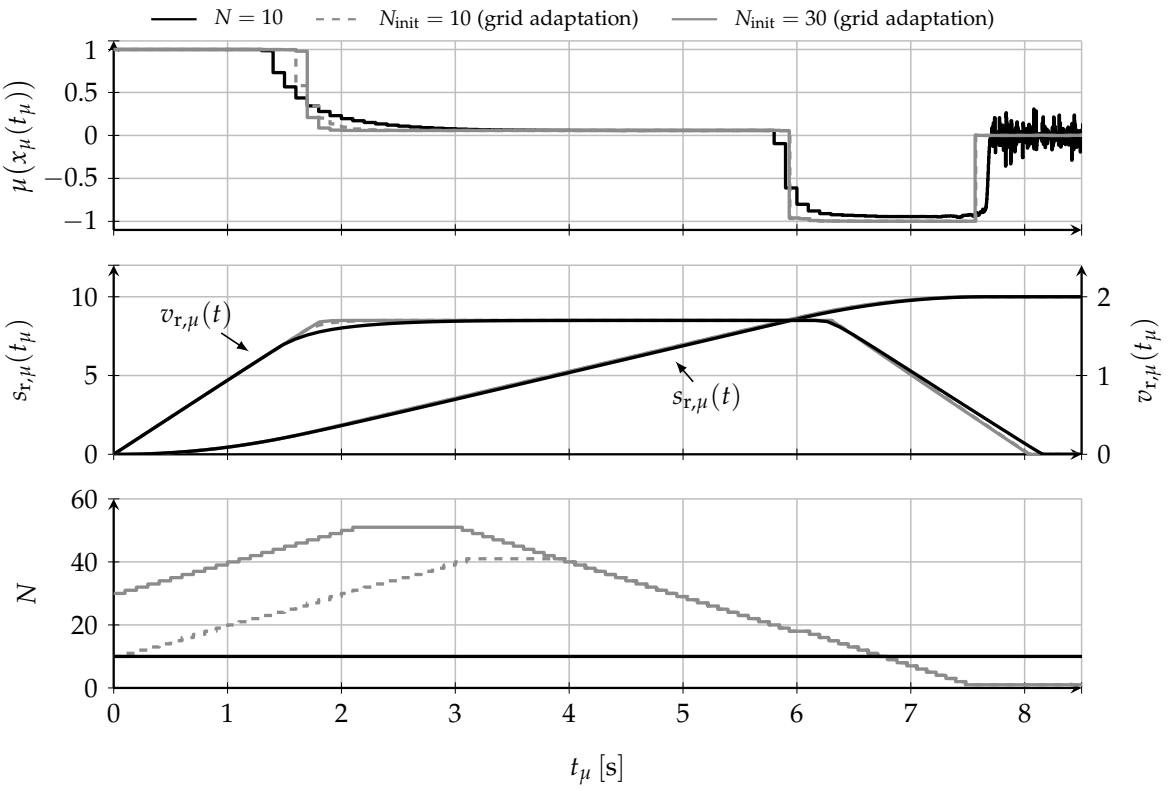


Figure 6.4.: Closed-loop control of the rocket system with multiple shooting (forward Euler) and a quasi-uniform grid. Two realizations perform with grid adaptation.

Line 12 calls the nonlinear program solver as described in Section 4.2. Closed-loop control is realized in the same way as in Chapter 5.

Example 6.3.1 (Closed-Loop Control of the Rocket System with a Quasi-Uniform Grid)**.** As an example consider the rocket system as before. Hereby, multiple shooting with forward Euler is chosen ($M = N$). The closed-loop sampling time is inherited from the optimal control problem and is further bounded to the interval $[0\,\text{s}, 0.1\,\text{s}]$. Figure 6.4 shows the results of three different configurations. First, closed-loop control is performed with $N = 10$ and no grid adaptation. Note, the resulting grid resolution is $\Delta t^* \approx 0.57\,\text{s}$ but the closed-loop sampling time is bounded to 0.1 s. This is intended to demonstrate a further effect of time-optimal MPC realizations (quasi-uniform and uniform grids). In case the closed-loop sampling time is lower than Δt^* and the switching point is in the grid interval $[t_k, t_{k+1}]$ for some k, then the controller does not switch instantaneously at t_k but more softly as the actual grid point moves for the underlying optimal control problems. Now consider the results with grid adaptation according to Algorithm 6.1, in particular with $N_{\text{init}} = 10$ and $N_{\text{init}} = 30$. The grid adaptation is configured with $\Delta t_{\text{min}} = 0.001\,\text{s}$, $\Delta t_{\text{ref}} = 0.1\,\text{s}$ and $\Delta t_\epsilon = 0.1\Delta t_{\text{ref}}$. As the grid resolution increases, the switches in the control trajectory are more discontinuous as $\Delta t^* \to \Delta t_{\text{ref}}$ and hence better matches the closed-loop sampling time.

7

Sparsity Structure Exploitation with Hypergraphs

The efficient solution of the optimal control problems respectively nonlinear programs as presented in the previous chapters requires first-order and sometimes second-order derivatives of the cost function and constraints. For larger and complex nonlinear programs it is often difficult or at least inconvenient to provide these derivatives analytically. Therefore, these derivatives are usually computed automatically, especially numerically. Numerical computation is often based on finite differences, for example central differences, since they balance the compromise between accuracy and computational effort. Sparse finite differences are already established in optimal control and optimization in general. Thereby sparsity patterns are identified by graph coloring techniques [Bet10; NW06]. On the other hand, automatic differentiation became more and more a popular and established method for the efficient computation of derivatives like in the ACADO toolkit [Hou+11a]. Automatic differentiation inherently retains the sparse structure of the nonlinear program and avoids the numerical evaluation of structured zero elements in first- and second-order derivatives. An open-source, profound and efficient framework is CasADi [And13] which has been widely used in the recent optimal control and MPC literature.
Related work in robotics and computer vision, simultaneous localizing and mapping as well as bundle adjustment uses hypergraphs for representing sparse optimization problems [Küm+11]. These least-squares optimization problems are large albeit unconstrained. In earlier work, the hypergraph was adapted to an unconstrained trajectory optimization problem for mobile robots [Rös+13a; Rös+15a; Rös+17a; Rös+17c]. Against this background, it is of considerable interest to extend the hypergraph and analyze its application for constrained (time-optimal) control problems in MPC, especially in comparison to the automatic differentiation alternative. These optimal control problems, in particular the previously proposed formulations, have a sparse structure. The representation of the optimal control problem as a hypergraph allows efficient derivative computations which are crucial for the solution in real-time. Consequently, the sparsity patterns for sparse finite differences are defined a priori by the problem design and are not automatically identified by graph coloring. The hypergraph is well suited to algorithmically change the sparsity pattern online by rearranging the set of vertices and edges.

This chapter proposes the use and extension of the hypergraph formulation to nonlinear programs with constraints in general and concentrates on its application to the time-optimal control methods described in the previous chapters. In addition to the related work, two strategies on computing the derivatives of cost and constraint terms are provided. The hypergraph formulation and benchmark results for conventional MPC with a fixed time grid are provided in Appendix D. Parts of this chapter have been published in [Rös+18a].

7.1. Hypergraph Representation for Nonlinear Programs

This section introduces the hypergraph for a nominal nonlinear program which is defined as follows:

$$\min_{z_0, z_1, \dots, z_\mathrm{Y}} J\big(z_0, z_1, \dots, z_\mathrm{Y}\big) \tag{7.1.1}$$

subject to

$$\begin{aligned} & z_\mathrm{min} \leq z_v \leq z_\mathrm{max},\ v = 0, 1, \dots, \mathrm{Y}, \\ & z_\omega = z_{\mathrm{ref},\omega},\ \omega \in \{0, 1, \dots, \mathrm{Y}\}, \\ & g\big(z_0, z_1, \dots, z_\mathrm{Y}\big) \leq 0, \\ & h\big(z_0, z_1, \dots, z_\mathrm{Y}\big) = 0. \end{aligned}$$

Hereby, variables $z_v \in \mathbb{R}$ with $v = 0, 1, \dots, \mathrm{Y}$ are the optimization parameters. The set of all optimization parameters is denoted as $\mathcal{Z} = \{z_0, z_1, \dots, z_\mathrm{Y}\}$. The nonlinear cost function $J : \mathcal{Z} \mapsto \mathbb{R}$ maps the set of optimization parameters to a scalar real number. Furthermore, $g : \mathcal{Z} \mapsto \mathbb{R}^R$ combines the inequality constraints and $h : \mathcal{Z} \mapsto \mathbb{R}^S$ defines the equality constraints with $R, S \in \mathbb{N}_0$. Lower and upper bounds on optimization parameters, z_min and z_max respectively, as well as trivial equality constraints $z_\omega = z_{\mathrm{ref},\omega}$ are defined separately from $g(\cdot)$ and $h(\cdot)$ since the hypergraph treats them dedicatedly. The provision of trivial equality constraints is optional, but most nonlinear programs in optimal control impose initial conditions and often further boundary conditions in this form.
Without loss of generality, $J(\cdot)$ is composed of $l = 1, 2, \dots, L$ accumulated terms, in particular:

$$J\big(z_0, z_1, \dots, z_\mathrm{Y}\big) = \sum_{l=1}^{L} J_l\big(z_0, z_1, \dots, z_\mathrm{Y}\big). \tag{7.1.2}$$

Ideally, individual terms $J_l : \mathcal{Z}_l^\mathrm{J} \mapsto \mathbb{R}$ depend only on a subset of optimization parameters $\mathcal{Z}_l^\mathrm{J} \subseteq \mathcal{Z}$. Note, $L = 1$ holds in case $J(\cdot)$ cannot be divided into several summands. Also, constraints might depend only on a few optimization parameters each. Let constraint $g(\cdot)$ be defined in terms of $r = 1, 2, \dots, R$ individual scalar parts $g_r : \mathcal{Z}_r^\mathrm{g} \mapsto \mathbb{R}$ with $\mathcal{Z}_r^\mathrm{g} \subseteq \mathcal{Z}$:

$$g\big(z_0, z_1, \dots, z_\mathrm{Y}\big) = \begin{pmatrix} g_1\big(z_0, z_1, \dots, z_\mathrm{Y}\big) \\ g_2\big(z_0, z_1, \dots, z_\mathrm{Y}\big) \\ \vdots \\ g_R\big(z_0, z_1, \dots, z_\mathrm{Y}\big) \end{pmatrix}. \tag{7.1.3}$$

Similarly, equality constraints $h(\cdot)$ are decomposed into $s = 1, 2, \ldots, S$ individual parts $h_s : \mathcal{Z}_s^{\mathrm{h}} \mapsto \mathbb{R}$ with $\mathcal{Z}_s^{\mathrm{h}} \subseteq \mathcal{Z}$.
In the following, the nominal nonlinear program (7.1.1) with decomposed cost function and constraints according to (7.1.2) and (7.1.3) is formulated in terms of a hypergraph with the objective to exploit the inherent sparse problem structure.
A hypergraph is a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ composed of a set of vertices $\mathcal{V}$ and a set of (hyper-)edges $\mathcal{E}$. Edges in a hypergraph connect several vertices rather than only pairs of vertices compared to regular graphs. Vertices and edges in the context of nonlinear program (7.1.1) are defined in the following.
A vertex refers to an optimization parameter $z_v \in \mathcal{Z}$ with $v = 0, 1, \ldots, \mathrm{Y}$. Furthermore, to facilitate efficiency from an implementation point of view, a vertex is augmented by specific properties:

- The upper and lower bounds on z_v, in particular $z_{\max}$ and $z_{\min}$ respectively, are directly cached in the vertex.
- A vertex is marked as *fixed* in the presence of trivial equality constraints such as $z_v = z_{\mathrm{ref},v}$. Hereby z_v is substituted by $z_{\mathrm{ref},v}$ and the so-called fixed vertex is not subject to optimization, but nevertheless appears in cost and constraint terms.

Thus a vertex is formally defined as $v_v \in \mathcal{V}$ with tuple $v_v = (z_v, z_{\min}, z_{\max}, z_{\mathrm{ref},v})$. Part $z_{\mathrm{ref},v}$ is omitted in case no trivial equality constraint is active for v_v. The vertex is then rendered non-fixed. The overall set of vertices is $\mathcal{V} = \{v_0, v_1, \ldots, v_{\mathrm{Y}}\}$.
An edge refers to a scalar cost term $J_l(\cdot)$ as well as equality and inequality constraints, $g_r(\cdot)$ and $h_s(\cdot)$ respectively. Also, edges are augmented by additional properties, for example:

- The type of the edge, in particular cost term, equality or inequality.
- Additional variables (for instance obtained from measurements) that are constant during optimization but are required for evaluating cost terms and constraints.

For ease of notation, a cost edge is defined as $e_l^{\mathrm{J}} \in \mathcal{E}$, an edge for equality constraints as $e_s^{\mathrm{h}} \in \mathcal{E}$ and an edge for inequality constraints as $e_r^{\mathrm{g}} \in \mathcal{E}$. The corresponding set of edges is $\mathcal{E} = \{e_1^{\mathrm{J}}, e_2^{\mathrm{J}}, \ldots, e_L^{\mathrm{J}}, e_1^{\mathrm{h}}, e_2^{\mathrm{h}}, \ldots, e_S^{\mathrm{h}}, e_1^{\mathrm{g}}, e_2^{\mathrm{g}}, \ldots, e_R^{\mathrm{g}}\}$.
By taking the definition of vertices and edges into account, an edge connects all vertices which the representing function or constraint depends on. In particular, a cost edge connects $|\mathcal{Z}_l^{\mathrm{J}}|$ vertices. Operator $|\cdot|$ retrieves the cardinality (the number of elements) from the underlying set and the cost term J_l is assumed to be associated with edge e_l^{J}. Accordingly, for equality and inequality edges, a number of $|\mathcal{Z}_s^{\mathrm{h}}|$ respectively $|\mathcal{Z}_r^{\mathrm{g}}|$ vertices are connected.
Note, from an implementation point of view, cost functions and constraints which share a similar set of parameters can be combined into a single edge, for instance if $|\mathcal{Z}_l^{\mathrm{J}} \cap \mathcal{Z}_s^{\mathrm{h}} \cap \mathcal{Z}_r^{\mathrm{g}}|$, $|\mathcal{Z}_l^{\mathrm{J}} \cap \mathcal{Z}_s^{\mathrm{h}}|$, $|\mathcal{Z}_l^{\mathrm{J}} \cap \mathcal{Z}_r^{\mathrm{g}}|$ or $|\mathcal{Z}_s^{\mathrm{h}} \cap \mathcal{Z}_r^{\mathrm{g}}|$ is large for some l, s and r. The idea behind considering a tupel of costs and constraints is to share common resources which is advantageous for computationally expensive operations such like higher-order numerical integrations. This can be beneficial especially in the context of optimal control,

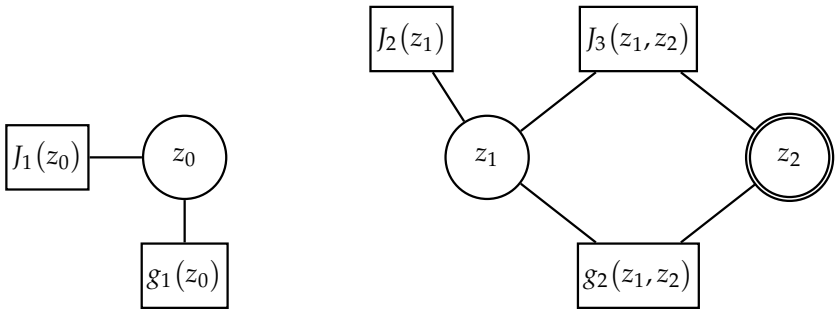


Figure 7.1.: Example of a hypergraph for nonlinear program (7.1.4). Vertices are represented as circles whereas rectangles refer to edges. A double circle indicates a fixed vertex.

in which several evaluations of cost terms and constraints rely on the very same intermediate solutions of an initial value problem. For ease of notation, edges are treated individually for the theoretical description throughout this thesis. In a practical implementation, a so-called mixed edge provides a pre-computation function for filling shared memory before individual functions for computing cost or constraint terms return the actual values.

Example 7.1.1 (Hypergraph Representation)**.** For clarity, a simple example demonstrates the representation of a nonlinear program as hypergraph. Consider the following nonlinear program with optimization parameters $z_0, z_1, z_2 \in \mathbb{R}$:

$$\begin{aligned} \min_{z_0,z_1,z_2} \quad & z_0^2 + z_1^2 + \left(z_1 + z_2\right)^2 \\ \text{subject to} \quad & \\ & z_2 = 1, \\ & \begin{pmatrix} -z_0^2 - 4 \\ z_1 + z_2 \end{pmatrix} \leq 0. \end{aligned} \tag{7.1.4}$$

For simplicity, non-trivial equality constraints $h(\cdot)$ are omitted. Figure 7.1 shows the corresponding hypergraph. Vertices are represented as circles whereas edges for the cost terms and constraints are depicted as rectangles. The cost function in (7.1.4) can be decomposed into the three summands $J_1(z_0) = z_0^2$, $J_2(z_1) = z_1^2$ and $J_3(z_1, z_2) = \left(z_1 + z_2\right)^2$. Equality $z_2 = 1$ is in the form of a trivial equality constraint which renders the associated vertex v_2 as fixed. Fixed vertices are indicated with a double circle in Figure 7.1. The inequality constraint is decomposed into parts $g_1(z_0) = -z_0 - 4$ and $g_2(z_1, z_2) = z_1 + z_2$.

7.2. Sparse Derivative Computations

Solving nonlinear programs like (7.1.1) efficiently, requires at least first-order derivatives of the cost function and constraint terms. Especially Newton-type solvers also require second-order derivatives. However, it is often preferable to iteratively refine the Hessian using the Broyden-Fletcher–Goldfarb-Shanno (BFGS) method rather than

calculating the Hessian explicitly [NW06]. On the other hand, the benchmark results in Chapter 8 demonstrate that utilizing the sparse Hessian obtained from the hypergraph performs well for the time-optimal case. For brevity, the mathematical and algorithmic description in this section focuses on first-order derivatives as second-order derivatives are computed similarly.
Consider the decomposition of (7.1.2) into summands $J_l(\cdot)$ with $l = 1, 2, \ldots, L$ and the set of non-fixed vertices $\tilde{\mathcal{Z}} = \{z_v \in \mathcal{Z} \mid \text{for all } v \text{ for which } v_v = (z_v, z_{\min}, z_{\max}, z_{\text{ref},v}) \text{ is not fixed}\}$. To clarify the difference between fixed and non-fixed vertices respectively parameters in the definition of the gradient, the sequence of non-fixed parameters are denoted by $\tilde{z}_{\tilde{v}} \in \tilde{\mathcal{Z}}$ with $\tilde{v} = 0, 1, \ldots, \tilde{\mathrm{Y}}$. The gradient $\nabla_{\tilde{\mathcal{Z}}} J(\cdot)$ is defined as follows:

$$\nabla_{\tilde{\mathcal{Z}}} J(\cdot) = \begin{pmatrix} \frac{\partial J(\cdot)}{\partial \tilde{z}_0} & \frac{\partial J(\cdot)}{\partial \tilde{z}_1} & \cdots & \frac{\partial J(\cdot)}{\partial \tilde{z}_{\tilde{\mathrm{Y}}}} \end{pmatrix}^{\mathsf{T}} = \sum_{l=0}^{L} \nabla_{\tilde{\mathcal{Z}}} J_l(\cdot). \tag{7.2.1}$$

Arguments are omitted for better readability. Similarly, the Jacobian matrices of the inequality and equality constraints, $\mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot) \in \mathbb{R}^{R \times \tilde{\mathrm{Y}}}$ and $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot) \in \mathbb{R}^{S \times \tilde{\mathrm{Y}}}$ respectively, are defined as follows:

$$\mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot) = \begin{pmatrix} \big(\nabla_{\tilde{\mathcal{Z}}} g_1(\cdot)\big)^{\mathsf{T}} \\ \big(\nabla_{\tilde{\mathcal{Z}}} g_2(\cdot)\big)^{\mathsf{T}} \\ \vdots \\ \big(\nabla_{\tilde{\mathcal{Z}}} g_R(\cdot)\big)^{\mathsf{T}} \end{pmatrix}, \quad \mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot) = \begin{pmatrix} \big(\nabla_{\tilde{\mathcal{Z}}} h_1(\cdot)\big)^{\mathsf{T}} \\ \big(\nabla_{\tilde{\mathcal{Z}}} h_2(\cdot)\big)^{\mathsf{T}} \\ \vdots \\ \big(\nabla_{\tilde{\mathcal{Z}}} h_S(\cdot)\big)^{\mathsf{T}} \end{pmatrix}. \tag{7.2.2}$$

Now recall that the number of optimization parameters that affect for example $g_r(\cdot)$ is defined by $|\mathcal{Z}_r^g|$. Accordingly, the number of non-fixed parameters is defined as $|\tilde{\mathcal{Z}}_r^g|$. Hence the number of structured zeros in the gradient $\nabla_{\tilde{\mathcal{Z}}} g_r(\cdot)$ is given by $\tilde{\mathrm{Y}} - |\tilde{\mathcal{Z}}_r^g|$. So by computing the gradient of $g_r(\cdot)$ only with respect to the non-fixed parameters respectively vertices that are connected by the edge e_r^{g} associated with $g_r(\cdot)$ avoids any evaluation that results in structured zeros. From an implementation point of view, the structured non-zeros are directly mapped into the overall pre-allocated memory for Jacobian $\mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$. The Jacobian $\mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$ is sparse if the majority of elements are structured zeros. The same applies to $h_s(\cdot)$ and the Jacobian $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$. Also for the gradient computation of the cost function $\nabla_{\tilde{\mathcal{Z}}} J(\cdot)$ only terms are evaluated for parameters z_v that actually contribute to $J(\cdot)$. In case multiple terms are combined into a single edge, the computation of the underlying gradients corresponds to the computation of the dense Jacobian of that edge. Note, upper and lower bounds on the optimization parameters in (7.1.1) are not included in $g(\cdot)$ since its non-zero derivatives can be trivially obtained by $\partial\big(z_{\min} - z_v\big)/\partial z_v = -1$ and $\partial\big(-z_v - z_{\max}\big)/\partial z_v = 1$ and consequently the (sparse) Jacobian matrices for all lower and upper bounds combined are just the identity respectively negative identity matrix.

Example 7.2.1 (Derivative Computation)**.** Consider again the Example 7.1.1 with the hypergraph as shown in Figure 7.1. The gradient of the cost function is obtained by:

$$\nabla_{\tilde{\mathcal{Z}}} J(\cdot) = \underbrace{\begin{pmatrix} \frac{\partial J_1(\cdot)}{\partial \tilde{z}_0} = 2\tilde{z}_0 \\ 0 \end{pmatrix}}_{\text{Edge } e_1^{\mathrm{J}}} + \underbrace{\begin{pmatrix} 0 \\ \frac{\partial J_2(\cdot)}{\partial \tilde{z}_1} = 2\tilde{z}_1 \end{pmatrix}}_{\text{Edge } e_2^{\mathrm{J}}} + \underbrace{\begin{pmatrix} 0 \\ \frac{\partial J_3(\cdot)}{\partial \tilde{z}_1} = 2\big(\tilde{z}_1 + \tilde{z}_2\big) \end{pmatrix}}_{\text{Edge } e_3^{\mathrm{J}}}. \tag{7.2.3}$$

Algorithm 7.1.: Compute derivatives from edge iterations.

1: **procedure** COMPUTEFIRSTORDERDERIVATIVESEB($\mathcal{V}, \mathcal{E}, \nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$)
2: **for all** edges $e \in \mathcal{E}$ **do**
3: **for** each connected unfixed vertex v **do**
4: Compute gradient of the associated function $J_l(\cdot)$, $g_r(\cdot)$ or $h_s(\cdot)$ with respect to the optimization parameter of vertex v ▷ Central differences (7.2.5)
5: Write values to proper row and column positions in $\nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$ or $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$
6: **return** $\nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$ or $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$

All zeros are also structured zeros and are not evaluated by edges e_1^{J}, e_2^{J}, and e_3^{J}. The Jacobian of the inequality constraints is:

$$\mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot) = \begin{pmatrix} \frac{\partial g_1(\cdot)}{\partial \tilde{z}_0} = -2\tilde{z}_0 & 0 \\ 0 & \frac{\partial g_2(\cdot)}{\partial \tilde{z}_1} = 1 \end{pmatrix}. \tag{7.2.4}$$

Note, due to $z_2 = 1$ vertex v_2 is fixed and hence not subject to optimization ($z_2 \notin \tilde{\mathcal{Z}}$), the computation of a possible third row for z_2 in (7.2.3) and the third column in (7.2.4) are omitted.

The previous description assumes that the derivative information of each edge with respect to its vertices is available. Even though it is possible to provide arbitrary edges with analytic derivatives, automatic computation relies on finite differences. According to the remarks in the introduction of this chapter, central differences are utilized such that the partial derivative of a function, for example, $J_l(\cdot)$ with respect to $\tilde{z}_{\tilde{v}}$, is approximated by

$$\frac{\partial J_l(\cdot)}{\partial \tilde{z}_{\tilde{v}}} \approx \frac{1}{2\delta_{\mathrm{D}}} \Big(J_l\big(\tilde{z}_1, \ldots, \tilde{z}_{\tilde{v}} + \delta_{\mathrm{D}}, \ldots, \tilde{z}_{\tilde{\mathrm{Y}}}\big) - J_l\big(\tilde{z}_1, \ldots, \tilde{z}_{\tilde{v}} - \delta_{\mathrm{D}}, \ldots, \tilde{z}_{\tilde{\mathrm{Y}}}\big) \Big). \tag{7.2.5}$$

The same applies to partial derivatives of $g_r(\cdot)$ and $h_s(\cdot)$. A general purpose choice for the discretization width δ_{D} on computer architectures with double precision is $\delta_{\mathrm{D}} = 10^{-9}$ which is also used in [Küm+11].
In the following, two different strategies for traversing the hypergraph are presented to numerically calculate the cost gradient $\nabla_{\tilde{\mathcal{Z}}} J(\cdot)$ and constraint Jacobian matrices $\mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$ and $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$. In practice, however, both strategies require a preparatory phase to automatically construct the graph and allocate the memory required for the data types. Note that for many sparse algebra routines it is typical to first analyze the structure of the problem. Later, only the actual values (structured non-zeros) are calculated for the previously determined structure. Accordingly, the preparatory phase of the hypergraph optimization links all vertices with a unique set of indices mapped to the cost gradient and constraint Jacobian columns. Edges are tagged with their equality and inequality Jacobian row indices. The derivatives $\nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$ and $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$ are allocated in a sparse matrix format, for example the compressed column/row format, or as triplet list as required for IPOPT. Hereby, the number and positions of non-zeros are easily obtained by iterating the edges of the graph and to keep track on the edge

Algorithm 7.2.: Compute derivatives from vertex iterations.

1: **procedure** computeFirstOrderDerivativesVB($\mathcal{V}, \mathcal{E}, \nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$)
2: **for all** unfixed vertices $v \in \mathcal{V}$ **do**
3: $z \leftarrow z + h$ ▷ Central differences step with $\delta_{\mathrm{D}} = 10^{-9}$, z is the optimization parameter of v
4: Evaluate and cache $J_l(\cdot)$, $g_r(\cdot)$ and $h_s(\cdot)$ of connected edges
5: $z \leftarrow z - 2h$ ▷ Central differences backward step
6: Evaluate $J_l(\cdot)$, $g_r(\cdot)$ and $h_s(\cdot)$ of connected edges
7: Apply central differences formula and write values to proper positions in $\nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$ or $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$
8: $z \leftarrow z + h$ ▷ Revert parameter to original value
9: **return** $\nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$

row indices and the column indices of the connected vertices. The preparation phase must be invoked whenever the structure of the nonlinear program (7.1.1) changes.

The second phase is the actual solution phase which computes new derivative values based on the previously determined structure. Algorithm 7.1 presents the first strategy which iterates the set of edges $\mathcal{E}$ and computes the gradient of the associated cost respectively constraint term with respect to every adjacent vertex (line 4). Hereby, the local dense gradient vectors are written directly into the global data structures for $\nabla_{\tilde{\mathcal{Z}}} J(\cdot), \mathrm{D}_{\tilde{\mathcal{Z}}} g(\cdot)$ and $\mathrm{D}_{\tilde{\mathcal{Z}}} h(\cdot)$ (line 5). The edge-based strategy facilitates the incorporation of user-defined block Jacobian matrices rather than triggering central differences for computing edge gradients respectively Jacobian matrices in line 4.

On the other hand, the vertex-based strategy in Algorithm 7.2 requires the vertices to maintain a list of connected edges. Primarily, central differences are directly applied to the current vertex. For every modification of a vertex value, all connected edges (to which the current vertex contributes to) are evaluated for the current cost or constraint value (line 4 and 6). Similar to the edge-based strategy, the local derivative information is written directly into the global data structures in line 6. The incorporation of user-defined derivatives for edges requires additional logic to bypass the central difference formula. On the other hand, the vertex-based strategy facilitates the parallel processing of connected edges in line 4 and 6 which might be advantageous in case the evaluation is computationally demanding.

Remark 7.2.1. The Hessian is computed similarly by applying forward differences to the resulting gradients respectively block Jacobians, in particular $\nabla^2_{z_v z_v} J = \big(\nabla_{z_v} J(z_v) + + \nabla_{z_v} J(z_v + \delta_{\mathrm{H}}) \big) / \delta_{\mathrm{H}}$ for the scalar case. The pertubation is set to $\delta_{\mathrm{H}} = 1 \cdot 10^{-2}$. In the non-scalar case of constraints, gradients are replaced by Jacobian matrices which are accumulated row by row (and optionally directly multiplied by Lagrange multipliers to calculate the Hessian of the Lagrangian). Only the upper or lower triangular part of the Hessian has to be calculated which significantly decreases the computation time. Note that the first-order derivative $\nabla_{z_v} J(z_v)$ can be reused to calculate the Hessian.

7.3. Application to Time-Optimal Model Predictive Control

In the following, time-optimal control problems covered in the previous chapters are defined in relation to the hypergraph and the corresponding sparsity patterns are illustrated.
A comparative analysis with AD (CasADi) and the dense approach for conventional MPC is contained in Appendix D. It is important to note that both the preparation time and the actual solution time are evaluated. The preparation time for the hypergraph includes the algorithmic connection of vertices and edges as well as the identification of indices for the entire Jacobian and Hessian matrices. In addition, AD requires a preparation in which the underlying function tree is constructed and elementary derivatives are evaluated. The main benefits and results are summarized below:

- Both AD and the hypergraph outperform the dense approach even for small problems sizes.
- The computational effort of the hypergraph in the solving phase is inferior but comparable to AD.
- AD requires a considerable computational overhead for the preparation phase which is negligible for the hypergraph.

To conclude, since the time-optimal MPC formulations include grid adaptation schemes, so that the nonlinear program dimensions vary during closed-loop control, the hypergraph is ideal as the total solution time (preparation and solving phase) is significantly lower (see Appendix D).
Switching from the nominal nonlinear program (7.1.1) to nonlinear programs (4.1.8), (4.1.12), (4.1.17) and the corresponding local uniform counterparts is easily achieved by replacing the nominal optimization parameters z_v with the corresponding state, control and time interval parameters. For simplicity, the additional constraints for states are assumed to be either empty or box constraints, especially $\mathbb{X} = \{x \in \mathcal{X} \mid x_{\min} \leq x \leq x_{\max}\}$. The control constraints which are necessary for time-optimal control tasks are also defined by upper and lower limits, in particular $\mathbb{U} = \{u \in \mathcal{U} \mid u_{\min} \leq u \leq u_{\max}\}$. Note that if additional conditions for states and controls are desired, these can be added as additional constraints to the optimization problem respectively as edges in the hypergraph.
The proposed hypergraph formulation is seamlessly integrated into the previous closed-loop control algorithms with grid adaptation. It is important to note that each time the structure of the nonlinear program changes, for example when the grid is adapted, the preparation phase must be called again as mentioned in Section 7.2.

Remark 7.3.1. In this thesis, the order of optimization variables in the Jacobians and Hessians is sorted on the basis of occurrence in relation to time. For example in case of collocation via finite differences $u_0, \Delta t_0, x_1, u_1, \Delta t_1, \ldots, u_{N-1}$ for the local grid and $u_0, x_1, u_1, \ldots, u_{N-1}, \Delta t$ for the global grid. This choice leads to an almost block diagonal matrix as also indicated in [WB08; WB10]. It would be possible to rearrange the order of variables and evaluate the performance properties. For instance, many solvers

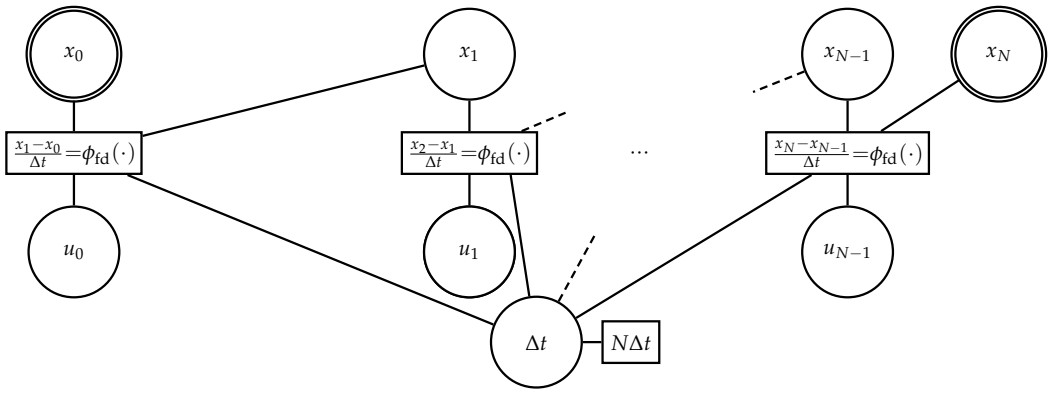


Figure 7.2.: Hypergraph of collocation via finite differences and the global uniform grid

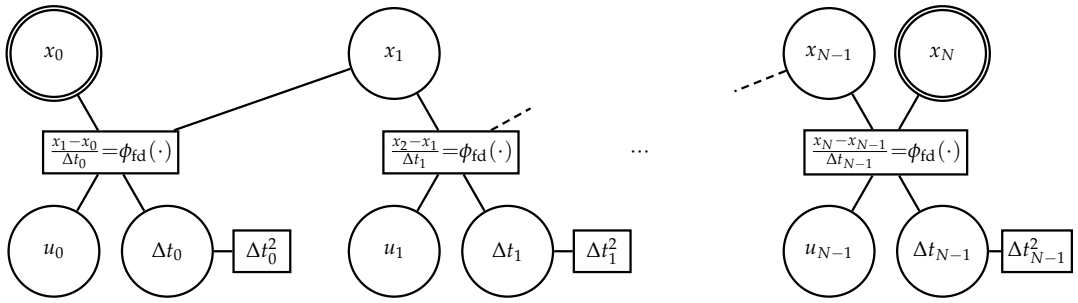


Figure 7.3.: Hypergraph of collocation via finite differences and the (quasi-)uniform grid

perform a Cholesky factorization to solve the underlying linear systems, so the number of non-zeros in the Cholesky factor depends on the original structure. However, since many efficient solvers also implement automatic block ordering techniques and results are solver-specific, the analysis of the respective order is not in the scope of this thesis.

7.3.1. Hypergraph for Collocation via Finite Differences

Figure 7.2 shows the hypergraph of the uniform grid formulations with a global time interval as introduced in Chapter 4. Accordingly, the hypergraph for the quasi-uniform grid with local time intervals from Chapter 6 is depicted in Figure 7.3. Obviously, the global time interval formulation exhibits fewer optimization parameters but on the other hand, the vertex of the global time interval is connected with all system dynamic constraints. The hypergraph enables an intuitive way to verify the sparsity pattern while designing the optimization problem. For example, Δt in the global uniform grid case is connected with all system dynamic edges. Consequently, a dense column is expected in the Jacobian of the equality constraints as well as a dense row and column pair in the related Hessian. For the local uniform grid (uniformity enforced), additional edges for $\Delta t_k = \Delta t_{k+1}$ are required as shown for multiple shooting below.

7.3.2. Hypergraph for Multiple Shooting

Multiple shooting with $N = M$ exhibits the same sparsity pattern as for collocation via finite differences. For $M < N$, the number of optimization variables decreases but the

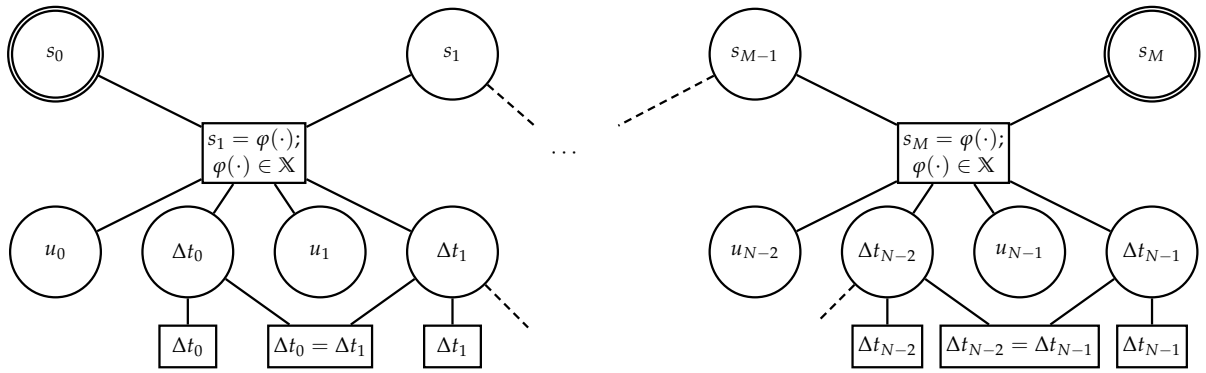
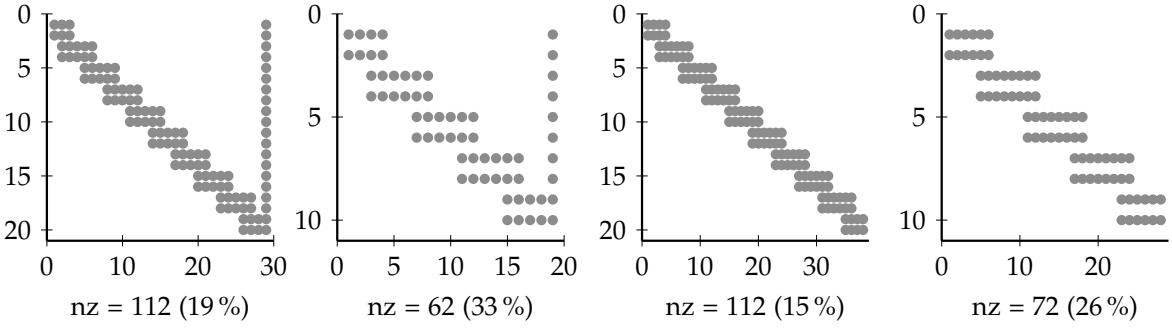


Figure 7.4.: Hypergraph of the multiple shooting realization with the local uniform grid, N even and $M = N/2$.

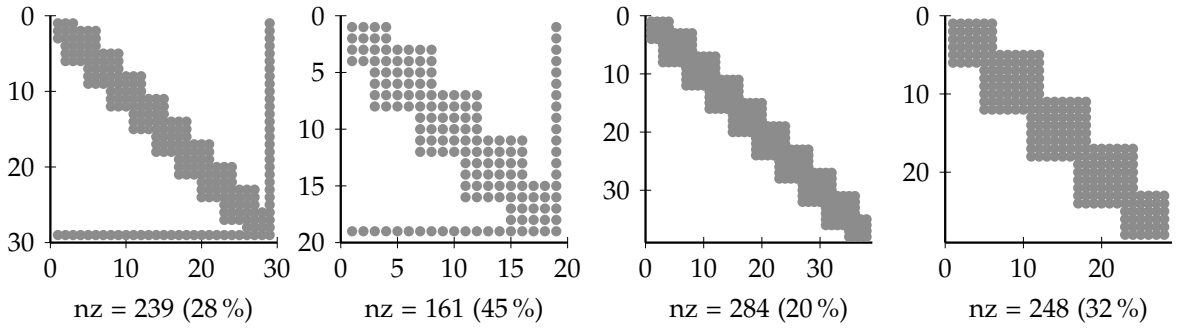


(a) Finite differences (b) Multiple shooting (c) Finite differences (d) Multiple shooting

Figure 7.5.: Equality Jacobian for the Van der Pol oscillator with $N = 10$. Sparsity patterns for the global uniform grid (a, b) and for the quasi-uniform grid (c, d). Multiple shooting is configured with $M = N/2$. The number of non-zeros is denoted by nz.

resulting sparsity pattern is less sparse, since multiple controls and time intervals are connected with a single edge. Figure 7.4 shows the hypergraph for the local uniform grid, $M = N/2$ and N even. Note, the shooting deflection constraint $s_{i+1} = \varphi(\cdot)$ and the constraint function $\varphi(\cdot) \in \mathbb{X}$ share the same edge as they depend on a similar parameter set.

Example 7.3.1 (Sparsity Patterns for Collocation via Finite Differences and Multiple Shooting)**.** This example shows the sparsity patterns for particular optimal control formulations. Consider the Van der Pol oscillator with $N = 10$ and only simple box constraints (these are not visible in the hypergraphs as they are cached in the vertices). Figure 7.5a and Figure 7.5b indicate which entries in the Jacobian of the equality constraints are non-zero. Multiple shooting is configured with $M = N/2$ and $\tilde{m}_i = 2, i = 0, 1, \ldots, M-1$. The order of optimization variables (columns) is chosen according to Remark 7.3.1. Obviously, the matrices belong to the global uniform grid due to the dense last column. The Jacobian for multiple shooting is smaller than for finite difference collocation due to $M < N$. On the other hand, the percent-wise number of non-zeros is quite larger. Similar observations are made for the quasi-uniform grid in Figure 7.5c and Figure 7.5d. These matrices are also larger in comparison to their global counterpart. However, their structure is block-diagonal and more sparse.



(a) Finite differences (b) Multiple shooting (c) Finite differences (d) Multiple shooting

Figure 7.6.: Hessian of the Lagrangian for the Van der Pol oscillator with $N = 10$. Sparsity patterns for the global uniform grid (a, b) and for the quasi-uniform grid (c, d). Multiple shooting is configured with $M = N/2$. The number of non-zeros is denoted by nz.

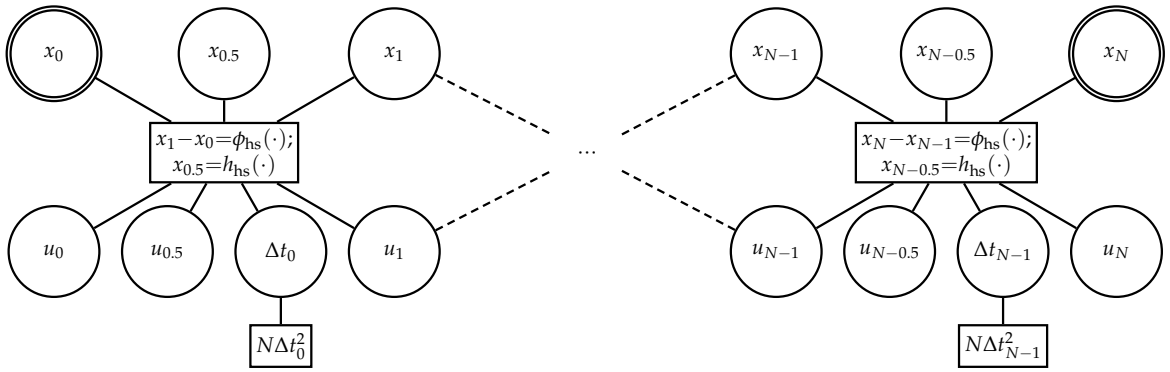


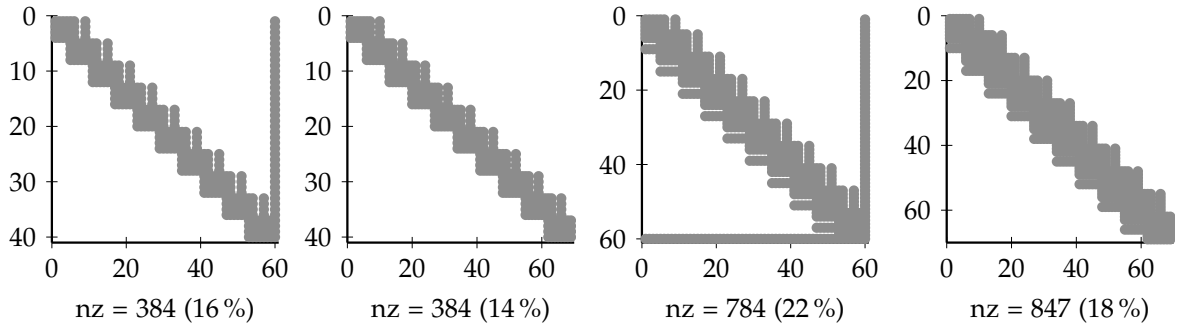
Figure 7.7.: Hypergraph of uncompressed Hermite-Simpson collocation with local temporal discretization grid (quasi-uniform)

Figure 7.6 shows the sparsity patterns of the Hessian of the Lagrangian for the very same settings and the same observations are made.

7.3.3. Hypergraph for Hermite-Simpson Collocation

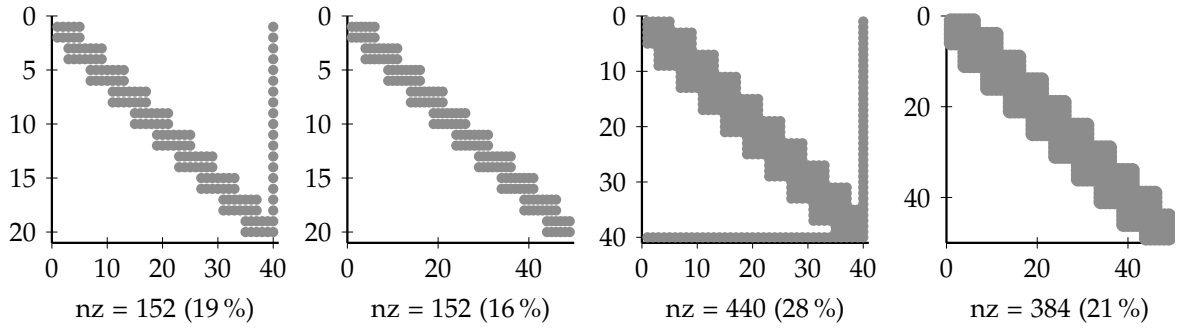
The complexity of the hypergraph for Hermite-Simpson collocation highly depends on the chosen form (uncompressed or compressed) and the control representation. Figure 7.7 depicts the hypergraph for uncompressed Hermite-Simpson collocation with controls at midpoints (quadratic and linear spline representation) and the quasi-uniform grid. Constructing the hypergraphs for the other formulations is analogue. The structure in Figure 7.7 shows that the corresponding Hessian and Jacobian consists of relatively large block matrices on the diagonal. This is further detailed in the next example.

Example 7.3.2 (Hermite-Simpson Sparsity Pattern)**.** The optimal control setup is similar to the previous Example 7.3.1. Figure 7.8 shows the Jacobian and Hessian matrices for uncompressed Hermite-Simpson collocation including control midpoints. Sparsity structures for both the global and the quasi-uniform grid are shown. In addition,



(a) Jacobian (global) (b) Jacobian (local) (c) Hessian (global) (d) Hessian (local)

Figure 7.8.: Equality Jacobian and Hessian of the Lagrangian for the Van der Pol oscillator with *uncompressed* Hermite-Simpson collocation and $N = 10$: Sparsity patterns for the global uniform grid (a, c) and for the quasi-uniform grid (b, d). The number of non-zeros is denoted by nz.



(a) Jacobian (global) (b) Jacobian (local) (c) Hessian (global) (d) Hessian (local)

Figure 7.9.: Equality Jacobian and Hessian of the Lagrangian for the Van der Pol oscillator with *compressed* Hermite-Simpson collocation and $N = 10$. Sparsity patterns for the global uniform grid (a, c) and for the quasi-uniform grid (b, d). The number of non-zeros is denoted by nz.

Figure 7.9 depicts the sparsity patterns for the compressed form. In all cases, the number of non-zeros and problem dimension for the uncompressed form are much greater than for the compressed form. On the other hand, the growth of sparsity is comparatively small.

8

Comparative Analysis and Benchmark Results

The previous chapters introduced three different types of discretization grids, namely the global and local uniform grid as well as the quasi-uniform grid. Several direct transcription methods are applied to each grid. The formulation of the associated optimal control problems as hypergraph, as described in Chapter 7, enables the exploitation of structural sparsity and thus an efficient computation of derivatives.
This chapter begins with a comprehensive benchmark between the different methods of variable discretization grids and time transformation. Details on the benchmark hardware are provided in Appendix F.2.1 and for the software packages in Appendix E. The second part compares the variable discretization grid with TOMPC and the approach based on the ℓ_1-norm cost with respect to their open-loop performance. Finally, this chapter examines the closed-loop performance on the *ECP Model 220*.

8.1. Time-Optimal Control with Variable Discretization Grids

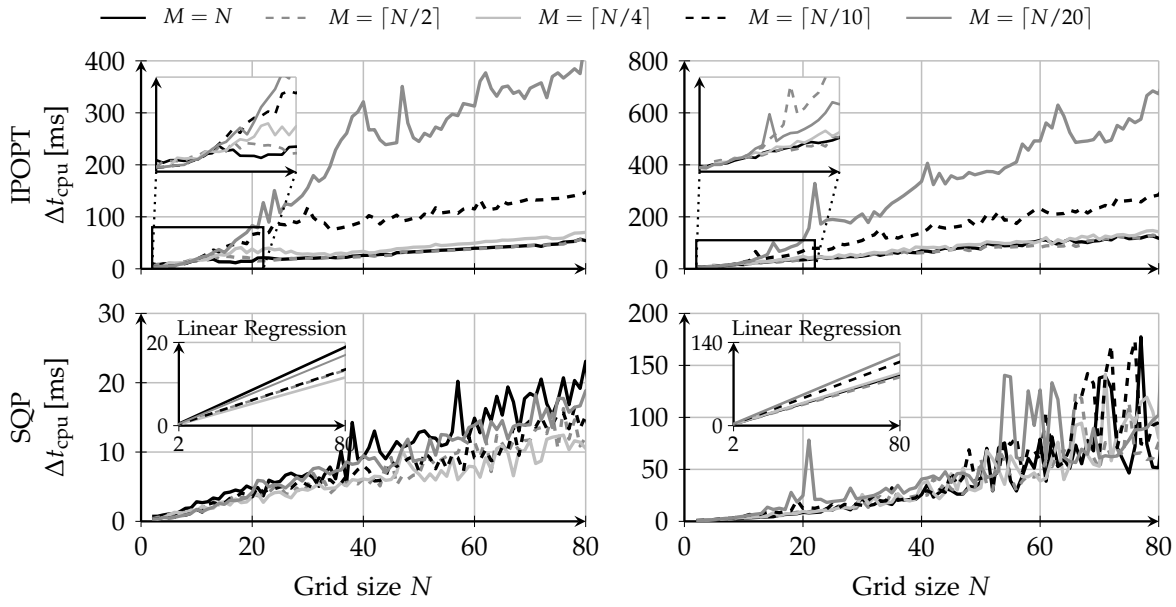
The computation time and accuracy of the solution to the initial optimal control problem are decisive, since these values serve as an upper bound for the realization of closed-loop control. Note that the very first solution assumes a generic initial guess such as a straight line in the state space and zero controls. In subsequent sampling intervals, the solution is warm-started, so that lower computation times and better accuracy can be expected.
The number of different configurations, including the selected grid, the transcription method and the solver implementation, is large and it is often unclear how they behave in competition. The scope of this thesis limits the number of benchmarks, but attempts are being made to gain meaningful insights into the characteristics of the individual combinations and their possible uses. It is well-known that the dynamics approximation and thus direct transcription methods strongly depend on the individual system properties [Bet10; Kel17]. Although the benchmark systems selected in this thesis are chosen for various reasons (see Section 3.3), their complexity in terms of system approximation differs only slightly. The presentation of the results is structured as follows:

- Comparison of transcription methods.
- Comparison of grids (local/global uniform, quasi-uniform).
- Performance implications based on the selected solver.

For benchmarks with the Van der Pol oscillator, standard constraint sets according to Section 3.3.1 and the transition from the origin to $x_\mathrm{f} = (0.8, 0)^\intercal$ are considered. The rocket system with its default constraint sets (see Section 3.3.2) is configured for the transition from the origin to $x_\mathrm{f} = (10, 0, 0.5)^\intercal$. Note that the last state component is unfixed during optimization and value 0.5 is used as initialization. Initialization in general is carried out according to Section 5.1. Particularly, the state trajectory is initialized as straight line between start and final state and the controls are set to zero. To maintain a grid size independent initialization with respect to time, individual time intervals are initialized according to $\Delta t_k = 3\,\mathrm{s}/N$. The duration of 3 s overestimates the final time of the Van der Pol oscillator and underestimates the final time for the rocket system. The calculation of the integral error $e_{\hat{x}}(t_\mathrm{f})$ according to Section 3.2.2 takes the optimal references from Example 4.3.1 for the Van der Pol oscillator and from Example 6.2.1 for the rocket system into account. Computation times are represented by the median obtained from 20 repetitions on the benchmark hardware (refer to Appendix F.2.1 for details).
The benchmarks are performed with three different solver configurations. IPOPT constitutes a well-established sparse and efficient general purpose solver based on the interior-point method. First-order derivatives are computed for all benchmarks using the hypergraph. Second-order derivatives are either approximated with IPOPT's limited memory BFGS implementation or also computed using the hypergraph. The first configuration is referred to as IPOPT with explicit Hessian and the second as IPOPT with BFGS. Appendix E.2.1 lists the specific IPOPT solver settings. In addition, a sparse SQP solver constitutes the third configuration and its implementation is described in Appendix E.2.2. The underlying quadratic program solver is OSQP (**O**perator **S**plitting **Q**uadratic **P**rogram) [Ste+17], a recently published sparse general purpose solver based on the alternating direction method of multipliers. Appendix E.2.2 summarizes related SQP and OSQP parameters. Unless otherwise specified, all optimizations are performed until convergence with respect to first-order conditions.

Remark 8.1.1. The benchmarks do not aim to provide general performance statements on interior-point versus SQP methods. Practicable nonlinear program solvers implement many heuristics. Especially the selection of the line search strategy, the merit function and second-order correction is crucial. The dedicated SQP implementation uses the Hessian of the cost function to enforce positive semidefiniteness for the time-optimal nonlinear programs while IPOPT takes the Hessian of the Lagrangian into account. The latter requires considerably more computing power. The benchmarks consider IPOPT as it is, and additionally show what can be expected with a custom sparse SQP implementation.

The following benchmarks analyze computation time and accuracy for increasing grid sizes N. The interval of interest is chosen as $2 \leq N \leq 80$ to take into account the usual horizon lengths that can be realized in practice today under real-time conditions.



(a) Van der Pol oscillator

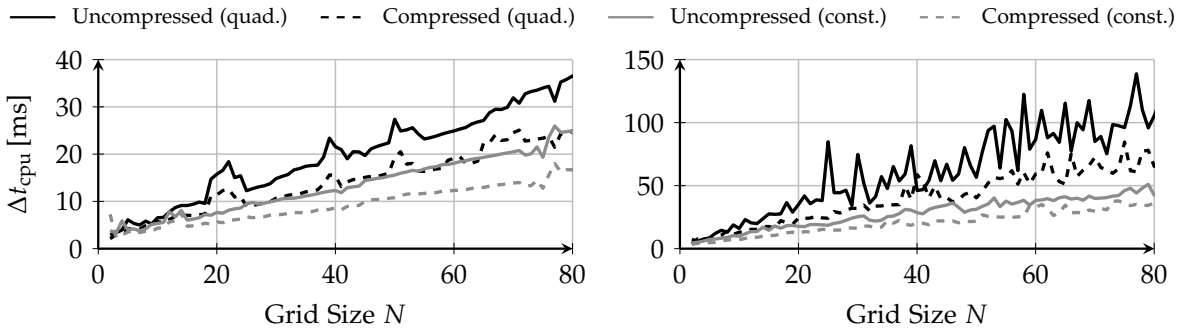
(b) Rocket system

Figure 8.1.: Benchmark results for multiple shooting with varying shooting grid sizes M and the global uniform grid. Initial value problems are solved with 5$^{\text{th}}$-order Runge-Kutta. IPOPT with explicit Hessian computation is utilized at the top and SQP at the bottom.

8.1.1. Performance Comparison between Transcription Methods

The first analysis investigates computation times of multiple shooting for varying shooting grid sizes M. Since the size N of the variable discretization grid increases, the shooting grid (4.1.9) is defined with $\tilde{m}_i := \tilde{m} \in \mathbb{N}$ regular control grid partitions (4.1.2) per shooting interval for $i = 0, 1, \ldots, M-1$. If $N/\tilde{m}$ is not a positive integer, the last interval is filled with the remaining number of controls to match N properly. Therefore, the number of intervals is $M = \lceil N/\tilde{m} \rceil$ in which $\lceil \cdot \rceil$ denotes the ceiling operator.

Figure 8.1 shows the benchmark results for several choices of $\tilde{m}$ respectively M. Initial value problems are solved with 5$^{\text{th}}$-order Runge-Kutta to load the solvers with non-negligible integration times. The results at the top are created with IPOPT and the explicit Hessian computation. Small shooting intervals, in particular $\tilde{m} = 1$ and $\tilde{m} = 2$ perform similarly and are faster than larger shooting intervals. Case $\tilde{m} = 4$ is also very similar, but already inferior for the Van der Pol oscillator. Note that a small $\tilde{m}$ and thus a large M leads to a large number of parameters that must be optimized, but to a sparser structure. SQP is not as sensitive as IPOPT with explicit Hessian (see the bottom plots in Figure 8.1). The computation times are generally shorter and more similar. Their linear trends (regression) show the best results for $\tilde{m} = 4$ for the Van der Pol oscillator and $\tilde{m} = 2$ for the rocket system. Note, the SQP implementation does not compute the Hessian of the deflection constraints compared to IPOPT which avoids many shooting evaluations. These results correspond to the literature, since there is always a compromise between the number of parameters and the sparsity structure [Raw+17; GP17]. The benchmarks show that the size of the shooting interval size should be chosen relatively small, for example $\tilde{m} = 1$ to $\tilde{m} = 4$ here.



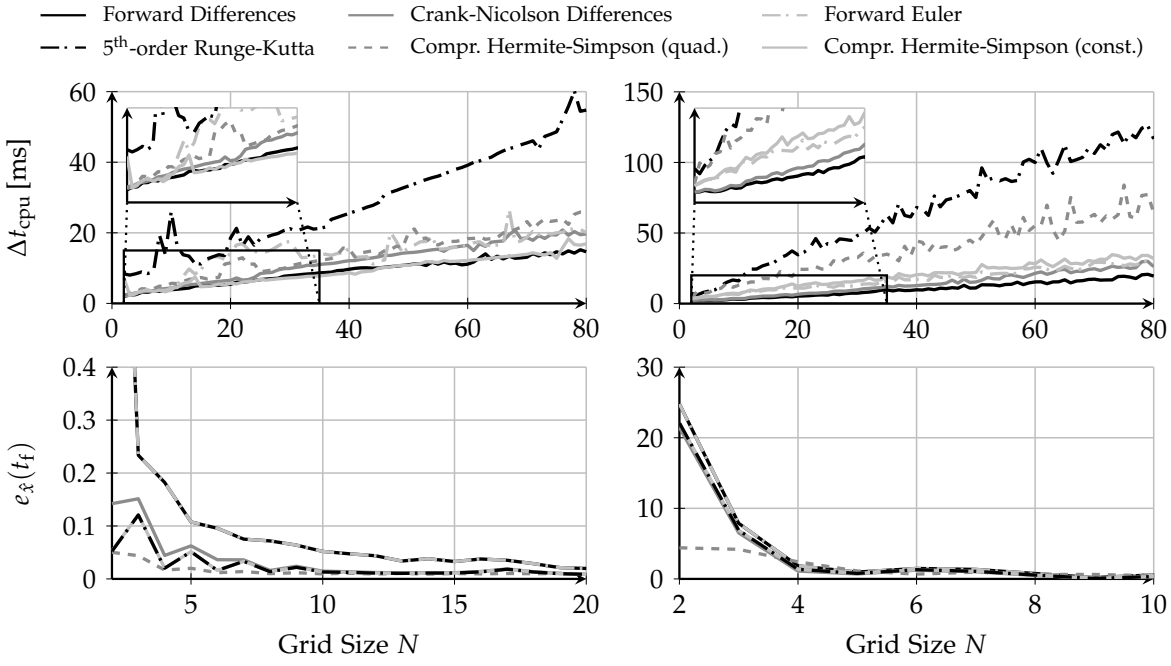
(a) Van der Pol oscillator

(b) Rocket system

Figure 8.2.: Computation times for selected Hermite-Simpson collocation variants and the global uniform grid. The solver is IPOPT with explicit Hessian computation.

The second evaluation considers Hermite-Simpson collocation with quadratic and constant control splines and either in compressed or uncompressed form as shown in Figure 8.2. The compressed form reveals much lower computation times for both quadratic and constant control splines which confirms the observation in Example 7.3.2. For generic optimal control problems with many parameters, [Kel17] prefers the uncompressed form, but the accuracy requirements and thus the problem sizes of individual open-loop control problems are smaller in MPC real-time realizations than in feedforward optimal control.
The variants with constant control splines are faster than their quadratic counterparts because the number of optimization parameters and degrees of freedom are reduced. Also from the time-optimal control perspective the constant control spline is better suited for bang-singular-bang control as discussed in Example 6.2.2.
A comparison between the various direct transcription methods is presented below. Figure 8.3 depicts several candidates for collocation via finite differences, multiple shooting and (compressed) Hermite-Simpson collocation. The grid is set to global uniform and the solver is IPOPT with exact Hessian computation. Forward differences and forward Euler are the fastest methods, but on the other hand, they lead to large integral errors, especially at low N. The integral error $e_{\hat{x}}(t_{\text{f}})$ for the Van der Pol oscillator at $N = 2$ is 1.39. Note that the error of forward Euler and forward differences is identical according to Remark 4.1.1. Concerning computation time, forward Euler is comparable with forward differences while for the rocket system forward differences are slightly faster. The latter applies to both benchmark systems in case of the local grid (refer to Appendix F.2.2). The higher-order transcription variants with Runge-Kutta and Hermite-Simpson collocation result in lower errors with respect to N.

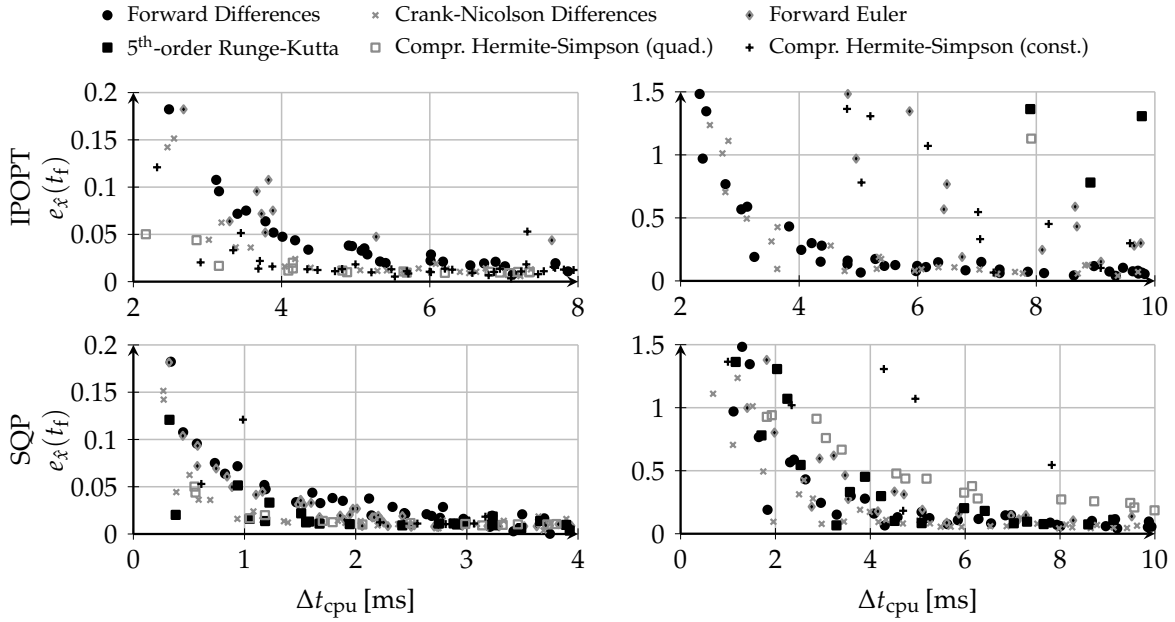
Remark 8.1.2. The results also indicate that the rocket system requires larger computation times than the Van der Pol oscillator, mainly due to the larger state space. On the other hand, the dynamics of the rocket system are simpler because the integral error converges for forward Euler and forward differences rapidly even for very small N. The different transcription methods perform similarly except Hermite-Simpson with quadratic splines which also leads to constraint violations as discussed in Example 4.3.2 and Example 6.2.2.



(a) Van der Pol oscillator (b) Rocket system

Figure 8.3.: Computation times and integral errors for several direct transcription methods and the global uniform grid. Multiple shooting is configured with $M = N$. The solver is IPOPT with explicit Hessian computation. The integral error for forward differences and forward Euler is by definition identical. Remember that Hermite-Simpson collocation with a quadratic control spline violates constraints.

To visualize the effectiveness of the transcription methods independent of the grid size N, Figure 8.4 shows a scatter plot with the integral error on the ordinate and the computation time on the abscissa. The samples furthest down on the left perform best because they produce a small integral error and require less computational burden. For the Van der Pol oscillator, higher-order schemes perform better, especially Hermite-Simpson in case of IPOPT and Runge-Kutta in case of SQP which confirms the observations in Remark 8.1.2. In contrast, forward differences are much better suited for the rocket system. Interestingly, Crank-Nicolson differences perform quite well both between systems and between solvers. Crank-Nicolson is an implicit second-order method and therefore less complex than Hermite-Simpson collocation and 5^{th}-order Runge-Kutta, but more accurate than forward differences. Therefore, it offers a reasonable compromise between dynamics accuracy and computation time for both benchmark systems. Similar results apply to the local grid as shown in Appendix F.2.2. The benchmark results indicate that for rather simple system dynamics it might be worth to select larger grid sizes and hence larger optimization parameters and on the other hand use simpler transcription schemes. For more complex systems it is still worth to consider higher-order schemes. Also note that stiff systems usually require implicit methods because they are numerically stable [SG79]. In addition, implicit methods are easily integrated by direct transcription since (Newton-type) nonlinear program solvers immediately process implicit equations.



(a) Van der Pol oscillator

(b) Rocket system

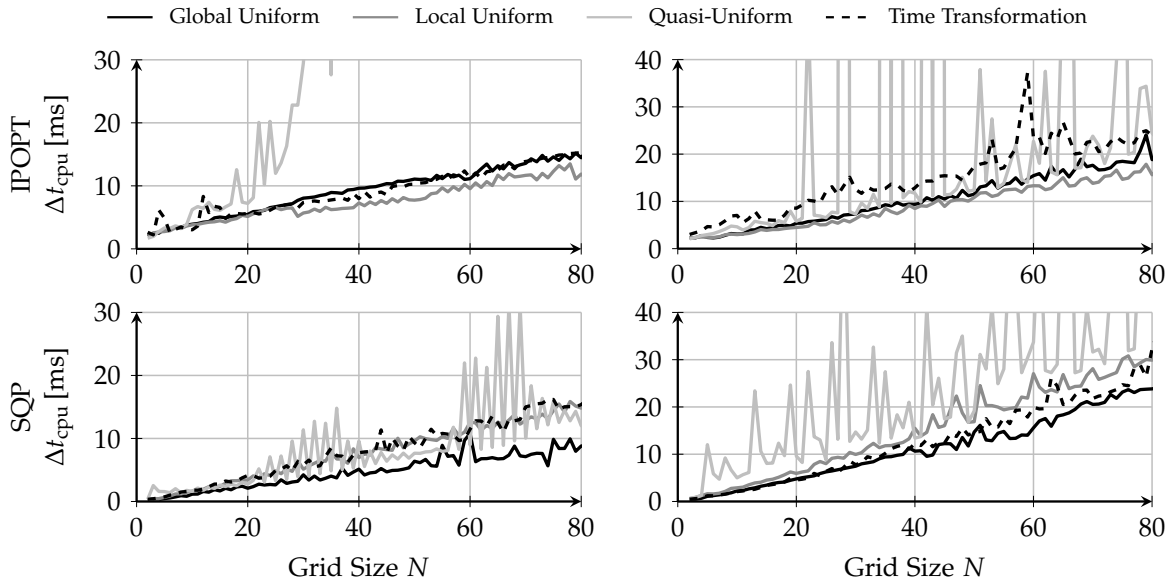
Figure 8.4.: Scatter plot of integral errors with respect to the computation time (global uniform grid). IPOPT with explicit Hessian computation is utilized at the top and SQP at the bottom. Multiple shooting is configured with $M = N$ for IPOPT and $M = \lceil N/4 \rceil$ for SQP.

8.1.2. Comparison of Discretization Grids

In the following, the necessary computation times for the various discretization grids are evaluated. Benchmarks apply to both forward Euler and forward differences because they have different numerical properties while sharing the same accuracy (see Remark 4.1.1).

Figure 8.5 shows the results for collocation via forward differences and Figure 8.6 for forward Euler. The comparative analysis includes results from the time transformation approach as summarized in Appendix C.1.1. In the case of forward differences, the quasi-uniform grid has considerable computation times and is very sensitive to the grid size N. The plots in Figure 8.5 are truncated to emphasize the difference between the other methods. The observations apply to both IPOPT and SQP, but are worse in the case of IPOPT. In these cases, both IPOPT and SQP require much more Newton and line search iterations until acceptable steps are found, however, the solvers converge sufficiently in all benchmarks. A more detailed convergence analysis regarding these observations is provided in Appendix F.2.4. The local uniform grid is best suited for IPOPT while the global grid is slightly better suited for SQP. In addition, these two configurations show lower computation times compared to the time transformation method.

The numerical effects of the quasi-uniform grid with finite differences do not apply to multiple shooting in general, especially to forward Euler as shown in Figure 8.6. In contrast, both the global uniform grid and time transformation show larger peaks at larger N for the rocket system with SQP. IPOPT instead performs comparable or inferior for the quasi-uniform grid. In the case of SQP, time transformation is faster



(a) Van der Pol oscillator

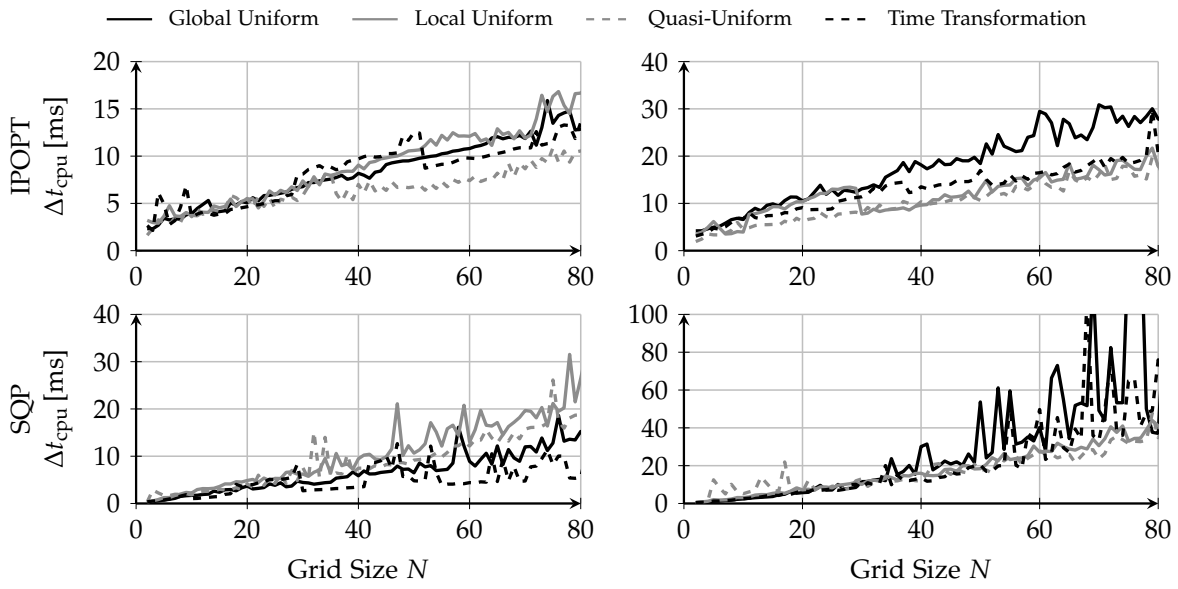
(b) Rocket system

Figure 8.5.: Comparison of different grids with forward differences. IPOPT with explicit Hessian computation is utilized at the top and SQP at the bottom.

for large N and the Van der Pol oscillator, while the quasi-uniform and local uniform grid perform well and exhibit fewer peaks in computation time for the rocket system. The benchmark results show that all grids perform well in some configurations and worse with others. It is therefore recommended to validate several grid variants for a particular application. The results show that the local uniform grid could serve as a general purpose grid, as it is more robust against peaks in the computation time and performs quite well in all scenarios.

8.1.3. Comparison between Solvers

This section compares the computation times of different solvers while other settings such as grid and transcription methods are fixed. Figure 8.7 shows the results for IPOPT with explicit Hessian, IPOPT with BFGS approximation and SQP. The two upper plots are generated with forward differences and the lower plots with forward Euler for the same reasons as in the previous section. IPOPT with explicit Hessian computation is comparable to the BFGS approximation with the exception of the rocket system and forward Euler. SQP outperforms IPOPT in all cases for small grid sizes N. In the case of the Van der Pol oscillator, SQP even outperforms IPOPT for the almost entire range of grid sizes ($N < 70$). In all other cases, IPOPT is faster than SQP for larger grid sizes. This is also observed for the results with respect to the local uniform grid in Figure 8.8. In addition, the explicit Hessian beats the BFGS method for finite differences and is comparable for forward Euler. Plots for the quasi-uniform grid are provided in Appendix F.2.3 which also show the computation time issues from the previous sections.



(a) Van der Pol oscillator

(b) Rocket system

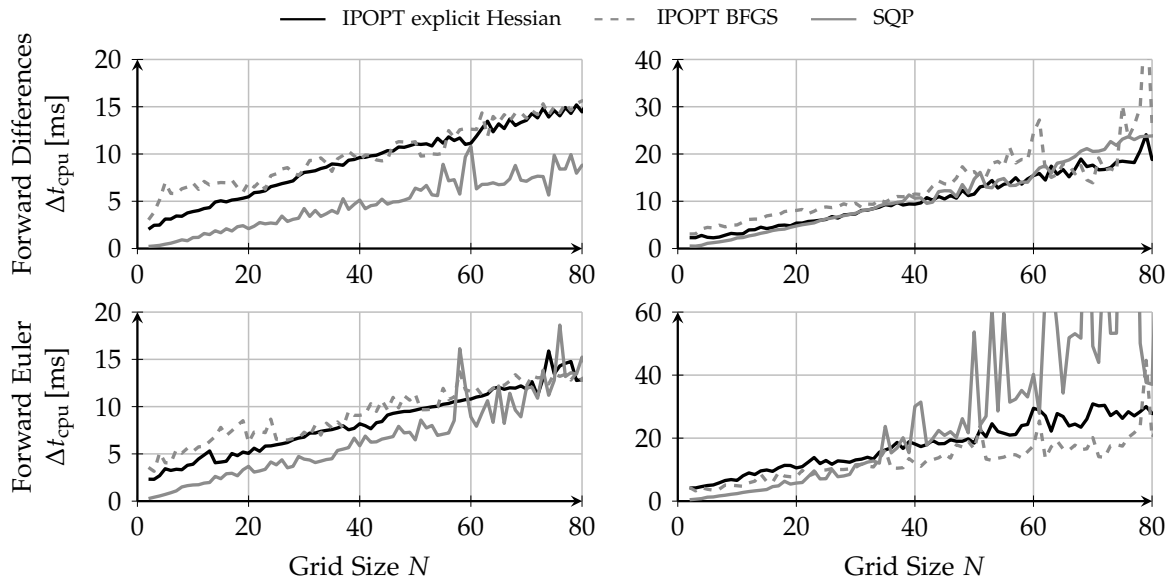
Figure 8.6.: Comparison of different grids with multiple shooting ($M = N$) and forward Euler. IPOPT with explicit Hessian computation is utilized at the top and SQP at the bottom.

8.1.4. Discussion

The benchmark results demonstrate that the use of the hyerpgraph leads to an almost linear complexity $\mathcal{O}(N)$, ignoring all peaks and excluding the quasi-uniform grid with forward differences. In contrast, the computation time in earlier work [Rös+15c] increases quadratically, which do not use the hypergraph (see also the results of the dense grid in Appendix D). Even if some grids are affected by large peaks, almost all curves are rather nonsmooth. It has to be considered that the choice of N influences the position of the switching points in the discretized control trajectory and thus even a small change of N has a significant impact on the underlying numerical properties. If a grid point does not match correctly, the switch in control applies to two consecutive intervals. A general overview and conclusion is later provided in Chapter 10. Note that the developed software package is versatile and generic, so the comparative analysis can be easily repeated for arbitrary dynamic systems (see Appendix E).

8.2. Comparison with Fixed Grid Methods

This section compares the variable discretization grids with state-of-the-art fixed grid methods for time-optimal MPC. These are TOMPC [Van+11a] and a stabilizing approach based on the ℓ_1-norm [Ver+17]. Appendix C describes both approaches in more detail. This section continues to focus on open-loop solutions while the next section deals with closed-loop realizations. The analysis takes into account the local uniform grid because its performance is almost comparable to the other grids and at the same time robust for computation times under all previous benchmarks. Figure 8.9 provides a first qualitative comparison for the open-loop optimal control of the rocket



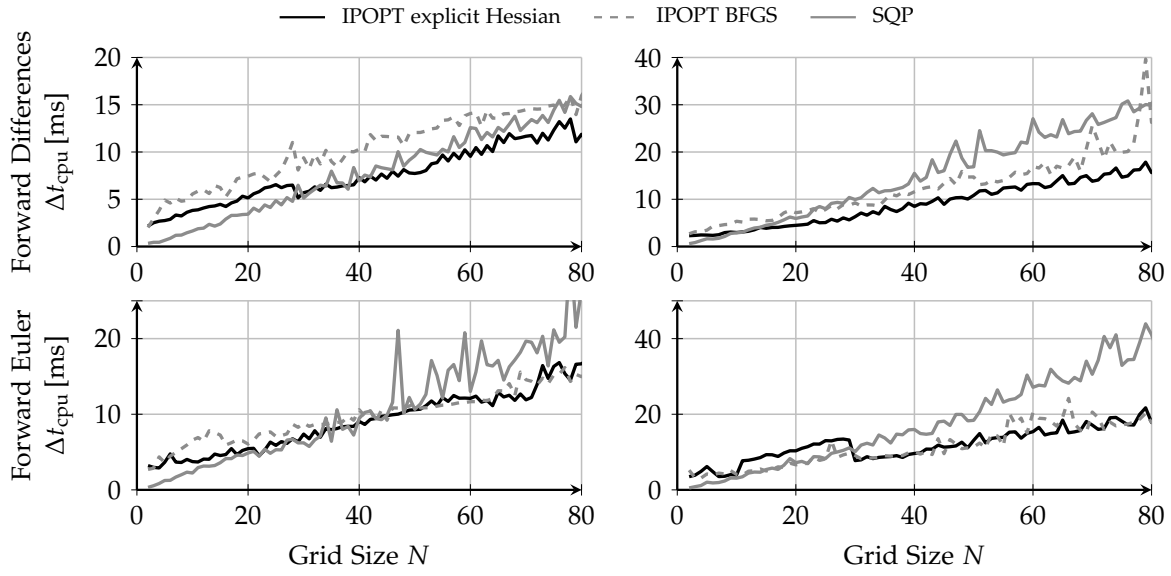
(a) Van der Pol oscillator

(b) Rocket system

Figure 8.7.: Comparison of the solver configurations on the global uniform grid. Forward differences are utilized at the top and forward Euler with $M = N$ at the bottom.

system. The task is similar to Example 6.2.1 and consist of controlling the system from $(0, 0, 1)^\intercal$ to $x_\text{f} = (s_{\text{r,f}} = 10, 0, \cdot)^\intercal$. Note that again the last state component is not fixed. The benchmark performs with collocation via forward differences. TOMPC and the ℓ_1-norm approach are configured with a fixed grid of temporal resolution $\Delta t = 0.1\,\text{s}$. Notice in Figure 8.9 that TOMPC adapts the horizon length N until the (quasi) minimum-time feasible solution is found (dead-beat control). The determined horizon length is $N = 76$ and the optimization result in Figure 8.9 is obviously not bang-bang and time-optimal. This is because $N = 76$ is the lowest integer for which $\Delta t = 0.1\,\text{s}$ still leads to a feasible solution. For discrete-time systems $N = 76$ corresponds to the minimum-time (dead-beat) solution, but is only a rough approximation to the actual continuous-time problem (3.2.2). A change of the horizon to $N = 75$ and a slight increase of the fixed interval length Δt leads to a better approximation. On the other hand, the local uniform grid with $N = 75$ identifies the optimal grid interval length as part of the optimization problem, that is $\Delta t^* = 0.100\,716\,2\,\text{s}$. Similar to TOMPC, the ℓ_1-norm approach requires at least a horizon length of $N = 76$ for the fixed interval $\Delta t = 0.1\,\text{s}$ to ensure feasibility with the underlying terminal equality condition. To emphasize a main advantage of the ℓ_1-norm approach, Figure 8.9 shows the optimal control trajectory for $N = 79$. The ℓ_1-norm approach is able to keep the system at x_f even in the open-loop solution. Note that the control switches to zero at approximately 7.5 s, allowing for a receding horizon implementation without grid adaption and thus fixed dimensions of the nonlinear program.

The following part presents a further comparative analysis concerning the computation times of the first optimal control problem for the rocket system configuration as before and the Van der Pol oscillator. The control task of the Van der Pol oscillator is configured according to Example 8.3, especially the transition from the origin to



(a) Van der Pol oscillator (b) Rocket system

Figure 8.8.: Comparison of the solver configurations on the local uniform grid. Forward differences are utilized at the top and forward Euler with $M = N$ at the bottom.

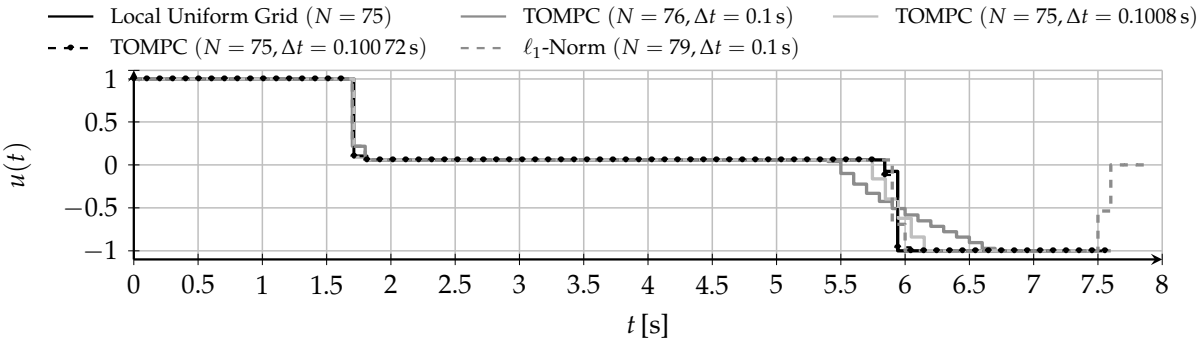


Figure 8.9.: Comparison of optimal control trajectories for the rocket system.

$x_\text{f} = (0.8, 0)^\intercal$ under control bounds $\mathbb{U} = \{u \in \mathbb{R} \mid |u| \leq 1\}$. The grid size for the Van der Pol oscillator which corresponds to $\Delta t = 0.1\,\text{s}$ is $N^* = 16$. Table 8.1 lists the medians of the computation times for both benchmark systems (20 repetitions). Optimizations are performed with IPOPT and the explicit computation of Hessians. Results for different values of θ are shown for the ℓ_1-norm approach. In theory, θ must be larger than a lower bound θ_0 to ensure time-optimality [Ver+17]. Checking the solution for $\theta = 1.01$ in Figure 8.9 confirms that the solution for the rocket system is time-optimal. Similarly, $\theta = 1.01$ applies for the Van der Pol oscillator. Choosing θ too large leads to fast growing values θ^k in the cost function which results in ill-conditioned problems especially for large N. Notice that for larger values of θ the computation times increase, although the optimization results respectively errors $e_{\hat{x}}(t_\text{f})$ remain unchanged. For the rocket system and $N = 100$ IPOPT terminates early with its internal local infeasibility detection and the resulting error is $e_{\hat{x}}(t_\text{f}) = 0.13$. The ℓ_1-norm approach also requires to transform the non-smooth problem into a smooth one by introducing further opti-

Table 8.1.: Computation times of the optimal control problems for Van der Pol oscillator and the rocket system with varying grid sizes respectively horizon lengths N. The ideal grid size which corresponds to a discretization width Δt of almost 0.1 s is indicated by N^*.

		$N = 5$		$N^* = 16$		$N = 25$		$N = 50$	
		Δt_{cpu} [ms]	$e_{\hat{x}}(t_{\text{f}})$	Δt_{cpu} [ms]	$e_{\hat{x}}(t_{\text{f}})$	Δt_{cpu} [ms]	$e_{\hat{x}}(t_{\text{f}})$	Δt_{cpu} [ms]	$e_{\hat{x}}(t_{\text{f}})$
Van der Pol	TOMPC	135.27	0.06	13.88	0.06	49.25	0.06	174.16	0.06
	ℓ_1-Norm ($\theta = 1.01$)	—	—	4.05	0.04	7.38	0.04	9.88	0.04
	ℓ_1-Norm ($\theta = 1.1$)	—	—	4.46	0.04	6.14	0.04	12.88	0.04
	ℓ_1-Norm ($\theta = 1.5$)	—	—	8.81	0.04	9.91	0.04	30.12	0.04
	Local Uniform Grid	3.18	0.07	3.52	0.04	4.45	0.02	7.58	≈ 0
		$N = 25$		$N = 50$		$N^* = 76$		$N = 100$	
Rocket system	TOMPC	3246.69	0.19	2155.71	0.19	151.52	0.19	909.84	0.19
	ℓ_1-Norm ($\theta = 1.01$)	—	—	—	—	38.01	0.09	47.73	0.09
	ℓ_1-Norm ($\theta = 1.1$)	—	—	—	—	757.60	0.09	101.51	0.13
	Local Uniform Grid	5.47	0.11	10.60	0.05	17.44	0.07	21.48	0.04

mization parameters and inequality constraints (refer to Appendix C). This additional computational overhead is also noticeably for growing N. For small problem sizes (Van der Pol oscillator), the ℓ_1-norm approach is comparable to the local uniform grid in terms of computation times. Note, the error $e_{\hat{x}}(t_{\text{f}})$ slightly differs for the ℓ_1-norm approach and the local uniform grid since the ℓ_1-norm approach already starts to switch the control to zero at t_{f}. As expected TOMPC has the lowest computation times at N^*, because TOMPC has to search for N^* in every other case with a linear search. The integral error is still larger compared to the other approaches according to the previously discussed integer optimization with fixed grid intervals. In addition, even if TOMPC starts at N^*, at least the solution for $N - 1$ is checked for feasibility, which results in additional computational overhead. The computation times of TMOPC are usually also larger due to the additional effort in calculating the quadratic form costs. The local uniform grid requires in all cases the lowest computation times and its dynamics accuracy scales with N. For $N < N^*$, the solution is suboptimal with respect to the dynamics approximation as the error $e_{\hat{x}}(t_{\text{f}})$ is larger. On the other hand, for $N > N^*$ the error can be further reduced while at the same time the computation times are kept low.

8.3. Closed-Loop Control Performance

This section evaluates the closed-loop control performance on the real *ECP industrial plant emulator 220* as introduced in Section 3.3.3. Both the local uniform grid and the ℓ_1-norm approach are subject to comparison. TOMPC is omitted because it is not real-time capable in a comparative setup to the other approaches. Furthermore, the evaluation includes the dual-mode and hybrid cost realization according to Section 5.3. The control design follows Example 5.3.1 and as already mentioned, the linearized system for the dual-mode controller does not change for varying angular positions $x_{\text{f},1}$.

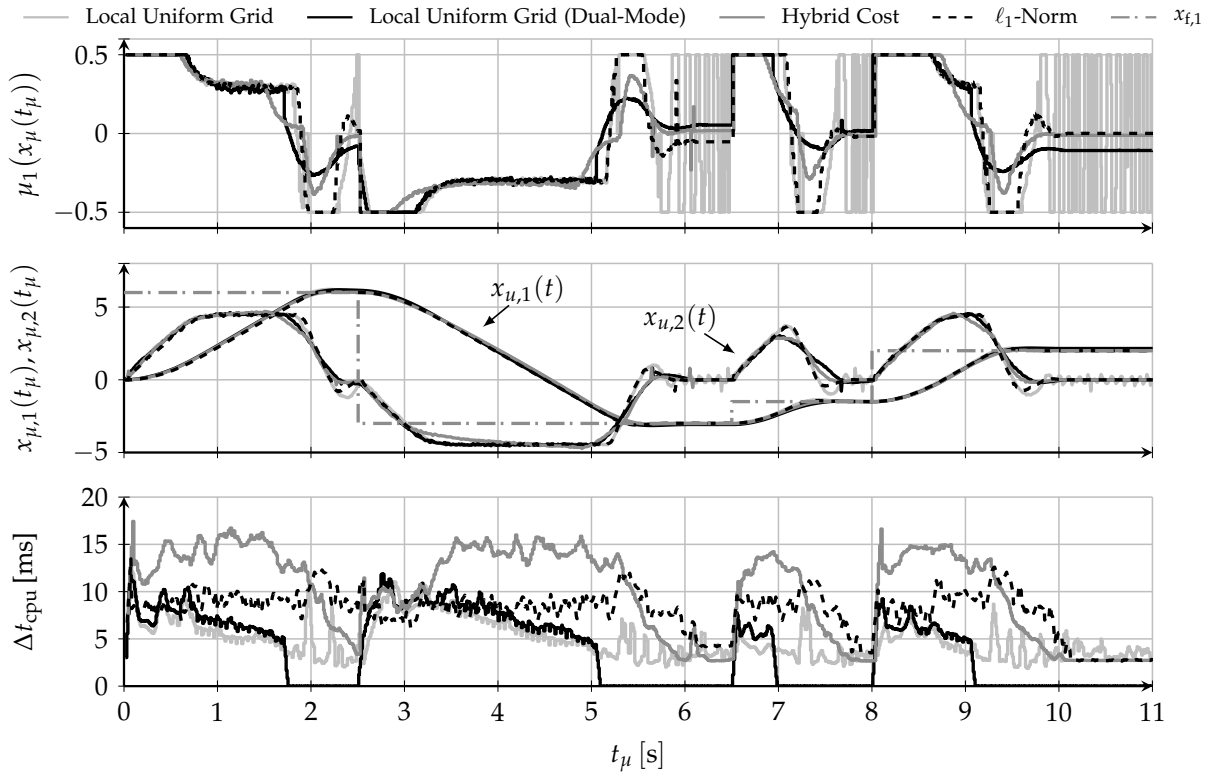


Figure 8.10.: Closed-loop control of the ECP Model 220 with varying final position $x_{\mathrm{f},1}$. These are the local uniform grid, a dual-mode and hybrid cost realization and the ℓ_1-norm approach.

As a result, the region of attraction for the LQR $\mathbb{X}_{\mathrm{lin}}$ is translated to $x_{\mathrm{f}} = (x_{\mathrm{f},1}, 0)$ by applying the corresponding coordinate transformation. To keep the computation times low and to operate the system at about 100 Hz, the discretization grids for the optimal control problems are set to $\Delta t_{\mathrm{ref}} = 0.05\,\mathrm{s}$ respectively $\Delta t = 0.05\,\mathrm{s}$ for the ℓ_1-norm approach. Collocation with forward differences approximates the system dynamics. The horizon of the ℓ_1-norm approach is fixed at $N = 40$ to ensure the feasibility of all transitions. The local uniform grid is subject to grid adaptation according to Algorithm 6.1 and is initialized with $N = N_{\mathrm{init}} = 20$. Figure 8.10 shows the closed-loop control results for varying reference position $x_{\mathrm{f},1}$. The closed-loop sampling time is set to $\Delta t_\mu = 0.01\,\mathrm{s}$, but the controller holds any previous control if the rate is exceeded. Figure 8.10 depicts only the control $\mu_1\big(x_\mu(t_\mu)\big)$ for the first motor, since it is similar to the control for the second motor (with opposite sign). The results demonstrate that the closed-loop performance between the local uniform grid and the ℓ_1-norm approach is very similar. On the one hand, this is reasonable because both controllers should be time-optimal. On the other hand, these results confirm that the closed-loop behavior of the variable discretization grid including grid adaptation is actually time-optimal. Upon arrival at x_{f}, the variable discretization grid shows chattering, while the ℓ_1-norm approach stabilizes the system smoothly. A useful remedy is to consider the dual-mode respectively hybrid cost realizations, which perform quite similar at the beginning of each transition, but then lead to a smooth stabilization at x_{f}.
The required computation times are shown in the bottom plot. For better readability, the sequence of computation times is processed by a moving mean filter of window

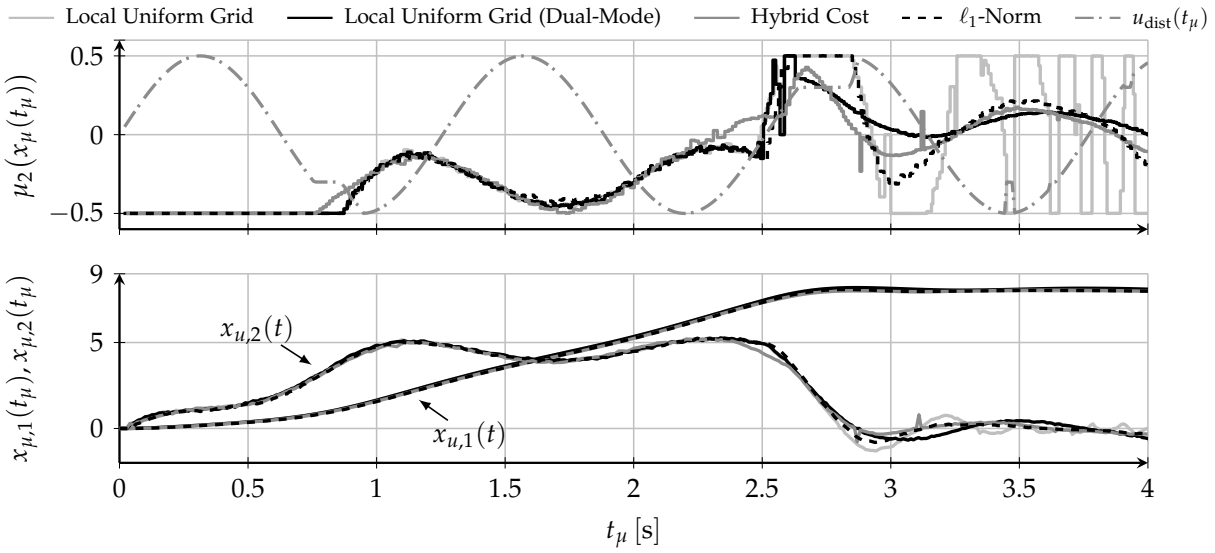


Figure 8.11.: Closed-loop control of the ECP Model 220 with an additive sinusoidal input disturbance at motor 2.

size 5. The computation time of the ℓ_1-norm approach is comparable to the local uniform grid. However, the grid size adaptation leads to slightly smaller computation times while approaching x_f. The dual-mode controller switches to the LQR with negligible computation times. Note that this drop also indicates when the LQR is active in the plot. The hybrid cost approach exhibits larger computation times as the additional (integral) cost function terms are computationally more expensive.
Appendix F.2.5 presents similar results of the global uniform grid containing both control sequences and the visualization of the evolution of N.
Another experiment compares the performance with additional disturbances. The control task consists of guiding the system to $x_\mathrm{f} = (8, 0)^\mathsf{T}$. A sinusoidal disturbance profile $u_{\text{dist}}(t_\mu)$ is added to the control signal of the second motor. The amplitude is set to 0.5 which corresponds to the internal control constraint in the controller and ensures a significant impact on the performance. The accumulated signal is limited to ± 0.8 so as to not damage the real system. Figure 8.11 shows the closed-loop control results including the effective disturbance signal (the disturbance accounts for restriction ± 0.8). Obviously, the approaches behave similarly and are almost able to compensate for the disturbance signal. The local uniform grid approach again leads to chattering which is handled well by the corresponding dual-mode and hybrid cost variants. Note that the frequency adapts to the disturbance signal while stabilizing at x_f. Appendix F.2.5 provides further examples on disturbance rejection.

9

Non-Uniform Grid for Bang-Singular-Bang Systems

Time-optimal controllers usually operate the plant either at control or state limits. Thus, many practical time-optimal control problems consist of either bang-bang, bang-singular-bang or a small finite set of piecewise constant controls. The optimal trajectory often emerges from a sequence of few piecewise constant controls $u_k \in \mathbb{U}$ that include controls $u_{\min} = \min \mathbb{U}, u_{\max} = \max \mathbb{U}$ (bang arcs) or a control from the interior (singular arc). This bang-singular-bang property is generally proven for single-input nonlinear control-affine systems in the plane with bounds only on controls [Sus79; Sus87b; Sus87a]. Or linear systems with only stable and real poles provide the bang-bang property with bounds on the control [Pon87]. Several publications verify this property for certain classes of nonlinear systems. However, a detailed summary of an analytical synthesis of control tasks is beyond the scope of this thesis. But for these types of problems the number of effective control interventions where the control changes is significantly smaller than the usual temporal resolution of the grid. This chapter deals with this circumstance by formulating the direct optimal control problem with a non-uniform shooting grid. Similar to Chapter 6, the underlying grid consists of individual time differences but the requirements to the system dynamics approximation, constraints as well as the grid adaptation differ essentially. Parts of this chapter have been published in [Rös+17i; Rös+17g].

9.1. Multiple Shooting Formulation

Individual time intervals of the discretization grid are retained as explicit parameters subject to optimization similar to the previously defined grid (6.1.1). But instead of forcing individual time intervals Δt_k for $k = 0, 1, \ldots, N-1$ to form a (quasi-)uniform grid, the optimizer explicitly determines the optimal time instances for switching the control signal with respect to the overall minimum-time cost. The resulting grid is possibly non-uniform and some grid intervals are much larger than others. Therefore, it is challenging to adequately satisfy the system dynamic equations with the desired accuracy. The approach favors multiple shooting for direct transcription since it facilitates the adequate solution of the initial value problem with varying Δt_k.

Considering grid (6.1.1) and shooting nodes s_k with $k = 0, 1, \ldots, N$ as defined in (4.1.10), the proposed nonlinear program is as follows:

$$V^*_{\mathrm{dyn},N} = \min_{\substack{u_0,u_1,\ldots,u_{N-1},\\ s_0,s_1,\ldots,s_N,\\ \Delta t_0,\Delta t_1,\ldots,\Delta t_{N-1}}} \sum_{k=0}^{N-1} \left(\Delta t_k + \varrho \Delta t_k^2\right) \tag{9.1.1}$$

subject to

$$\begin{aligned}
&s_0 = x_\mu(t_\mu), \quad s_N = x_\mathrm{f},\\
&s_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad \Delta t_{\min} \leq \Delta t_k \leq \Delta t_{\max},\\
&s_{k+1} = \varphi(\Delta t_k, s_k, u_k),\\
&g(s_k, u_k, \Delta t_k) \leq 0,\\
&h(s_k, u_k, \Delta t_k) = 0,\\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

Inequality constraint $g : \mathcal{X} \times \mathcal{U} \times \mathbb{R}_0^+ \mapsto \mathbb{R}^R$ and equality constraint $h : \mathcal{X} \times \mathcal{U} \times \mathbb{R}_0^+ \mapsto \mathbb{R}^S$ provide a means to ensure additional state and control related conditions. For now, assume $g(\cdot) := 0$ and $h(\cdot) := 0$ until its use cases are discussed in Section 9.1.2. The time intervals Δt_k might also be bounded from above with $\Delta t_{\max}$ in order to adhere to the desired accuracy in solving the system dynamics. Several options are addressed in Section 9.1.1. The term $\varrho \Delta t_k^2$ denotes a regularization term with weight $\varrho \in \mathbb{R}_0^+$. By setting $\varrho = 0$, the optimal control problem can be interpreted as a concatenation of N individual time-optimal control tasks with a constant control u_k each. For a sufficiently high resolution, optimality with respect to the quasi continuous-time solution (3.2.2) follows from Bellman's principle of optimality. The following cases are distinguished:

1. If the number of time intervals N is larger than the minimum number of control interventions N^*, $N - N^*$ time interval parameters become redundant and problem (9.1.1) is underdetermined.

2. If N is smaller than N^*, the optimal control problem might be either suboptimal or infeasible.

Remark 9.1.1. In case the particular nonlinear program solver does not handle underdetermined problems well, the additional regularization term $\varrho \Delta t_k^2$ with a small weight ϱ ensures feasibility of the optimization. Notice, for $\varrho \gg 1$ the objective becomes $V_{\mathrm{dyn},N} \approx \sum_k \Delta t_k^2$ which corresponds to the quasi-uniform grid as described in Chapter 6.

However, the following assumption must hold (similar to Assumption 4.2.2):

Assumption 9.1.1. There exists a finite $N > 0$ for which the optimal control problem (9.1.1) is feasible and its solution constitutes a unique minimizer such that necessary and sufficient conditions hold. Furthermore, the initial problem is solved adequately such that the inherent system dynamics error is almost negligible. For theoretical purposes, it is further assumed that the plant and system dynamics model have zero model mismatch.

9.1.1. System Dynamics Approximation

As already mentioned, some grid time intervals Δt_k can be much larger than others, which is a challenge for numerical evaluation of the system dynamics equation. Two different options are discussed below.

Bounded Time Intervals (Oversampling)

By using a finite upper bound $\Delta t_{\max}$ on individual time intervals Δt_k, a worst-case accuracy is defined a priori. The choice of the numerical integration scheme for calculating $\varphi(\cdot)$ allows different values for $\Delta t_{\max}$. For example, an explicit 5^{th}-order Runge-Kutta integrator allows a much higher bound $\Delta t_{\max}$ than forward Euler.
It is also possible to concatenate several, say $N_{\text{int}} \in \mathbb{N}$, numerical integrators by oversampling each interval Δt_k. Let $\Delta t_{\text{int}} \in \mathbb{R}_0^+$ define the step width for the numerical integration method. Then, the step width is automatically determined by $\Delta t_{\text{int}} = \Delta t_k / N_{\text{int}}$. The shooting equality constraint is then replaced by:

$$s_{k+1} = \sum_{\iota=0}^{N_{\text{int}}-1} \varphi\left(\frac{\Delta t_k}{N_{\text{int}}}, x_u\left(t_k + \iota \frac{\Delta t_k}{N_{\text{int}}}\right), u_k\right). \tag{9.1.2}$$

Here, as the solver adjusts Δt_k, the corresponding Δt_{int} is adjusted as well and $\Delta t_k \to 0$ implies $\Delta t_{\text{int}} \to 0$. Also, $\Delta t_{\max}$ is now chosen according to $N_{\text{int}} \Delta t_{\text{ref}}$. Note that this procedure is a (scaled) multistep method to solve the initial value problem on the entire interval length Δt_k. Other multistep methods apply accordingly.

Fixed-Step Integration

The previously described strategy requires a suitable upper bound $\Delta t_{\max}$ in order to satisfy the system dynamics up to the desired precision. However, whenever the maximum bound cannot be determined a priori or if the computational burden of performing N_{int} steps for even very small Δt_k should be avoided, the following non-standard optimization scheme provides a remedy.
Again, let $\Delta t_{\text{int}} \in \mathbb{R}^+$ define the step width for the numerical integration method. This step width is now set by the user instead of inheriting its actual value from Δt_k like before. For every $k = 0, 1, \ldots, N-1$, there exists a $N_{\text{int},k} \in \mathbb{N}_0$ such that

$$\Delta t_k = N_{\text{int},k} \Delta t_{\text{int}} + \Delta t_{\text{rem},k} \tag{9.1.3}$$

holds. Hereby, $N_{\text{int},k}$ follows from the floor function $\lfloor \Delta t_k / \Delta t_{\text{int}} \rfloor$ and $\Delta t_{\text{rem},k} \in \mathbb{R}_0^+$ constitutes the remainder. The adaptive shooting equation is now given as follows:

$$s_{k+1} = \sum_{\iota=0}^{N_{\text{int},k}-1} \varphi\big(\Delta t_{\text{int}}, x_u(t_k + \iota \Delta t_{\text{int}}), u_k\big) + \varphi\big(\Delta t_{\text{rem},k}, x_u(t_k + N_{\text{int},k} \Delta t_{\text{int}}), u_k\big). \tag{9.1.4}$$

Equation (9.1.4) enforces a maximum step width of Δt_{int} independent of the current value of Δt_k. Hence, the upper bound $\Delta t_{\max}$ can be omitted, in particular $\Delta t_{\max} \to \infty$. Note that this procedure also corresponds to a multistep method with subordinate

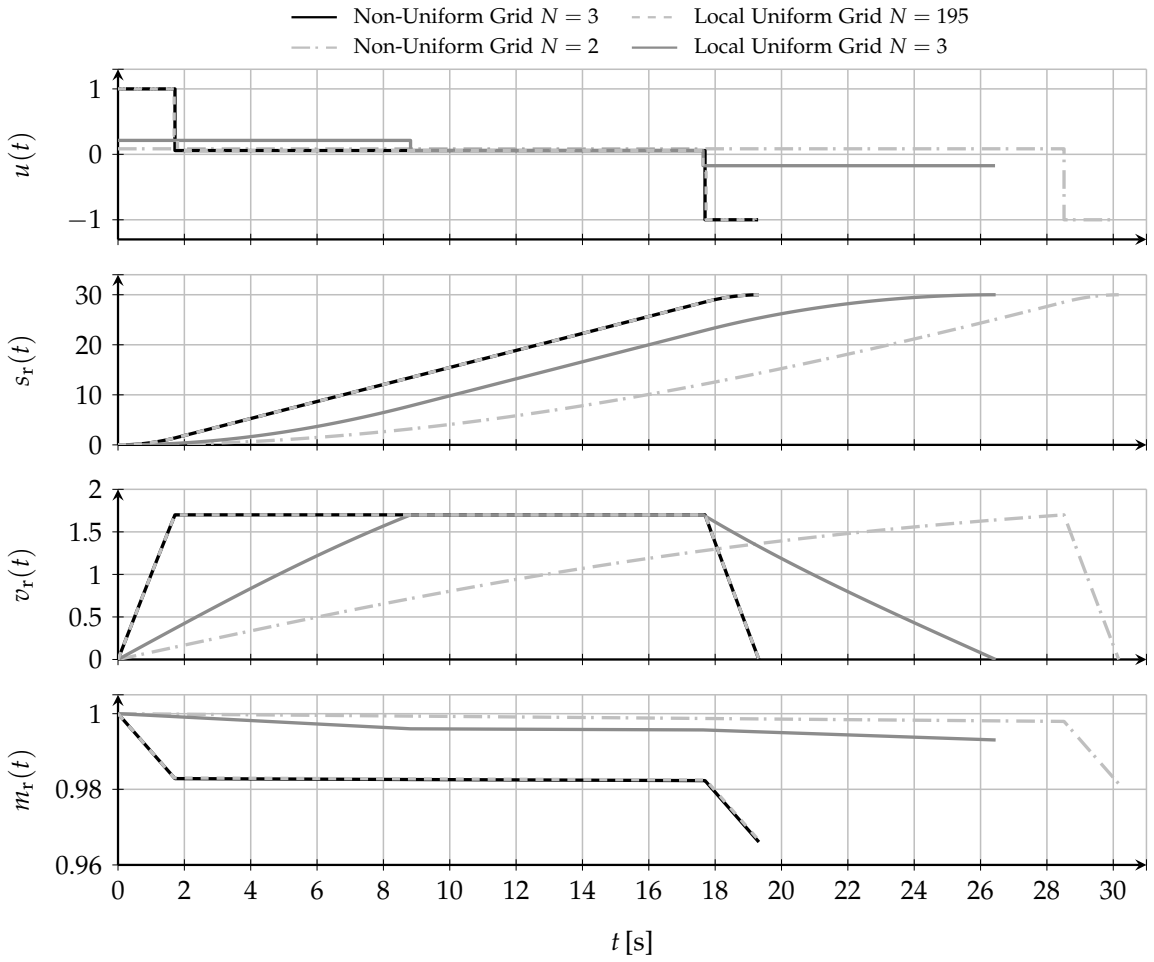


Figure 9.1.: Open-loop control of the free-space rocket system for different non-uniform and local uniform grid types with multiple shooting (forward Euler).

interval lengths Δt_{int}. Incorporating the adaptive shooting equation (9.1.4) in nonlinear program (9.1.1) renders the nonlinear program as non-standard. Unfortunately, equality constraint (9.1.4) is non-smooth with respect to Δt_k which might prevent standard solvers from finding a feasible solution. However, simulations and experiments in this thesis performed well with both IPOPT and SQP. Obviously, this cannot be guaranteed for applications in general. Notice, since u_k is constant over Δt_k, decreasing Δt_{int} also reduces the expected change in terms of the gradient.

Example 9.1.1 (Bang-Singular-Bang Control of the Rocket System)**.** Consider the rocket system with constraint sets as described in Section 3.3.2. The control task is to transit the system from $(0, 0, 1)^\intercal$ to $(30, 0, \cdot)^\intercal$ in minimum time. Notice from the previous examples that the rocket control task is of bang-singular-bang type. It can also be verified that whenever the velocity bound is active between two consecutive shooting nodes the corresponding admissible control is constant for some value within the control bounds (singular arc). Condition $\dot{x}_2(t) = 0$ in (3.3.6) implies $u(t) = 0.02 \cdot 1.7^2 = 0.0578$ for $t \in [t_1, t_2]$ (second time interval) and the active upper bound $x_2(t) \leq 1.7$. Figure 9.1 shows the solutions to the optimal control problem with several grid configurations. Numerical integration is performed with forward Euler and a step width of 0.1 s. For

the uniform grid $\Delta t^* \approx 0.1\,\mathrm{s}$ is approximately achieved with $N = 195$. The non-uniform grid solution with $\varrho = 0$ is obtained via fixed-step integration ($\Delta t_{\mathrm{int}} = 0.1\,\mathrm{s}$). It is remarkable that the non-uniform grid perfectly matches the uniform grid solution with a grid size of only $N = 3$. Further suboptimal solutions are shown for the uniform grid with $N = 3$ and the non-uniform grid with $N = 2$.

9.1.2. Constraint Satisfaction

State constraints $s_k \in \mathbb{X}$ and control constraints $u_k \in \mathbb{U}$ in (9.1.1) are defined only at grid points t_k. In the uniform case, the resolution of the grid is usually high and thus often sufficient to avoid adding further constraints between grid points. But in the case of nonlinear program (9.1.1), the solver adjusts time intervals Δt_k separately and thus shifts the temporal basis of the grid. Consequently, the constraints must either be independent of the variable interval Δt_k or explicitly take them into account.
The former case is ensured if constraints are independent of Δt_k which is stated in the following assumption:

Assumption 9.1.2. Constraints $s_k \in \mathbb{X}$, $u_k \in \mathbb{U}$, $g(\cdot) \leq 0$ and $h(\cdot) = 0$ are satisfied for all intermediate states and controls on any interval Δt_k with $k = 0, 1, \ldots, N-1$ and hence s_k can be substituted by any $x_u(t)$ from $t \in [t_k, t_{k+1}]$ without violating the corresponding constraints in (3.2.2).

If constraints depend only on controls, Assumption 9.1.2 is fulfilled since $u(t)$ is constant for each Δt_k. For constraints involving the state evolution, this property depends on the system equations. For example, the assumption is not satisfied if the optimal state trajectory satisfies constraints at s_k and s_{k+1} but not in the interior of $t \in (t_k, t_{k+1})$ for some k. From a practical and implementation point of view, the assumption might be ensured by one of the following options:

1. Auxiliary constraints $g(\cdot)$ and $h(\cdot)$ can be defined which oversample the state trajectory in the interior of $t \in (t_k, t_{k+1})$. Hereby, the number of interior constraints must be defined in advanced to set a proper dimension R of $g(\cdot)$ and S of $h(\cdot)$. It is recommended to choose a maximum bound on Δt_k in terms of $\Delta t_{\max}$ similar to Section 9.1.1. In order to define constraints in the interior of $t \in (t_k, t_{k+1})$, the solution of the initial value problem is required similar to the shooting constraint. Consequently, the computational burden might be reduced in case the evaluation of $g(\cdot)$ and $h(\cdot)$ relies on the same shared internal memory as the shooting constraint.

2. The standard constraints might be transformed into path constraints. Path constraints are defined in integral form and are ensured along the whole trajectory. For example, a maximum bound on the state trajectory $x_u(t) \leq x_{\max}$ could also be defined as

$$g\left(s_k, u_k, \Delta t_k\right) = \int_{t_k}^{t_k + \Delta t_k} \left\| \max\left(x_u(t) - x_{\max}, 0\right) \right\|_2^2 \mathrm{d}t. \tag{9.1.5}$$

Equation (9.1.5) includes $x_u(t)$ which still requires the solution of the initial value problem of the system dynamics with respect to initial state s_k and control u_k. The maximum operator is applied row-wise. Note, $g(\cdot)$ and the shooting constraint can be computed simultaneously to avoid the additional overhead of solving the same initial value problem several times. The basic idea is to combine both constraints into a single extended system of ordinary differential equations:

$$\begin{pmatrix} \dot{x}_u(t) \\ \tilde{g}(t) \end{pmatrix} = \begin{pmatrix} f\left(x_u(t), u_k\right) \\ \| \max\left(x_u(t) - x_{\max}, 0\right) \|_2^2 \end{pmatrix}. \tag{9.1.6}$$

Hereby, $\tilde{g}(t)$ denotes the integrand of (9.1.5). By solving the initial value problem for (9.1.6) with initial value $\left(s_k, 0\right)^\mathsf{T}$ on the interval $t \in [t_k, t_{k+1}]$ leads to both $x_u(t_{k+1})$ for the shooting equality constraint $s_{k+1} - x_u(t_{k+1}) = 0$ as well as the inequality constraint value $g(\cdot)$. Obviously, the remarks on the accuracy of the underlying numerical integration strategy as described in Section 9.1.1 still hold.

Whereas the first option significantly increases the number of constraints due to oversampling the state trajectory, the second option requires constraints to be expressed in terms of an integral form (ensuring smoothness), the solution of the initial value problem is more computationally expensive and the resulting constraints are nonlinear.

9.1.3. Closed-Loop Integration

The optimal control problem (9.1.1) can be tightly integrated with state feedback as in the previous chapters. Note, the non-uniform grid does not rely on grid adaptation to ensure forward invariance and convergence towards the target state. Since the switching points are not restricted to a uniform grid partition, Bellman's principle of optimality is fulfilled during closed-loop control.
In this chapter, theoretical investigations assume $\Delta t_{\min} = 0$. Feasibility and optimality of the first optimal control problem, zero model mismatch and the absence of disturbances are assumed as before (see Assumption 9.1.1). Since $\Delta t_{\min} = 0$, the following practical stability results ensures convergence to $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$:

Theorem 9.1.1. *Consider nonlinear program* (9.1.1) *with grid size N,* $\varrho = 0$, $\Delta t_{\min} = 0$ *and let* $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$ *represent a steady state such that there exists* $u \in \mathbb{U}$ *with* $f\left(x_\mathrm{f}, u\right) = 0$. *Assume that Assumption 9.1.2 and for any initial* $x_\mu(t_{\mu,n}) \in \mathbb{X}$ *Assumption 9.1.1 hold. The closed-loop feedback is realized with the sampled control law* (5.1.1). *Then, the closed-loop system is* $\mathbb{X}_\mathrm{f}$*-practically asymptotically stable on* $\mathbb{X}$. *If the control law further ensures Remark 3.2.1, the closed-loop system is asymptotically stable on* $\mathbb{X}$.

The final state x_f is excluded according to the potential loss of recursive feasibility as discussed in Chapter 5. The proof is provided in Appendix B.2. It is important to note that $\varrho = 0$ is required to perfectly fulfill the principle of optimality even though small ϱ might be applied in practice. Notice that no requirements on the closed-loop sampling instances are made. If the control law is sampled once, (5.1.1) mimics a feedforward open-loop control law. Instead, fast sampling is required as usual to enhance the capabilities of handling model mismatch and disturbances. Similar to the uniform

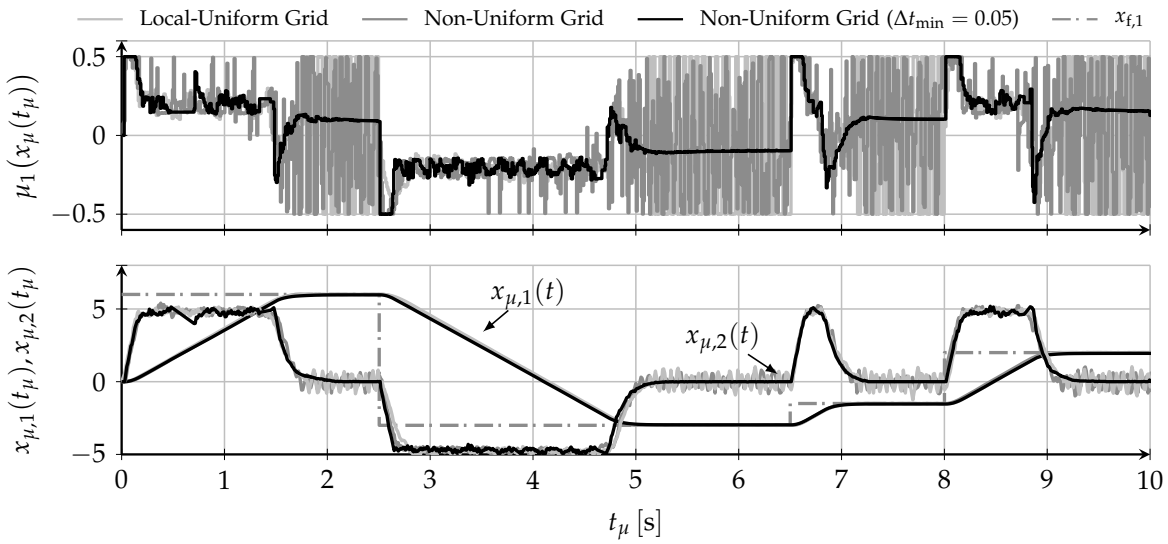


Figure 9.2.: Closed-loop control of the *ECP Industrial Plant Emulator Model 220*. The first component of the final state $x_{\mathrm{f},1}$ changes with respect to time.

case, smooth stabilization is recommended for most practical realizations. Hereby, dual-mode operation is preferred over the hybrid cost as the integral of the cost must be considered appropriately (similar to the solution of the initial value problem as described in Section 9.1.1) and therefore a higher computational burden is expected. Furthermore, note that $\Delta t_{\min} \neq 0$ also invalidates the principle of optimality. Simulations and experiments indicate that bounded intervals are often shifted to the end of the prediction horizon for $\Delta t_{\min} > 0$ and hence implying $P_{\Delta t_{\min}}(x_{\mathrm{f}}, N)$-practically asymptotic stability. However, this observation cannot be generalized to arbitrary control systems. For very small $\Delta t_{\min}$, the principle of optimality holds in an approximate manner which could be sufficient for certain practical applications.

Example 9.1.2. This example shows closed-loop control results with the non-uniform grid approach and the real *ECP Industrial Plant Emulator Model 220* for a varying reference state. Numerical integration is performed with forward Euler and $\Delta t_{\mathrm{int}} = 0.1\,\mathrm{s}$. The sampled control law operates at 100 Hz. Figure 9.2 depicts the control sequence for motor 1 and both state trajectories for the local uniform grid approach ($N = 20$, grid adaptation), the non-uniform grid with $\Delta t_{\min} = 0.0001$ and non-uniform grid with $\Delta t_{\min} = 0.05$. The results show that the solution of the non-uniform grid with $N = 3$ coincides with the higher resolved uniform grid. The special choice of $\Delta t_{\min} = 0.05$ coincidentally leads to a smooth stabilization. However, larger values result in a remaining offset from x_{f}.

9.2. Adaptation of Control Interventions

As mentioned in the previous section, this type of time-optimal control formulation does not require any grid adaptation. However, the ideal number of switching points respectively shooting intervals N^* must be known in advance which might be difficult

Algorithm 9.1.: Adaptation of the number of control interventions.

procedure SOLVEWITHGRIDADAPTATION($\mathcal{P}_{\text{local}}, x_\mu(t_{\mu,n}), x_\text{f}, I_{\text{adapt}}, \tilde{N}_\text{b}$)
  Initialize or update optimization parameters $\mathcal{P}_{\text{local}}$ ▷ Warm-starting
  for all Iterations $i = 1, 2, \ldots, I_{\text{adapt}}$ **do**
    if $i > 1$ **or** $\mathcal{P}_{\text{local}}$ is a warm-start **then**
      $\{N_\text{b}, \mathcal{K}\} \leftarrow$ Count validity of $|u_{k+1} - u_k| \leq u_\epsilon$ **and**
        $\Delta t_k + \Delta t_{k+1} \leq \Delta t_{\max}$ $\forall k$ in $\mathcal{P}_{\text{local}}$
      if $N_\text{b} < \tilde{N}_\text{b}$ **then**
        Insert $\tilde{N}_\text{b} - N_\text{b}$ new grid partitions to $\mathcal{P}_{\text{local}}$
      else if $N_\text{b} > \tilde{N}_\text{b}$ **then**
        erase $N_\text{b} - \tilde{N}_\text{b}$ grid partitions from $\mathcal{P}_{\text{local}}$, in particular
        all u_{k+1}, s_{k+1} and Δt_{k+1} with $k \in \mathcal{K}$
    Solve nonlinear program w.r.t. $\mathcal{P}_{\text{local}}$ ▷ see (9.1.1)
    if $V^*_{\text{dyn},N}$ is infeasible **then**
      Insert a new grid partitions to $\mathcal{P}_{\text{local}}$
      Goto 10 and resolve
  return $\mathcal{P}_{\text{local}}$ ▷ Note, the optimized control sequence is included

in some cases. On the other hand, the benchmarks below show that a poor choice of N can lead to significantly higher computation times and hence finding the required number of switching points can still be beneficial. This section proposes a grid adaptation scheme that seeks a non-uniform shooting grid with the minimum number of control interventions in which the number of constant controls u_k coincides with the number of bang respectively singular arcs. The proposed grid adaptation scheme either is invoked offline, especially if the control task is fixed, or online if reference states change or past time intervals are to be deleted to speed up subsequent optimization.
A rather formal way to obtain the minimum number of control interventions with (9.1.1) is determined iteratively by concurrently testing the effect on an increase of interventions to $N + 1$ or decrease to $N - 1$. If the cost value decreases, at least one control intervention is obsolete. On the other hand, an increase in cost indicates that the current grid structure is either suboptimal or infeasible. Testing and regulating N requires multiple nonlinear programs to be solved in parallel and the linear search substantially depends on the initial length N_{init} which is not preferable for an online integration. For brevity, the computationally expensive procedure is described in more detail in Appendix C.3 and here a more practical and faster strategy is proposed: The redundant control interventions are identified by analyzing the control trajectory obtained from the previous solution of the nonlinear program (9.1.1).
The overall procedure is depicted in Algorithm 9.1. Hereby, $\mathcal{P}_{\text{local}}$ denotes the current parameter set subject to optimization. It is defined as $\mathcal{P}_{\text{local}} := \{u_0, u_1, \ldots, u_{N-1}, s_0, s_1, \ldots, s_N, \Delta t_0, \Delta t_1, \ldots, \Delta t_{N-1}\}$ with $N = N_{\text{init}}$ at the first iteration. The argument $\tilde{N}_\text{b} \in \mathbb{N}$ denotes a desired surplus in the number of control interventions as explained below. The initial $\mathcal{P}_{\text{local}}$ is obtained from a linear interpolation as described in the previous Sections 5.2 and 6.3 between start and final state with zero controls $u_k = 0$.

If $\mathcal{P}_{\text{local}}$ does not constitute a warm-start, that is a solution from a previous solver invocation, the procedure first solves nonlinear program (9.1.1) in line 10. In subsequent iterations i the control sequence is investigated for potentially redundant control interventions N_{b} in terms of equality of subsequent controls $|u_{k+1} - u_k| \leq u_\epsilon$. Theoretically, one expects $u_\epsilon = 0$ but in practical implementations, a small but finite margin accounts for numerical imprecision. Furthermore, the condition $\Delta t_k + \Delta t_{k+1} \leq \Delta t_{\max}$ verifies that the optimizer is able to combine both intervals without loosing feasibility guarantees. Consequently, a large $\Delta t_{\max}$ is crucial if the number of control interventions should be minimal. Refer to Section 9.1.1 and Section 9.1.2 for details on the selection of $\Delta t_{\max}$. The indices k of redundant controls are gathered in set $\mathcal{K}$. If $N_{\text{b}} < \tilde{N}_{\text{b}}$, the grid is augmented by $\tilde{N}_{\text{b}} - N_{\text{b}}$ additional intermediary grid partitions in line 7. Insertion is performed subsequently for the currently largest time interval $\max\{\Delta t_k | k = 0, 1, \ldots, N-1\}$ by linear interpolation. In case $N_{\text{b}} > \tilde{N}_{\text{b}}$, the redundant grid points and parameters with indices $\mathcal{K}$ are removed from the control sequence (line 9).
If there is no surplus of grid points $\tilde{N}_{\text{b}} = 0$ and the initial N_{init} is overestimated, nonlinear program (9.1.1) is underdetermined such that $N_{\text{b}} > 0$ and redundant grid points are removed. Convergence is reached at this point. On the other hand, if N_{init} is underestimated such that the nonlinear program becomes infeasible, a new intermediary grid point is inserted in line 12 and the nonlinear program is resolved with the augmented grid structure. In case N_{init} is underestimated, but the solution of the nonlinear program is suboptimal albeit feasible, all subsequent controls differ ($\mathcal{K} = \emptyset$). Thus the algorithm does not enhance the grid structure even though the true optimal solution requires additional interventions. Consequently, $\tilde{N}_{\text{b}} > 0$ is crucial for recovering from suboptimal solutions with too few interventions. $\tilde{N}_{\text{b}}$ is chosen such that $V^*_{\text{dyn},N} > V^*_{\text{dyn},N+\tilde{N}_{\text{b}}}$ holds for all suboptimal N. In practice, $\tilde{N}_{\text{b}} = 1$ is often sufficient for recovery.

Remark 9.2.1. Notice, the nonlinear program solver in line 10 might be terminated prior to convergence. Suitable scheduling of adaptation iterations I_{adapt} and internal solver iterations significantly increases the overall convergence speed. Due to the sparse structure, partial solutions are likely to indicate redundant control interventions at an early stage of convergence. These grid partitions are removed prior to complete convergence with the benefit of reducing the number of parameters in subsequent iterations. In case of limited computational resources, the number of control interventions is determined a priori by invoking Algorithm 9.1 on samples of initial states.

Example 9.2.1 (Optimal Control of a Non-Bang-Bang Configuration)**.** This example considers the Van der Pol oscillator as introduced in Section 3.3.1 with an additional state constraint $|x_2(t)| \leq 0.5$. Example 4.3.2 already demonstrates that adding this constraint results in a non-bang-bang control type as $\dot{x}_2(t) = 0$ in (3.3.2) implies a non-constant $u(t)$ and thus Assumption 9.1.2 might become invalid. The optimal control problem in this example demands a transition from the origin to $x_{\text{f}} = (1, 0)^\intercal$. Figure 9.3 shows the solutions to the optimal control problem for various configurations. The size of the local uniform grid is set to $N = 21$ in order to achieve a discretization width of about 0.1 s. The non-uniform grid approaches with $\varrho = 0$ are applied with

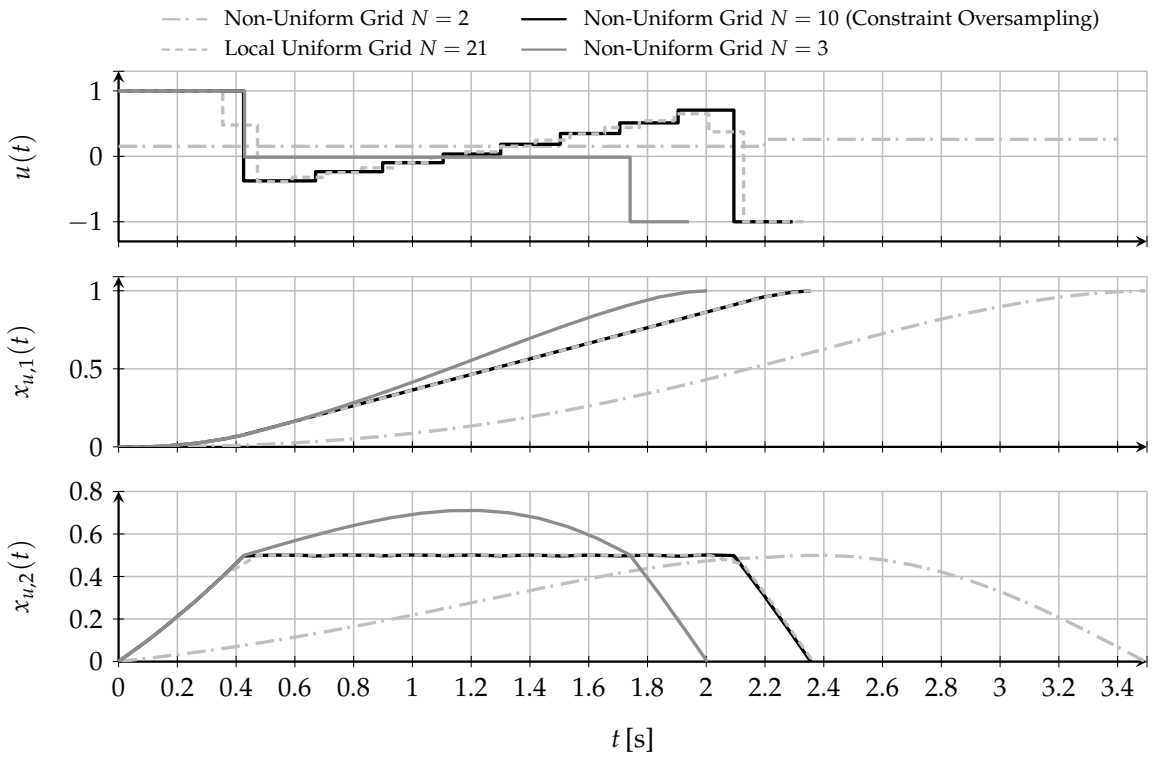


Figure 9.3.: Open-loop control of the Van der Pol Oscillator with several non-uniform and local uniform grid configurations.

fixed-step integration at $\Delta t_{\text{int}} = 0.1\,\text{s}$. Furthermore, Algorithm 9.1 is invoked with $N_{\text{init}} = 20, u_\epsilon = 0.1, N_{\text{b}} = 0$ and $I_{\text{adapt}} = 10$. The algorithm converges to $N = 3$ and the corresponding solution is of bang-singular-bang type. Figure 9.3 clearly shows that the state bound on $x_2(t)$ and thus Assumption 9.1.2 is violated. According to the first proposed option in Section 9.1.2, two intermediate state constraints in every interval at $t_k + \Delta t_k/3$ and $t_k + 2\Delta t_k/3$ are imposed. Algorithm 9.1 is reinvoked and the non-uniform grid size converges to $N = 10$ while adhering to all constraints. The solution matches the optimal theoretical switching points even more precisely (the overall transition is 5 ms faster). A further suboptimal solution for the non-uniform grid with $N = 2$ is shown as well.

9.3. Performance Comparison and Benchmark Results

This section provides benchmark results for the required computation times. The optimal control problems are solved with settings as described in Section 8 for IPOPT. On the other hand, the SQP algorithm cannot be applied as before since squaring the cost function eliminates the desired non-uniformity. The SQP implementation still favors the Hessian of the (non-squared) cost function over the Hessian of the Lagrangian. However, in order to obtain practical results, a suited inertia correction is inevitable. Refer to Appendix E.2.2 for more details. As before, first- and second-order derivatives are efficiently computed by using the hypergaph according to the multiple

Table 9.1.: Comparison of computation times for varying final positions $s_{r,f}$ of the rocket system.

		$s_{r,f} = 5$	$s_{r,f} = 10$	$s_{r,f} = 30$	$s_{r,f} = 50$	$s_{r,f} = 100$	$s_{r,f} = 300$	$s_{r,f} = 500$
		Δt_{cpu} [ms]	Δt_{cpu} [ms]	Δt_{cpu} [ms]	Δt_{cpu} [ms]	Δt_{cpu} [ms]	Δt_{cpu} [ms]	Δt_{cpu} [ms]
Explicit	Oversampling	6.48	8.85	42.32	186.68	227.31	12 386.27	20 958.22
	Fixed-Step	9.62	13.34	28.46	80.46	35.57	273.88	234.45
	Local Uniform	11.95	19.42	36.51	57.22	115.71	337.63	647.10
BFGS	Oversampling	5.88	9.38	11.68	19.37	39.95	117.18	1377.74
	Fixed-Step	8.33	**6.88**	**9.02**	**13.50**	**13.64**	**32.06**	**57.99**
	Local Uniform	10.82	15.86	38.77	60.14	120.62	361.43	752.37
SQP	Oversampling	**4.98**	13.04	24.44	33.97	53.71	105.61	1181.58
	Fixed-Step	6.78	9.16	19.34	32.93	33.64	45.25	691.82

shooting configuration for both IPOPT and SQP. Finally, the section concludes with the closed-loop integration and an application to the *ECP Industrial Plant Emulator 220*.

9.3.1. Performance Evaluation of the Non-Uniform Grid Approach

The first benchmark is based on the control task of Example 9.1.1 but the final position $s_{r,f}$ is subject to change. By increasing $s_{r,f}$, the intermediate shooting interval in which the system is operated at the upper bound on $x_2(t)$ is extended in terms of time. This in turn increases the non-uniformity of the grid. Similar as in the examples before, the optimal control approaches are tuned to almost adhere to forward Euler integration with step length 0.1 s and the regularization weight ϱ is set to zero. Table 9.1 shows the results for several final positions $s_{r,f}$. The local uniform approach serves as a reference and its grid size is chosen manually to match $\Delta t^* \approx 0.1\,\mathrm{s}$, in particular $N = \{47, 76, 194, 311, 605, 1782, 2959\}$. Also results for the bounded time intervals respectively oversampling approach as described in Section 9.1.1 are provided. Hereby, it is still $N = 3$ but the number of additional samples per interval is chosen manually as $N_{int} = \{19, 39, 159, 279, 579, 174, 2949\}$ to ensure that the length of the singular arc Δt_1 (second shooting interval) is lower than Δt_{max}. The grid size for the fixed-step approach is also set to $N = 3$. In contrast to the uniform grid, the BFGS realization is more efficient for larger problem sizes than the explicit computation of the Hessian in case of the fixed-step non-uniform grid. The results indicate a much faster performance of the fixed-step approach rather than for oversampling. It is remarkable that the non-uniform grid with fixed-step integration requires only a fraction of the computation time that is required for the (local) uniform grid and large $s_{r,f}$. Similar to the benchmark results for the uniform grid, SQP is faster for small problem sizes but IPOPT gains more efficiency as $s_{r,f}$ increases.

The previous evaluation considered computation times for the case that the ideal grid size N^* for the non-uniform grid is specified in advance. A second benchmark evaluates the impacts of solving (9.1.1) for other choices of N. Note, grid adaptation is still inactive and its benefits are analyzed later in Section 9.3.2. The final position of the rocket is now fixed to $s_{r,f} = 30$. Table 9.2 presents the computation times Δt_{cpu} and determined final times t_f for several grid sizes and values of ϱ. As a larger value of

Table 9.2.: Computation and transition times for the fixed-step setting with $\Delta t_{\text{int}} = 0.1$, varying grid sizes N and varying cost weights ϱ applied to the rocket system with $s_{\text{r,f}} = 30$.

		$N^* = 3$		$N = 4$		$N = 9$		$N = 19$	
		Δt_{cpu} [ms]	t_{f} [s]	Δt_{cpu} [ms]	t_{f} [s]	Δt_{cpu} [ms]	t_{f} [s]	Δt_{cpu} [ms]	t_{f} [s]
BFGS	$\varrho = 0$	9.02	19.32	9.89	19.32	431.37	19.32	525.40	19.32
	$\varrho = 0.01$	7.91	19.32	8.17	19.32	280.14	19.48	347.58	19.32
	$\varrho = 0.1$	403.04	21.81	24.69	19.33	17.07	19.32	208.49	19.51
	$\varrho = 0.5$	448.07	23.18	401.49	20.78	11.96	19.32	696.21	19.32
SQP	$\varrho = 0$	19.34	19.33	85.42	19.32	707.23	19.30	1338.77	19.31
	$\varrho = 0.01$	7.83	19.32	10.50	19.32	108.30	19.33	323.44	19.31
	$\varrho = 0.1$	13.37	21.91	24.77	19.33	19.37	19.32	117.64	19.32
	$\varrho = 0.5$	14.26	23.20	17.37	20.77	16.27	19.32	56.22	19.32

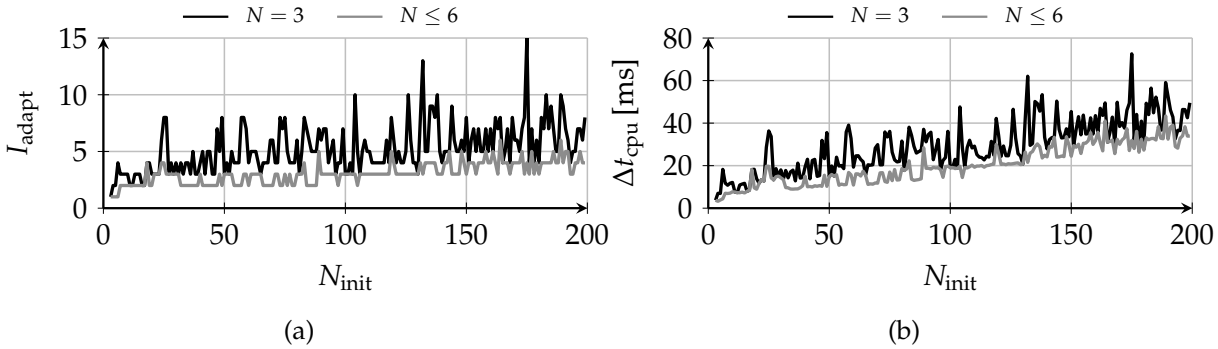


Figure 9.4.: Convergence analysis of Algorithm 9.1: (a) The number of iterations until $N = 3$ respectively $N \leq 6$ is reached and (b) the required computation time.

ϱ reduces the amount of non-uniformity, the transition time t_{f} provides a reasonable measure of suboptimality. Note, the system dynamics are solved similarly for all configurations based on the fixed-step configuration, forward Euler and $\Delta t_{\text{int}} = 0.1\,\text{s}$. The results for $\varrho = 0$ indicate that the required computation time grows rapidly for increasing grid sizes N. As a consequence, it is crucial to provide a proper initial guess for N whenever applying the non-uniform grid. Otherwise, the uniform grid still performs better as it can be verified in Table 9.1 for $s_{\text{r,f}} = 30$ in comparison to $N = 9$ or $N = 10$. For small $\varrho > 0$ the computation time can still be reduced without having a significant impact on the final time t_{f}. The additional cost serves as regularization and improves the numerical properties of the optimization problem. Further increasing values $\varrho > 0$ devolve the grid into the quasi-uniform grid and affect the final time t_{f} noticeably. Notice the large computation times for IPOPT BFGS which were already revealed for certain quasi-uniform grid configurations in Section 8. In contrast, SQP is better suited to handle larger values of ϱ.

9.3.2. Grid Adaptation

The previous results show that starting with a poor initial guess of the grid size N can reduce the performance significantly. Furthermore, the minimum number of switches

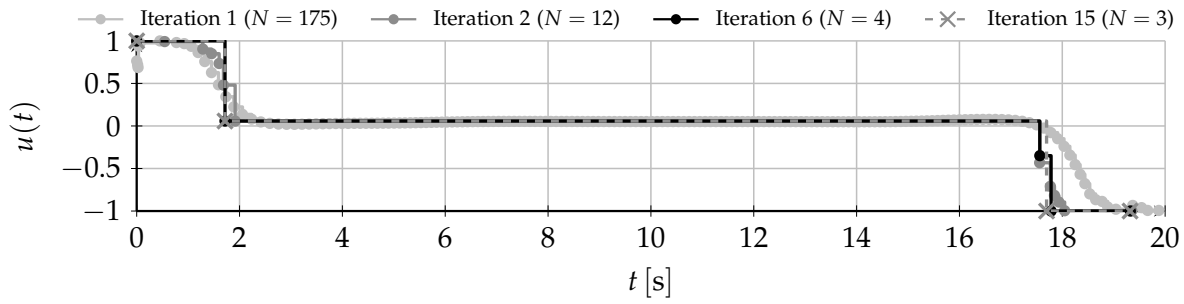


Figure 9.5.: Intermediate optimal control sequences during grid adaptation.

in the control might not be available a priori or it changes during closed-loop control. To this end, Section 9.2 proposes a grid adaptation scheme suited for the non-uniform grid. This section investigates the induced performance implications of applying Algorithm 9.1 to solve the non-uniform grid optimal control problem for the very same control task and $\varrho = 0$. The application of Algorithm 9.1 exploits the fact that early-terminated solutions are usually fairly close to the optimal solution after only a few iterations as discussed in Remark 9.2.1. In the following analysis, IPOPT with BFGS is utilized and the maximum number of solver iterations is set to 10. The similarity threshold for two consecutive controls is set to $u_\epsilon = 0.1$. Figure 9.4a shows the number of adaptation iterations I_{adapt} until the grid size converges to $N \leq 6$ and $N = 3$ with respect to varying initial grid sizes N_{init}. Note, a small grid size $N \leq 6$ is reached already after $1 - 6$ iterations for all tested N_{init}. Converging to the optimal grid size requires a few more iterations. Cases in which the number of iterations is comparably large suffer from the following effect. Some redundant grid points accumulate at the switching points. Consider Figure 9.5 for a detailed illustration of the convergence behavior for $N_{\text{init}} = 175$ at which 15 iterations are required. Already after the first adaptation iteration (which corresponds to 10 solver iterations) the control sequence is close to the bang-singular-bang type. Since many consecutive controls already fulfill the desired similarity threshold, the grid size is reduced to $N = 12$ in the second iteration. Starting from iteration 6, the grid size stagnates at $N = 4$ since a redundant control sample is located directly within the two control arcs at $\approx 18\,\text{s}$. In further iterations, the redundant control sample slightly moves towards the previous arc until the similarity threshold is reached in iteration 15. However, as noted in Section 9.2 a small surplus $\tilde{N}_\text{b}$ is often desired. Figure 9.4b depicts the computation times until convergence to $N = 3$ respectively $N \leq 6$ with respect to N_{init}. The overall trend is almost linear and compared to the results in Table 9.2 the performance can be improved significantly also for poor grid size choices. Note, that the number of required control switches is often similar to the magnitude of the system dimension (by assuming bang-singular-bang characteristics) and thus exaggerated initial guesses as $N_{\text{init}} = 100$ or $N_{\text{init}} = 200$ are presented just for demonstrating the effectiveness of the approach.

10
Conclusion and Outlook

The time-optimal control of nonlinear systems has long been a topic of interest in the research and application of control technology. The majority of approaches are defined for optimal feedforward control in which state feedback is either omitted or achieved by an additional tracking controller. Even if the established time transformation method can be applied for closed-loop control similar to conventional predictive control with terminal equality conditions, the scaled dynamics make a profound stability analysis difficult. To this end, this thesis contributes new theoretical concepts and stability results for time-optimal control with state feedback based on direct transcription with variable discretization grids and suitable grid adaptation schemes. On top of that, this thesis deals with the practical realization and implementation in the context of sparsity exploitation and efficient online optimization. An extensive benchmark analysis using a developed C++ framework, which is tailored for speed and efficiency, compares the proposed algorithms with each other and with methods available in the literature. All benchmarks are performed with an efficient and already established interior-point optimization algorithm (IPOPT) and a proposed sequential quadratic programming (SQP) implementation. The SQP algorithm is tailored for the time-optimal control problems that suffer from indefinite Hessians. Numerous examples in simulation and with a real experiment demonstrate the effectiveness of the proposed approaches and show both positive and negative effects. A brief summary and concluding remarks for the main concepts and results are listed below.

Time-Optimal Control with Variable Discretization Grids

Direct transcription follows the idea of first discretizing and then optimizing the optimal control problem. The control trajectory and often also the state trajectory are discretized with respect to a temporal grid. The basic idea is to define the grid interval lengths as dedicated optimization parameter(s) and thus replace the minimization of the entire transition time by the minimization of the local time information given by the grid interval lengths. As a starting point for any controller design, it is reasonable to rely on a uniform grid for which three different types of variable (and still uniform) discretization grids are proposed:

- The *global uniform grid* is defined by a *single optimization parameter* which serves as discretization width for the individual grid partitions.

- In contrast, the *local uniform grid* specifies an individual *optimization parameter for each grid partition*. Although the number of optimization parameters is larger than for the global grid, the structure of the inherent optimization problem is sparser.
- A modification of the local uniform grid, which copes with *fewer equality constraints*, leads to a *quasi-uniform grid* representation.

In general, all grids are capable of realizing time-optimal control. The local and global uniform grid perform comparatively well in computation time under all benchmarks. In some cases, the local uniform grid slightly exceeds the global one and vice versa. However, the local uniform grid shows fewer occasional peaks in the computation time. The quasi-uniform grid performs well for SQP, but for IPOPT it does not perform well with collocation via finite differences, although the solutions converge towards the optimal solution. Accordingly, a practical implementation could begin with the local uniform grid and tries to improve performance by other grid options if necessary.
In case the control system exhibits bang-singular-bang control characteristics and constraint sets are rather simple (for example control bounds), the additionally proposed *non-uniform grid* outperforms any previous uniform grid approach. The non-uniform grid comes with an extended multiple shooting scheme to appropriately approximate the system dynamics with respect to varying time interval lengths. A comparative analysis shows that this grid is advantageous for these special control systems because the computation times are low even if the horizon is large. On the other hand, benchmarks indicate that performance is worse when the grid size is poorly initialized. To remedy this, a proposed grid adaptation scheme searches for the proper grid size either online or offline.

Analysis and Comparison of Numerical Schemes for Dynamics Approximation

The numerical solution or approximation of the system dynamics with respect to the discretization grid is indispensable for direct transcription. Standard direct transcription methods work well on the proposed variable discretization grids. This thesis investigates and analyzes the applicability of multiple shooting and two collocation methods for time-optimal control. Hermite-Simpson collocation approximates the system dynamics by quadratic polynomials, resulting in a Hermite cubic spline in the state trajectory. Several choices for the control parameterization are investigated, for example constant, linear, linear with midpoint, or quadratic polynomials. Simulations show that the linear and quadratic polynomial with midpoint are not suitable for time-optimal control problems in their standard form as the additional degrees of freedom allow constraint violations between grid points. Instead, piecewise constant controls or linear control representations without midpoints are to be preferred.
For simple dynamics, which means that low-order numerical integration or collocation methods exhibit almost negligible discretization errors with respect to the given grid, the benchmarks reveal that it is more efficient to increase the grid size instead of switching to a higher-order discretization method to achieve the desired accuracy. Explicit methods in multiple shooting or straightforward schemes in collocation via

finite differences perform well, especially in combination with state feedback. On the other hand, for more complex systems, the choice of higher integration orders while reducing the grid size is slightly more efficient on the benchmarks. Implicit methods are required for stiff systems to account for numerical stability.

Feedback Control

MPC generally realizes feedback by repeatedly solving the subordinate optimal control problem during runtime. The optimal control problems with time transformation include the final state as equality condition such that stability and recursive feasibility are often *assumed* to be valid. Similarly, the formulations based on variable discretization grids also include this terminal condition. By introducing a suitable control parameterization for direct transcription, however, the principle of optimality and hence recursive feasibility can no longer be guaranteed. In addition, after reaching the steady state, the time-optimal control problem does not necessarily provide controls to stabilize the system properly. The proposed shrinking horizon grid adaptation scheme ensures asymptotic stability and feasibility under viability and zero-model mismatch assumptions. With small grid sizes, however, viability in practice cannot be guaranteed sufficiently, since the final condition is rather restrictive and even small disturbances lead to loss of viability. For this purpose, practical stability results are presented which take into account a predefined minimum grid size in order to retain viability. Integration with a dual-mode control scheme results in smooth stabilization while restoring asymptotic stability. A further extension of the grid adaptation scheme automatically finds the grid size for a desired temporal resolution. Even if practical stability can only be guaranteed for a proper initialization, the scheme has advantages in practice, for example increasing the grid in case of disturbances or changing reference states.
The control schemes are compared to the state-of-the-art methods TOMPC and the stabilizing ℓ_1-norm cost approach. Both the ℓ_1-norm approach and variable discretization grids outperform TOMPC in terms of computation times. The main advantage of the ℓ_1-norm approach is that it is a receding horizon controller that does not require any grid adaptation as long as there is a sufficiently large horizon length. The benchmark results show that the ℓ_1-norm approach is computationally more demanding for longer horizons.

Sparsity Structure Exploitation and Online Optimization

This thesis proposes hypergraph representations for the nonlinear programs resulting from direct transcription. Hypergraphs capture the inherent sparse structure used to efficiently compute first- and second-order derivatives that are needed for the solution. Due to the explicit sparsity exploitation, the computational burden for the solution of nonlinear programs increases almost linearly in relation to the grid size. In addition, the hypergraph allows the continuous reconfiguration of problem dimensions while maintaining the inherent sparse structure with almost negligible overhead. This is especially crucial for grid adaptation during closed-loop control.
The Hessian of the Lagrangian is cheaper to compute for time-optimal control prob-

lems as the cost function is linear. Benchmarks show that computing the Hessian explicitly through finite differences and hypergraph is more efficient than the application of the common BFGS method. The SQP implementation is fast for small problem sizes (here approximately around $N \leq 30$). For larger problems, IPOPT usually outperforms SQP except for the global uniform grid.

Outlook

The results of this thesis clearly confirm that variable discretization grids are well suited for time-optimal MPC realizations. Nevertheless, although the systems, examples, simulations and real experiments have been carefully selected for the limited scope of this thesis, a wider range of applications and benchmarks are particularly important to further generalize the results. In particular, runtime performance depends heavily on specific solver configurations and implementation details. Solvers are generally tuned to perform well on a broad range of optimization problems. Even effective interior-point and sequential quadratic programming solvers depend heavily on individual choices about the underlying globalization strategy, inertia correction, watchdog respectively second-order correction and empirically determined heuristics. In addition to further benchmarking and practical realizations, the following list provides some suggestions for further research directions:

- The stability results in this thesis assume zero model mismatch and no disturbances. However, the theory of robust MPC is already very sophisticated in the literature and takes into account modeling errors, numerical errors, external disturbances, uncertain parameters and measurement errors. Future work could investigate the robustness properties of time-optimal control with variable discretization grids.

- Applications with tight real-time constraints often require solvers that terminate the current optimization step early in order to maintain the desired sampling rate. Especially in practice, the premature termination in combination with a proper warm-start has proven to be very effective. From a theoretical point of view, the literature provides results under which these suboptimal realizations are able to retain properties like recursive feasibility and stability. The transfer of these results to the variable grids could be a possible direction for future work.

A

Mathematical Definitions

A.1. Lipschitz Continuity

Let the dynamic system be defined as

$$\dot{x}(t) = f\big(x(t), u(t)\big) \tag{A.1.1}$$

with $x : \mathbb{R} \mapsto \mathcal{X}$ and $u : \mathbb{R} \mapsto \mathcal{U}$. Usually, state and control sets are real vectors $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{U} = \mathbb{R}^q$ with $p \in \mathbb{N}$ and $q \in \mathbb{N}$, respectively.

Definition A.1.1 (Lipschitz Continuity)**.** The vector field $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$ is continuous and Lipschitz in its first argument if for each $\bar{r} > 0$ there exists a constant $\bar{L} > 0$ such that

$$\|f\big(x_1, u\big) - f\big(x_2, u\big)\| \leq \bar{L}\|x_1 - x_2\| \tag{A.1.2}$$

holds for all $x_1, x_2 \in \mathcal{X}$ and all $u \in \mathcal{U}$ with $\|x_1\| \leq \bar{r}$, $\|x_2\| \leq \bar{r}$ and $\|u\| \leq \bar{r}$ [GP17].

A.2. Control Parameterization

The locally Lebesgue integrable control trajectory $u(t)$ is defined by $u : \mathbb{R} \mapsto \mathcal{U}$ and $\mathcal{U} = \mathbb{R}^q$ with some positive q. The corresponding function space is denoted by $L^\infty(\mathbb{R}, \mathcal{U})$ [GP17]. Point-to-point time-optimal control is restricted to the closed interval $[t_0, t_\mathrm{f}]$ and hence it is sufficient to consider $u \in U = L^\infty([t_0, t_\mathrm{f}], \mathcal{U})$. However, t_f is subject to change.
Direct transcription methods discretize the close time-interval $[t_0, t_\mathrm{f}]$ by a grid with N partitions:

$$t_0 \leq t_1 \leq \ldots \leq t_k \leq \ldots \leq t_N = t_\mathrm{f}. \tag{A.2.1}$$

The general control space on interval $[t_k, t_{k+1}]$ is denoted by $U_k := L^\infty([t_k, t_{k+1}], \mathcal{U})$. However, direct transcription also restricts the control space to certain subsets of U.

Constant Controls

The control space for grid (A.2.1) is defined as follows:

$$\begin{aligned} \mathscr{U}^N(t_\mathrm{f}) :=& \{u \in L^\infty([t_0, t_\mathrm{f}], \mathcal{U}) \mid \exists u_0, u_1, \ldots, u_{N-1} \in \mathcal{U}, \\ & u(t) := u_k \text{ with } t \in [t_k, t_{k+1}) \text{ for } k = 0, 1, \ldots, N-1\}. \end{aligned} \tag{A.2.2}$$

Linear (Mean) Control Spline

This control parameterization defines a linear polynomial for each interval. In Hermite-Simpson collocation it is referred to as (linear) mean control parameterization (see Section 4.1.3) as the control at the collocation point is defined in terms of the mean.
The control space for grid (A.2.1) is defined by:

$$\begin{aligned}\mathscr{U}^N(t_\mathrm{f}) :=\{&u \in L^\infty([t_0, t_\mathrm{f}], \mathcal{U}) \mid \exists u_0, u_1, \dots, u_N \in \mathcal{U}, u(t_k) := u_k, u(t_N) := u_N, \\ &u(t) = u_k + (t - t_k)(u_{k+1} - u_k) \text{ with } t \in [t_k, t_{k+1}] \text{ for } k = 0, 1, \dots, N-1\}.\end{aligned} \tag{A.2.3}$$

Linear Control Spline

Hermite-Simpson collocation with controls at midpoints $t_{k+0.5} := 0.5(t_k + t_{k+1})$ increase the degrees of freedom in control. A straightforward approach interpolates the interval linearly, in particular by two linear functions. The resulting control space is:

$$\begin{aligned}\mathscr{U}^N(t_\mathrm{f}) :=\{&u \in L^\infty([t_0, t_\mathrm{f}], \mathcal{U}) \mid \exists u_0, u_{0.5}, u_1, \dots, u_N \in \mathcal{U}, \\ &t_{k+0.5} := 0.5(t_k + t_{k+1}), u(t_k) := u_k, u(t_{N-0.5}) := u_{N-0.5}, u(t_N) := u_N, \\ &u(t) = u_k + (t - t_k)(u_{k+0.5} - u_k) \text{ with } t \in [t_k, t_{k+0.5}) \text{ and} \\ &u(t) = u_{k+0.5} + (t - t_{k+0.5})(u_{k+1} - u_{k+0.5}) \text{ with } t \in [t_{k+0.5}, t_{k+1}] \\ &\text{for } k = 0, 0.5, 1, \dots, N-1\}.\end{aligned} \tag{A.2.4}$$

Piecewise Quadratic Polynomials

Similar to the linear control spline, Hermite-Simpson allows to interpolate each interval with a quadratic spline according to (4.1.16) as long as midpoint $t_{k+0.5} := 0.5(t_k + t_{k+1})$ is considered.
The control space is given by:

$$\begin{aligned}\mathscr{U}^N(t_\mathrm{f}) :=\{&u \in L^\infty([t_0, t_\mathrm{f}], \mathcal{U}) \mid \exists u_0, u_{0.5}, u_1, \dots, u_N \in \mathcal{U}, \\ &t_{k+0.5} := 0.5(t_k + t_{k+1}), u(t_k) := u_k \text{ for } k = 0, 0.5, 1, \dots, N-1, u(t_N) = u_N, \\ &u(t) \text{ according to (4.1.16) with } t \in [t_k, t_{k+1}] \text{ for } k = 0, 1, \dots, N-1\}.\end{aligned} \tag{A.2.5}$$

A.3. Numerical Solution to Initial Value Problems

Several numerical methods exist to solve initial value problems and they are usually categorized in single- and multistep methods as well as implicit and explicit methods. Euler and Runge-Kutta methods are well known singlestep methods whereas Adams–Bashforth and Adams–Moulton methods constitute common multistep methods. Individual schemes differ in terms of convergence, order and stability. A comprehensive overview of numerical methods is provided in [But16].

The benchmarks and experiments in this thesis consider explicit forward Euler and an explicit 5^{th}-order Runge-Kutta scheme for multiple shooting. Section 4 already defines the implicit methods used in collocation. Let the dynamic system be defined by (3.1) respectively (A.1.1) and Assumption 3.1.1 hold. According to (3.1.2), the solution with initial value $x(t_\text{s} = 0) = x_\text{s}$ and $t \in I$ with some open time interval $I \subseteq \mathbb{R}$ is then given by:

$$x(t) := \varphi\big(t, x_\text{s}, u(t)\big) = x_\text{s} + \int_{t_\text{s}=0}^{t} f\big(x(\tau), u(\tau)\big)\mathrm{d}\tau. \tag{A.3.1}$$

The numerical solution to (A.3.1) is based on the temporal discretization $t_k = t_s + k\Delta t$ with $k \in \mathbb{N}_0$ and step width $\Delta t \in \mathbb{R}_0^+$. Furthermore, let $x(t_k) := x_k$ and $u(t_k) := u_k$ define approximations to the state and control trajectory at grid points t_k.
Forward Euler computes any subsequent state x_{k+1} according to:

$$x_{k+1} = x_k + \Delta t f(x_k, u_k) \quad \text{with} \quad k \in \mathbb{N}_0. \tag{A.3.2}$$

Several variants of 5-th order Runge-Kutta schemes exist. This thesis utilizes a variant of Kutta's family which favors more zero coefficients instead of simplicity of coefficients [LK65]:

$$\begin{aligned}
x_{k+1} &= x_k + \big(4\bar{k}_1 + (16+\sqrt{6})\bar{k}_5 + (16-\sqrt{6})\bar{k}_6\big)/36 \quad \text{with} \quad k \in \mathbb{N}_0, \\
\bar{k}_1 &= \Delta t f(x_k, u_k), \\
\bar{k}_2 &= \Delta t f(x_k + 4\bar{k}_1/11, u_k), \\
\bar{k}_3 &= \Delta t f\big(x_k + (9\bar{k}_1 + 11\bar{k}_2)/50, u_k\big), \\
\bar{k}_4 &= \Delta t f\big(x_k + (-11\bar{k}_2 + 15\bar{k}_3)/4, u_k\big), \\
\bar{k}_5 &= \Delta t f\Big(x_k + \big((81+9\sqrt{6})\bar{k}_1 + (255-55\sqrt{6})\bar{k}_3 + (24-14\sqrt{6})\bar{k}_4\big)/600, u_k\Big), \\
\bar{k}_6 &= \Delta t f\Big(x_k + \big((81-9\sqrt{6})\bar{k}_1 + (255+55\sqrt{6})\bar{k}_3 + (24+14\sqrt{6})\bar{k}_4\big)/600, u_k\Big).
\end{aligned} \tag{A.3.3}$$

A.4. Fundamentals of Constrained Nonlinear Optimization

This section summarizes the necessary and sufficient conditions for a general nonlinear program. The definitions are originally taken from [NW06] and are slightly rewritten in order to match the nomenclature of this thesis. Let $z \in \mathbb{R}^{n_z}$ define the optimization parameter vector. For a given cost function $J : \mathbb{R}^{n_z} \mapsto \mathbb{R}$, equality constraint $g : \mathbb{R}^{n_z} \mapsto \mathbb{R}^R$ and inequality constraint $h : \mathbb{R}^{n_z} \mapsto \mathbb{R}^S$, the general nonlinear program which seeks for a minimum cost function value while adhering to constraints is written as follows:

$$\begin{aligned}
\min_z \quad & J(z) \\
\text{subject to} \quad & \\
& h(z) = 0, \\
& g(z) \leq 0.
\end{aligned} \tag{A.4.1}$$

Functions $J(z)$, $h(z)$ and $g(z)$ are assumed to be continuously differentiable. The vector valued constraints can be expressed in terms of their scalar components $h_i : \mathbb{R}^{n_z} \mapsto \mathbb{R}$

with $i \in \mathcal{E}_e$ and $g_i : \mathbb{R}^{n_z} \mapsto \mathbb{R}$ with $i \in \mathcal{I}$. Hereby, $\mathcal{E}_e$ corresponds to the set of equality constraint indices and $\mathcal{I}$ to the set of inequality constraints. Their cardinality is according to the above nonlinear program formulation, in particular $|\mathcal{E}_e| = R$ and $|\mathcal{I}| = S$. The notation with index sets follows the one from [NW06] and simplifies the following definitions.
Any vector z which satisfies the constraints is said to be feasible and the corresponding set is denoted as feasible set Ω:

$$\Omega = \{z \in \mathbb{R}^{n_z} \mid h_i(z) = 0,\ i \in \mathcal{E}_e,\ g_i(z) \leq 0,\ i \in \mathcal{I}\}. \tag{A.4.2}$$

Numerical nonlinear program solvers often result in locally optimal solutions:

Definition A.4.1 (Local Solution)**.** A vector z^* is a *local solution* of (A.4.1) if $z^* \in \Omega$ and there is a neighborhood $\mathcal{N}$ of z^* such that $J(z) \geq J(z^*)$ for $z \in \mathcal{N} \cap \Omega$.

Definition A.4.2 (Strict Local Solution)**.** A vector z^* is a *strict local solution* of (A.4.1) if $z^* \in \Omega$ and there is a neighborhood $\mathcal{N}$ of z^* such that $J(z) > J(z^*)$ for $z \in \mathcal{N} \cap \Omega$ with $z \neq z^*$.

Of particular importance for the optimality conditions is the active inequality set $\mathcal{A}_\mathcal{I}(z)$. The active inequality set at a feasible vector $z \in \Omega$ consists of the inequality constraint indices i for which $g_i(z) = 0$ holds:

$$\mathcal{A}_\mathcal{I}(z) = \{i \in \mathcal{I} \mid g_i(z) = 0\}. \tag{A.4.3}$$

Consequently, at a point $z \in \Omega$ inequality $g_i(z)$ is said to be active if $g_i(z) = 0$ and inactive if $g_i(z) < 0$ is strictly satisfied.

A.4.1. Constraint Qualification

First- and second-order optimality conditions require that certain constraint qualifications with respect to $h(z)$ and $g(z)$ are met. These ensure that the geometry of the feasible set Ω near the solution z^* coincides with its algebraic formulation that constitutes a linearized approximation based on first-order information. Formally, the actual geometry which describes whether feasible sequences $(z)_{\tilde{k}}$ approaching z exist is given by the tangent cone:

$$T_\Omega(z) = \left\{d \in \mathbb{R}^{n_z} \mid \exists (z)_{\tilde{k}} \in \Omega, z_{\tilde{k}} \to z, \exists \{t_{\tilde{k}}\} \in \mathbb{R}^+, t_{\tilde{k}} \to 0, d = \lim_{\tilde{k}\to\infty} \frac{z_{\tilde{k}} - z}{t_{\tilde{k}}}\right\}. \tag{A.4.4}$$

In contrast, the algebraic description in terms of the set of linearized feasible directions is given by:

$$\mathcal{F}(z) = \left\{ d \in \mathbb{R}^{n_z} \;\middle|\; \begin{array}{ll} d^\intercal \nabla h_i(z) = 0, & \text{for all } i \in \mathcal{E}_e, \\ d^\intercal \nabla g_i(z) \leq 0, & \text{for all } i \in \mathcal{A}_\mathcal{I}(z) \end{array} \right\}. \tag{A.4.5}$$

Note, only the active inequality constraints are considered. In general it is $T_\Omega(z^*) \subset \mathcal{F}(z^*)$ at a feasible vector z^*. However, constraint qualifications ensure that $T_\Omega(z^*) = \mathcal{F}(z^*)$ holds. For brevity, only the two most common constraint qualifications are summarized in the following:

Definition A.4.3 (LICQ)**.** Given the vector $z \in \mathbb{R}^{n_z}$, the set of equality constraint indices $\mathcal{E}_e$ and the active inequality set $\mathcal{A}_\mathcal{I}(z)$. The *linear independence constraint qualification* (LICQ) holds if the set of active constraint gradients is linearly independent. The set of active constraints is given by $\{\nabla h_i(z) \text{ for } i \in \mathcal{E}_e, \nabla g_i(z) \text{ for } i \in \mathcal{A}_\mathcal{I}(z)\}$.

Definition A.4.4 (MFCQ)**.** The *Mangasarian-Fromovitz constraint qualification* (MFCQ) holds if there exists a vector $w \in \mathbb{R}^{n_z}$ such that

$$\begin{aligned} \nabla h_i(z)^\intercal w &= 0, \quad \text{for all } i \in \mathcal{E}_e, \\ \nabla g_i(z)^\intercal w &< 0, \quad \text{for all } i \in \mathcal{A}_\mathcal{I}(z), \end{aligned}$$

and the set of equality constraint gradients $\{\nabla h_i(z), i \in \mathcal{E}_e\}$ is linearly independent.

Note, LICQ implies MFCQ but the converse does not hold. Furthermore, MFCQ is a weaker condition than LICQ. For detailed information regarding the theory of constrained optimization and proofs refer to [NW06].

A.4.2. First-Order Optimality Conditions

Similar to the theory of equality constrained optimization problems, the Lagrangian function plays an important role in the definition of the optimality conditions. The Lagrangian of nonlinear program (A.4.1) is defined as follows:

$$\mathcal{L}(z, \lambda, \mu) = J(z) + \sum_{i \in \mathcal{E}} \lambda_i h_i(z) + \sum_{i \in \mathcal{I}} \mu_i g_i(z) = J(z) + \lambda^\intercal h(z) + \mu^\intercal g(z). \tag{A.4.6}$$

Hereby, $\lambda \in \mathbb{R}^R$ and $\mu \in \mathbb{R}^S$ constitute the Lagrange multiplier vectors of the equality and inequality constraints respectively.
In the following, first-order necessary conditions are provided which are often referred to as Karush-Kuhn-Tucker (KKT) conditions [NW06]. They relate first-order derivatives of $J(z)$, the constraints and Lagrange multipliers to each other at a local solution z^*.

Theorem A.4.1 (First-Order Necessary Conditions)**.** *Suppose that z^* is a local solution of* (A.4.1)*, that the functions $J(z)$, $h(z)$ and $g(z)$ are continuously differentiable, and that either LICQ or MFCQ hold at z^*. Then there are Lagrange multipliers λ^* and μ^*, with components λ_i^*, $i \in \mathcal{E}_e$, and μ_i^*, $i \in \mathcal{I}$, respectively, such that the following conditions are satisfied at (z^*, λ^*, μ^*):*

$$\nabla_z \mathcal{L}(z^*, \lambda^*, \mu^*) = 0, \tag{A.4.7a}$$
$$h_i(z^*) = 0, \quad \text{for all } i \in \mathcal{E}_e, \tag{A.4.7b}$$
$$g_i(z^*) \leq 0, \quad \text{for all } i \in \mathcal{I}, \tag{A.4.7c}$$
$$\mu_i^* \geq 0, \quad \text{for all } i \in \mathcal{I}, \tag{A.4.7d}$$
$$\mu_i^* g_i(z^*) = 0, \quad \text{for all } i \in \mathcal{I}. \tag{A.4.7e}$$

Hereby, $\nabla_z \mathcal{L}(z^*, \lambda^*, \mu^*)$ denotes the gradient of $\mathcal{L}(z, \lambda, \mu)$ as defined in (A.4.6) with respect to only the primal variables z and evaluated at z^*, λ^* and μ^*.

A.4.3. Second-Order Optimality Conditions

First-order conditions ensure that a move along a vector w from $\mathcal{F}(z^*)$ either increases $w^\intercal \nabla J(z^*) > 0$ or keeps $w^\intercal \nabla J(z^*) = 0$. However, it remains unclear whether this result also applies to the actual cost function $J(z)$ and not only to the linearized approximation. Hereby, second-order information resolves this issue by using the curvature of the Lagrangian. It is assumed that $J(z)$, $h(z)$ and $g(z)$ are twice continuously differentiable. The so-called critical cone $\mathcal{C}(z^*, \lambda*, \mu*)$ is defined in term of the set of linearized feasible directions $\mathcal{F}(z^*)$ in which inequalities that reveal a strictly positive Lagrange multiplier $\mu_i^* > 0$ are treated as equality constraint:

$$\mathcal{C}(z^*, \lambda^*, \mu^*) = \{w \in \mathcal{F}(z^*) \mid \nabla g_i(z^*)^\intercal w = 0, \text{ for all } i \in \mathcal{A}(z^*) \text{ with } \mu_i^* > 0\}. \quad \text{(A.4.8)}$$

In contrast to $\mathcal{F}(z^*)$, the critical cone ensures that directions w are subject to active constraints with $\mu_i^* > 0$ even if the cost function value changes slightly.

Theorem A.4.2 (Second-Order Necessary Conditions)**.** *Suppose that z^* is a local solution of* (A.4.1)*, that the functions $J(z)$, $h(z)$ and $g(z)$ are twice continuously differentiable, and that either LICQ or MFCQ hold at z^*. Let λ^* and μ^* be the Lagrange multiplier vectors for which the KKT conditions* (A.4.7) *are satisfied. Then*

$$w^\intercal \nabla_{zz}^2 \mathcal{L}(z^*, \lambda^*, \mu^*) w \geq 0, \quad \text{for all } w \in \mathcal{C}(z^*, \lambda^*, \mu^*). \quad \text{(A.4.9)}$$

Theorem A.4.3 (Second-Order Sufficient Conditions)**.** *Assume that for some feasible point $z^* \in \mathbb{R}$ there are Lagrange multiplier vectors λ^* and μ^* such that the KKT conditions* (A.4.7) *are satisfied. Suppose also that*

$$w^\intercal \nabla_{zz}^2 \mathcal{L}(z^*, \lambda^*, \mu^*) w > 0, \quad \text{for all } w \in \mathcal{C}(z^*, \lambda^*, \mu^*), w \neq 0. \quad \text{(A.4.10)}$$

Then z^* is a strict local solution for (A.4.1)*.*

Hereby, $\nabla_{zz}^2 \mathcal{L}(z^*, \lambda^*, \mu^*)$ denotes the Hessian of the Lagrangian $\mathcal{L}(z, \lambda, \mu)$ with respect to only the primal variables z and evaluated at z^*, λ^* and μ^*. Details and proofs are provided in several numerical optimization books such as [NW06].

B

Stability Definitions and Results

B.1. Lyapunov Functions for Stability

Recall the comparison functions as defined in Definition 3.2.3 as well as the asymptotic stability property from Definition 3.2.4 for the closed-loop system (3.2.7). Stability results are defined with respect to grid $t_{\mu,0} < t_{\mu,1} < \ldots < t_{\mu,n} < \ldots < \infty$ with $n \in \mathbb{N}_0$ as introduced in Section 3.2.3. For many applications, seeking for a proper function $\beta \in \mathscr{KL}$ with two input arguments is often replaced by the so-called Lyapunov function. The following definitions and theorems on asymptotic and practical stability using Lyapunov functions are from [GP17] and are slightly modified in terms of notation:

Definition B.1.1 (Lyapunov Function)**.** Consider system (3.2.7), a point $x_\mathrm{f} \in \mathcal{X}$ and let $S \subseteq \mathcal{X}$ be a subset of the state space. A function $V : S \mapsto \mathbb{R}_0^+$ is called a *Lyapunov function* on S if the following conditions are satisfied:

(i) There exist functions $\alpha_1, \alpha_2 \in \mathscr{K}_\infty$ such that

$$\alpha_1\big(\|x_\mu - x_\mathrm{f}\|\big) \leq V\big(x_\mu\big) \leq \alpha_2\big(\|x_\mu - x_\mathrm{f}\|\big) \tag{B.1.1}$$

holds for all $x_\mu \in S$.

(ii) There exists a function $\alpha_V \in \mathscr{K}$ such that

$$V\Big(\varphi_\mu\big(t_{\mu,n+1}, t_{\mu,n}, x_\mu(t_{\mu,n})\big)\Big) \leq V\big(x_\mu(t_{\mu,n})\big) - \alpha_V\big(\|x_\mu(t_{\mu,n}) - x_\mathrm{f}\|\big) \tag{B.1.2}$$

holds for all $x_\mu(t_{\mu,n}) \in S$ with $\varphi_\mu\big(t_{\mu,n+1}, t_{\mu,n}, x_\mu(t_{\mu,n})\big) \in S$ and $n \in \mathbb{N}_0$.

The following theorem states that a proper Lyapunov function candidate ensures asymptotic stability of system (3.2.7).

Theorem B.1.1 (Asymptotic Stability using Lyapunov Functions)**.** *Let x_f be a steady state of system* (3.2.7) *and assume there exists a Lyapunov function V on S. If S contains a ball $\mathscr{B}_\nu(x_\mathrm{f})$ with $\varphi_\mu\big(t_{\mu,n+1}, t_{\mu,n}, x_\mu(t_{\mu,n})\big) \in S$ for all $x_\mu(t_{\mu,n}) \in \mathscr{B}_\nu(x_\mathrm{f})$ and $n \in \mathbb{N}_0$ then x_f is locally asymptotically stable with $\eta = \alpha_2^{-1} \circ \alpha_1(\nu)$. If $S = Y$ holds for some forward invariant set $Y \subseteq \mathcal{X}$ containing x_f then x_f is asymptotically stable on Y. If $S = \mathcal{X}$ holds then x_f is globally asymptotically stable.*

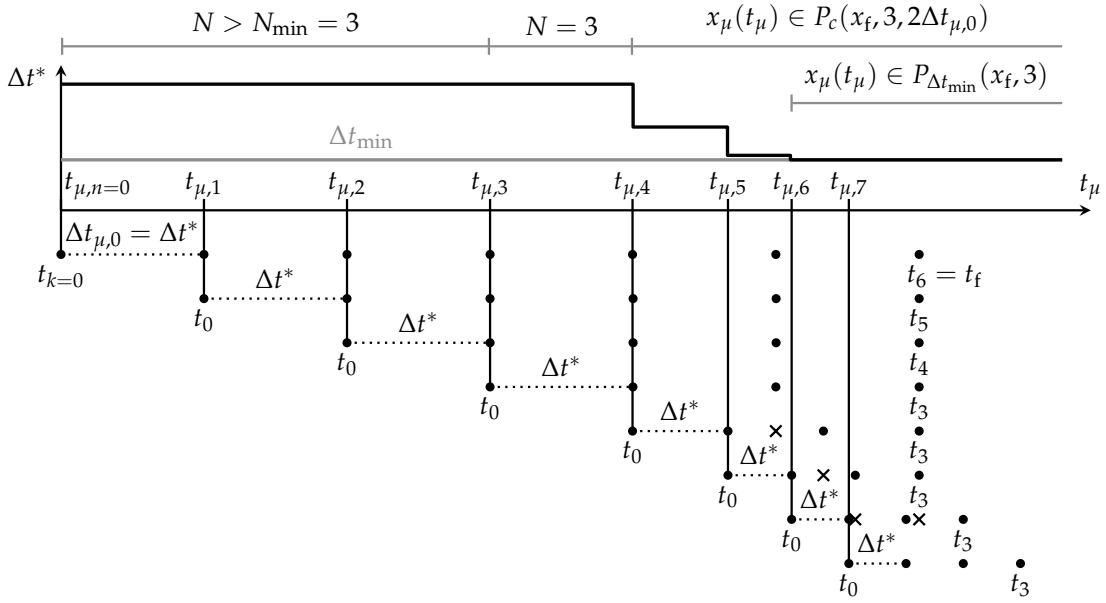


Figure B.1.: Illustration of practical stability for the uniform grids during closed-loop control. The points correspond to the discretization grid of the prediction (horizon) at closed-loop sampling times $t_{\mu,n}$. Crosses indicate a potential lack of recursive feasibility for states within the region $P_c(x_\mathrm{f}, N_{\min}, N_{\min}\Delta t_{\mu,0})$. As soon as $\Delta t^* = \Delta t_{\min}$ is reached, that is for $x_\mu(t_\mu) \in P_{\Delta t_{\min}}(x_\mathrm{f}, N_{\min})$, the cost no longer decreases and thus the temporal grid/horizon length is fixed and becomes receding. However, time-optimality and proper stabilization at x_f is no longer guaranteed.

Similar to Definition 3.2.5 a relaxed version based on Lyapunov functions is defined as follows:

Theorem B.1.2 (P-Practically Asymptotic Stability using Lyapunov Functions)**.** *Consider forward invariant sets Y and $P \subset Y$ and a point $x_\mathrm{f} \in P$. If there exists a Lyapunov function V on $S = Y \setminus P$ then x_f is P-practically asymptotically stable on Y.*

Note, these definitions are also made with respect to the grid points $t_{\mu,n}$ for $n \in \mathbb{N}$ and the continuous-time trajectory $x_\mu(t_\mu)$ is expected to be uniformly bounded. It is also possible to define the Lyapunov function in terms of comparison functions directly for the continuous-time case as proposed in [Lin+96].

B.2. Stability Results and Proofs

This section extends the stability results on time-optimal model predictive control with variable discretization grids as proposed in Chapter 5 respectively Chapter 9 and provides the corresponding proofs. Figure B.1 illustrates the different phases of closed-loop control with the uniform grids. First, the phase of the shrinking horizon reduces the grid by one in each iteration. Then, the horizon reaches $N = N_{\min}$ and the controller takes one more step with the previous time interval length until it enters $P_c(x_\mathrm{f}, N_{\min}, N_{\min}\Delta t_{\mu,0})$. Recursive feasibility and thus forward invariance can no longer be guaranteed as the grid size decreases in each step. As soon as the lower

bound $\Delta t^* \geq \Delta t_{\min}$ becomes active inside $P_c(x_\mathrm{f}, N_{\min}, N_{\min}\Delta t_{\mu,0})$, the cost no longer constitutes a suitable Lyapunov function candidate and further convergence to x_f cannot be ensured. The following theoretical results confirm these observations. Further results on some special configurations are presented as well, in particular $N_{\min} = 1$ leading to asymptotic stability under strict viability assumptions and the results on the non-uniform grid.
The following lemmas provide intermediate results which are later required to verify Lyapunov function candidates.

Lemma B.2.1. *Let $\Delta t^* \in \mathbb{R}^+$ denote the optimal time interval obtained from either* (4.1.8), (4.1.12) *or* (4.1.17) *with $x_u(t_0) = x_\mathrm{s} \in \mathbb{X}$, $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$ and grid size N. Further assume that Assumption 4.2.2 holds and let $P_{\Delta t_{\min}}(x_\mathrm{f}, N)$ be defined according to Definition 5.1.2. Then, relation*

$$\Delta t^* > \Delta t_{\min} \iff x_\mathrm{s} \notin P_{\Delta t_{\min}}(x_\mathrm{f}, N) \tag{B.2.1}$$

holds for $\Delta t_{\min} \geq 0$.

Proof. First, consider the case $x_\mathrm{s} \notin P_{\Delta t_{\min}}(x_\mathrm{f}, N) \implies \Delta t^* > \Delta t_{\min}$. The implication follows immediately from the definition of $P_{\Delta t_{\min}}(x_\mathrm{f}, N)$, in particular Definition 5.1.2, even for arbitrary Δt not subject to optimization but with $u \in \mathscr{U}^N(N\Delta t)$. By contraposition, the implication is equivalent to $\Delta t^* \leq \Delta t_{\min} \implies x_\mathrm{s} \in P_{\Delta t_{\min}}(x_\mathrm{f}, N)$. The optimal solution is feasible by Assumption 4.2.2 which implies that the control trajectory is admissible, in particular $u^* \in \mathscr{U}^N(N\Delta t^*)$. Note, the optimal control problems actually ensure $\Delta t^* \geq \Delta t_{\min}$ such that condition $\Delta t^* \leq \Delta t_{\min}$ is replaced by $\Delta t^* = \Delta t_{\min}$. Condition $\Delta t^* = \Delta t_{\min}$ implies $t_\mathrm{f} = N\Delta t_{\min}$ with $t_\mathrm{f} = N\Delta t^*$ and hence all requirements for $x_\mathrm{s} \in P_{\Delta t_{\min}}(x_\mathrm{f}, N)$ are met.
The second case $\Delta t^* > \Delta t_{\min} \implies x_\mathrm{s} \notin P_{\Delta t_{\min}}(x_\mathrm{f}, N)$ does not hold for arbitrary (non-optimal) Δt since control trajectories $u \in \mathscr{U}^N(N\Delta t)$ could exists which start and end in $P_{\Delta t_{\min}}(x_\mathrm{f}, N)$ but fulfill $\Delta t > \Delta t_{\min}$ (for example keeping the system at the steady state). However, to show that the implication holds for Δt^* subject to the time optimal control problems (4.1.8), (4.1.12) or (4.1.17), consider the contraposition $x_\mathrm{s} \in P_{\Delta t_{\min}}(x_\mathrm{f}, N) \implies \Delta t^* \leq \Delta t_{\min}$. If $x_\mathrm{s} \in P_{\Delta t_{\min}}(x_\mathrm{f}, N)$ holds, then there exists $u \in \mathscr{U}^N(N\Delta t_{\min})$ such that $x_\mathrm{f} = \varphi(t, x_\mathrm{s}, u)$ and $0 \leq t \leq N\Delta t_{\min}$ (see Definition 5.1.2). Solving the optimal control problems results in minimum-time solutions adhering to constraint $\Delta t_{\min} \leq \Delta t^*$ and hence the only feasible transition time for $x_\mathrm{s} \in P_{\Delta t_{\min}}(x_\mathrm{f}, N)$ is $t = N\Delta t_{\min}$ with $t = N\Delta t^*$. The existence of this particular $u \in \mathscr{U}^N(t)$ is confirmed by Assumption 4.2.2 even though it does not need to be unique. Consequently, $N\Delta t_{\min} = N\Delta t^*$ implies $\Delta t_{\min} = \Delta t^*$ which proves the original implication $x_\mathrm{s} \in P_{\Delta t_{\min}}(x_\mathrm{f}, N) \implies \Delta t^* \leq \Delta t_{\min}$.
Finally, equivalence (B.2.1) follows immediately since both implications are true. □

Lemma B.2.2. *Let Δt^* denote the optimal time interval obtained from either* (4.1.8), (4.1.12) *or* (4.1.17) *with $x_u(t_0) = x_\mathrm{s} \in \mathbb{X}$, $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$ and grid size N. Further assume that Assumption 4.2.2 and Assumption 3.1.1 hold. Then, the optimal cost function value $N\Delta t^*$ is bounded by $\alpha_1, \alpha_2 \in \mathscr{K}_\infty$ such that*

$$\alpha_1\big(\|x_\mathrm{s} - x_\mathrm{f}\|\big) \leq N\Delta t^* \leq \alpha_2\big(\|x_\mathrm{s} - x_\mathrm{f}\|\big) \tag{B.2.2}$$

holds for all $x_\mathrm{s} \in \mathbb{X}$ if $\Delta t_{\min} = 0$ or $x_\mathrm{s} \in \mathbb{X} \setminus P_{\Delta t_{\min}}(x_\mathrm{f}, N)$.

Proof. The proof starts with $\Delta t_{\min} = 0$. Under Assumption 4.2.2, the optimal value $\Delta t^* \geq 0$ constitutes a unique minimizer and is finite for any finite x_s and x_f. Furthermore, Assumption 4.2.2 ensures that the optimal control trajectory $u^*(t)$ with $u^* \in \mathscr{U}^N(N\Delta t^*)$ is admissible. Since by Assumption 4.2.2 the system dynamics error is negligible such that the relation between x_s and x_f is given as follows:

$$x_f = x_s + \int_0^{N\Delta t^*} f\big(x(t), u^*(t)\big)\,\mathrm{d}t \quad \text{with} \quad x(t=0) = 0. \tag{B.2.3}$$

Note, the original initial value $x(t=0) = x_s$ is substituted by $x(t=0) = 0$ and x_s. The vector field f is Lipschitz in its first argument by Assumption 3.1.1 and $u^*(t) \in \mathbb{U}$ is bounded since $\mathbb{U}$ is compact by definition. Consequently, $f(\cdot)$ is bounded on the close interval $[0, N\Delta t^*]$ and there exists a scalar $C > 0$ such that $\|f\big(x(t), u^*(t)\big)\| \leq C$ holds for $t \in [0, N\Delta t^*]$. For details on the relation between Lipschitz continuity and the boundedness of the function itself with respect to a bounded interval refer to [Eri+04]. In order to proof the lower bound in (B.2.2), consider (B.2.3), the triangle inequality in integral form and the boundedness of $f(\cdot)$:

$$\|x_f - x_s\| = \|\int_0^{N\Delta t^*} f\big(x(t), u^*(t)\big)\,\mathrm{d}t\| \tag{B.2.4}$$

$$\leq \int_0^{N\Delta t^*} \|f\big(x(t), u^*(t)\big)\|\,\mathrm{d}t \leq \int_0^{N\Delta t^*} C\,\mathrm{d}t = CN\Delta t^*. \tag{B.2.5}$$

Since $C > 0$, the lower bound is a $\mathscr{K}_\infty$ function and is given by:

$$N\Delta t^* \geq \frac{\|x_f - x_s\|}{C} = \alpha_1\big(\|x_s - x_f\|\big). \tag{B.2.6}$$

It is possible to interpret the lower bound from a rather technical point of view. Function $f(\cdot)$ defines the velocity in the state space such that an upper bound C represents the maximum velocity (here in sense of the ℓ_2-norm). So the minimum realizable transition time is estimated by the straight line distance $\|x_s - x_f\|$ divided by the maximum velocity C.

For the upper bound, consider that $\|x_s - x_f\| = 0$ implies $\|\int_0^{N\Delta t^*} f\big(x(t), u^*(t)\big)\,\mathrm{d}t\| = 0$. On the other hand, the norm of the integral must be strictly positive if $\|x_s - x_f\| > 0$. Note, the optimal control problem is feasible by Assumption 4.2.2. Consequently, there exists a scalar $\epsilon > 0$ such that

$$\|x_f - x_s\| = \|\int_0^{N\Delta t^*} f\big(x(t), u^*(t)\big)\,\mathrm{d}t\| \geq \epsilon \tag{B.2.7}$$

holds for $\|x_f - x_s\| > 0$. If $N\Delta t^*$ is finite, there also exist a scalar $\mu > 0$ which satisfies $\epsilon = \mu N\Delta t^*$ and a suitable upper bound is given by:

$$N\Delta t^* \leq \frac{\|x_f - x_s\|}{\mu} = \alpha_2\big(\|x_s - x_f\|\big). \tag{B.2.8}$$

The previous derivation assumed $N\Delta t^*$ to be finite and the unbounded case is now proven by contradiction. Consider the case in which the distance $\|x_f - x_s\|$ is unbounded. Now assume that $N\Delta t^*$ is still bounded. The integral over the closed interval

$[0, N\Delta t^*]$ with bounded integrand $f(\cdot)$ is bounded. Obviously, equality $\|x_\mathrm{f} - x_\mathrm{s}\| = \|\int_0^{N\Delta t^*} f\big(x(t), u^*(t)\big)\,\mathrm{d}t\|$ does not hold anymore which proves the claim. Consequently, $\|x_\mathrm{s} - x_\mathrm{f}\|$ is unbounded if $N\Delta t^*$ is unbounded and so (B.2.2) is satisfied for the unbounded case ($\mathscr{K}_\infty$ rather than $\mathscr{K}$).
Finally, consider the case $\Delta t_\mathrm{min} > 0$. Obviously, $\Delta t^* \geq \Delta t_\mathrm{min}$ holds due to constraint satisfaction (feasibility according to Assumption 4.2.2). By Lemma B.2.1, $x_\mathrm{s} \notin P_{\Delta t_\mathrm{min}}(x_\mathrm{f}, N)$ implies $\Delta t^* > \Delta t_\mathrm{min}$ and hence does not affect the previous results for the existence of α_1 and α_2. On the other hand, $x_\mu \in P_{\Delta t_\mathrm{min}}(x_\mathrm{f}, N)$ implies that $N\Delta t^*$ is constant such that α_1 and α_2 do not exists. However, this particular case is excluded in Lemma B.2.2. □

Chapter 5 provides both practical stability and asymptotic stability results whereas the latter are based on much stricter conditions. As common basis, results on practical stability are derived first:

Proof of Theorem 5.1.1. The proof relies on the Lyapunov stability theory as described in B.1. Let P abbreviate $P := P_c\big(x_\mathrm{f}, N_\mathrm{min}, (N_\mathrm{min} - 1)\Delta t^*\big)$ and define the successor state $x_\mu(t_{\mu,n+1}) := \varphi_\mu\big(t_{\mu,n+1}, t_{\mu,n}, x_\mu(t_{\mu,n})\big)$. It is necessary to find a function $V : \mathbb{X} \mapsto \mathbb{R}_0^+$ and $\alpha_1, \alpha_2 \in \mathscr{K}_\infty, \alpha_3 \in \mathscr{K}$ such that

$$\alpha_1\big(\|x_\mu(t_{\mu,0}) - x_\mathrm{f}\|\big) \leq V\big(x_\mu(t_{\mu,0})\big) \leq \alpha_2\big(\|x_\mu(t_{\mu,0}) - x_\mathrm{f}\|\big) \tag{B.2.9}$$

and

$$V\big(x_\mu(t_{\mu,n+1})\big) \leq V\big(x_\mu(t_{\mu,n})\big) - \alpha_V\big(\|x_\mu(t_{\mu,n}) - x_\mathrm{f}\|\big) \tag{B.2.10}$$

holds for all $x_\mu(t_{\mu,n}) \in \mathbb{X} \setminus P$ with $x_\mu(t_{\mu,n+1}) \in \mathbb{X} \setminus P$ and $n \in \mathbb{N}_0$.
As for model predictive control in general, the Lyapunov function candidate V is chosen as the optimal cost function value of the underlying optimal control problem. Lemma B.2.2 ensures the first condition (B.2.9) for all $x_\mu(t_{\mu,0}) \notin P_{\Delta t_\mathrm{min}}(x_\mathrm{f}, N)$. Since Theorem 5.1.1 demands to choose $0 \leq \Delta t_\mathrm{min} < \Delta t^*$ for the first optimal control problem, $\Delta t^* > \Delta t_\mathrm{min}$ implies $x_\mu(t_{\mu,0}) \notin P_{\Delta t_\mathrm{min}}(x_\mathrm{f}, N)$ according to Lemma B.2.1. Consequently, condition (B.2.9) is satisfied regardless of P.
The prove of the second condition (B.2.10) follows from the dynamic programming principle and hence its mathematical exposition is kept brief. For practical stability, the Lyapunov conditions must only be ensured for $x_\mu(t_{\mu,n}) \in \mathbb{X} \setminus P$. Assumption 4.2.2 ensures that the solution to the first optimal control problem at time $t_{\mu,0}$ is feasible and that an admissible control trajectory $u \in \mathscr{U}^N(N\Delta t^*)$ exists for grid size N. The optimal time interval is Δt^*.
First, consider the case $N > N_\mathrm{min}$. The optimal cost function value is $V\big(x_\mu(t_{\mu,n})\big) = N\Delta t^*$. Applying the principle of optimality results in

$$V\big(x_\mu(t_{\mu,n})\big) = N\Delta t^* = \Delta t^* + (N-1)\Delta t^* = \Delta t^* + V\big(x_\mu(t_{\mu,n+1})\big) \tag{B.2.11}$$

$$\iff V\big(x_\mu(t_{\mu,n+1})\big) - V\big(x_\mu(t_{\mu,n})\big) = -\Delta t^* \tag{B.2.12}$$

only if $t_{\mu,n+1} - t_{\mu,n} = \Delta t^*$ and the grid size at $t_{\mu,n+1}$ is reduced to $N - 1$. The former condition is ensured by inheriting sampling times from the open-loop solutions and Algorithm 5.1 explicitly reduces the horizon by one at $t_{\mu,n+1}$ for $n \in \mathbb{N}_0$.

To ensure Lyapunov condition (B.2.10), inequality $\Delta t^* \geq \alpha_V\big(\|x_\mu(t_{\mu,n}) - x_\mathrm{f}\|\big)$ must hold for $\alpha_V \in \mathscr{K}$. According to Lemma B.2.2, $\alpha_V = \alpha_1\big(\|x_\mu(t_{\mu,n}) - x_\mathrm{f}\|\big)/N$ provides a suitable upper bound for $x_\mu(t_{\mu,n}) \in \mathbb{X} \setminus P_{\Delta t_\mathrm{min}}(x_\mathrm{f}, N)$ and $N \geq N_\mathrm{min} > 0$ is ensured by definition. As stated previously, $x_\mu(t_{\mu,0}) \notin P_{\Delta t_\mathrm{min}}(x_\mathrm{f}, N)$ follows from Lemma B.2.1.
Note, the principle of optimality implies that Δt^* remains constant during closed-loop control as the horizon reduces by one in every iteration as long as $N > N_\mathrm{min}$. This in turn implies recursive feasibility and hence forward invariance on $\mathbb{X}$. As soon as $N = N_\mathrm{min}$ is reached, Algorithm 5.1 performs one more step with the previous Δt^*. Afterwards, Δt^* decreases in each step (see Figure 5.1b) until the lower bound $\Delta t^* \geq \Delta t_\mathrm{min}$ becomes active.
This in turn leads to a potential lack of recursive feasibility and hence forward invariance cannot be ensured in P which limits the stability results to $\mathbb{X} \setminus P$. □

Notice, if feasibility is assumed to hold anyway as soon as $N = N_\mathrm{min}$ is reached, the previous stability results apply to the larger set $\mathbb{X} \setminus P_{\Delta t_\mathrm{min}}(x_\mathrm{f}, N_\mathrm{min})$.

Proof of Corollary 5.2.1. By assumption, the initial grid size N_init ensures $|\Delta t_\mathrm{ref} - \Delta t^*| \leq \Delta t_\epsilon$ for the first optimal control problem. Similar to Theorem 5.1.1, closed-loop sampling times are inherited from the optimal control problems and the horizon is reduced by one (line 8 in Algorithm 5.2). Consequently, all subsequent sampling intervals are $t_{\mu,n+1} - t_{\mu,n} = \Delta t^*$ until $P_c\big(x_\mathrm{f}, N_\mathrm{min}, (N_\mathrm{min} - 1)\Delta t^*\big)$ is reached. The additional grid adaptation in line 5 of Algorithm 5.2 remains inactive. The remaining proof is identical to the one of Theorem 5.1.1. □

Proof of Corollary 5.3.1. Both closed-loop systems according to Theorem 5.1.1 and Corollary 5.2.1 are practically asymptotically stable on $\mathbb{X} \setminus P_c\big(x_\mathrm{f}, N_\mathrm{min}, (N_\mathrm{min} - 1)\Delta t^*\big)$. By assumption, the local linear controller ensures forward invariance and asymptotic stability on $\mathbb{X}_\mathrm{lin}$. Since control law (5.3.1) switches to the local controller as soon as $\mathbb{X}$ is reached, condition $P_c\big(x_\mathrm{f}, N_\mathrm{min}, (N_\mathrm{min} - 1)\Delta t^*\big) \subseteq \mathbb{X}_\mathrm{lin}$ immediately ensures that the whole dual-mode control system with steady state x_f is asymptotically stable. □

Lemma B.2.3. *For $x_\mu(t_{\mu,n}) = x_\mathrm{f}$, $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$, $N = 1$, $\Delta t_\mathrm{min} > 0$ and constant control trajectory $u(t) := u_0$, the optimal solution to* (4.1.8), (4.1.12) *with forward/backward Euler,* (C.2.7) *or* (C.2.8) *satisfies $f\big(x_\mathrm{f}, u_0^*\big) = 0$ if there exists $u_0^* \in \mathbb{U}$. Furthermore, assume that constraint qualifications and second-order necessary conditions hold.*

Proof. The set $\mathbb{U}$ is replaced by an algebraic description in any practical realization (see Chapter 4.2). Let this set be defined by $g(u_0) \leq 0$ with $g : \mathcal{U} \mapsto \mathbb{R}^S$.
In the following, the first-order optimality conditions for collocation via forward differences (4.1.8) are verified. Since $N = 1$, states are directly substituted and the remaining optimization parameters are u_0 and Δt. Accordingly, further state constraints are omitted. The system dynamics equality constraint is in this particular case $h(u_0) := \phi_\mathrm{fd}(x_\mathrm{f}, x_\mathrm{f}, u_0) - (x_\mathrm{f} - x_\mathrm{f})/\Delta t = \phi_\mathrm{fd}(x_\mathrm{f}, x_\mathrm{f}, u_0)$ by taking $\Delta t \geq \Delta t_\mathrm{min} > 0$ and start respectively final state x_f into account.
The Lagrangian with multipliers $\lambda_0 \in \mathbb{R}^p$, $\mu_0 \in \mathbb{R}^S$ and $\mu_1 \in \mathbb{R}$ is given by:

$$\mathcal{L}(u_0, \Delta t, \lambda_0, \mu_0, \mu_1) = \Delta t + \lambda_0^\intercal h(u_0) + \mu_0^\intercal g(u_0) + \mu_1(\Delta t_\mathrm{min} - \Delta t) \tag{B.2.13}$$

The first-order optimality conditions according to (A.4.7) for the optimal parameters (indicated by a star) are:

$$\nabla_{u_0} \mathcal{L}(\cdot) = \nabla_{u_0} \left(\lambda_0^{*\mathsf{T}} h(u_0^*)\right) + \nabla_{u_0} \left(\mu_0^{*\mathsf{T}} g(u_0^*)\right) = 0, \tag{B.2.14a}$$
$$\nabla_{\Delta t} \mathcal{L}(\cdot) = 1 - \mu_1^* = 0, \tag{B.2.14b}$$
$$h(u_0^*) = 0, \tag{B.2.14c}$$
$$\mu_0^{*\mathsf{T}} g(u_0^*) = 0, \tag{B.2.14d}$$
$$\mu_1^* (\Delta t_{\min} - \Delta t^*) = 0, \tag{B.2.14e}$$
$$g(u_0^*) \leq 0, \quad \Delta t^* \geq \Delta t_{\min}, \quad \mu_0^* \geq 0, \quad \mu_1^* \geq 0. \tag{B.2.14f}$$

Combining (B.2.14b) and (B.2.14e) confirms that the optimal time interval Δt^* satisfies $\Delta t^* = \Delta t_{\min}$. Condition (B.2.14c) keeps the system at x_f by enforcing $f(x_\mathrm{f}, u_0^*) = 0$. By assumption, an admissible $u_0^* \in \mathbb{U}$ exists, in particular $g(u_0^*) \leq 0$ holds. Note, this constraint does not hold for arbitrary control parameterizations but for constant controls since the system does not have any additional degrees of freedom to maintain $\phi_\mathrm{fd}(x_\mathrm{f}, x_\mathrm{f}, u_0^*) = 0$. The cases forward differences (4.1.5) and midpoint differences (4.1.6) are trivial. For Crank-Nicolson differences (4.1.7) verify that $\phi_\mathrm{fd}(x_\mathrm{f}, x_\mathrm{f}, u_0^*) = \big(f(x_\mathrm{f}, u_0^*) + f(x_\mathrm{f}, u_0^*)\big)/2 = f(x_\mathrm{f}, u_0^*)$ holds. Equations (B.2.14a) and (B.2.14d) are fulfilled by proper choices of λ_0^* and μ_0^*. For example, if control constraints $g(\cdot)$ are inactive, $\mu_0^* = \lambda_0^* = 0$ are possible solutions. Note, most nonlinear program solvers determine solutions according to the first-order optimality conditions. However, constraint qualification and second-order sufficient conditions according to Section A.4 must hold to ensure that solution is a strict minimum. These are assumed to hold here as the particular system dynamics equation is involved and these conditions must also hold for any nonlinear program solution.
The previous conditions imply that for initial state x_f, the optimal solution u_0^* keeps the system at the steady state x_f for interval $\Delta t^* = \Delta t_{\min}$.
The proofs for (4.1.12) with forward/backward Euler and (C.2.7) respectively (C.2.8) are omitted as the procedure is analogue as long as the control trajectory is constant. Note, it is crucial to ensure that the control representation for $N = 1$ is a constant value since otherwise the optimal control problem could chose different intermediate controls. For example in case of an integrator system, a potential system dynamics equality constraint $\tilde{h}(u_0, u_1) = 0.5\big(f(x_\mathrm{f}, u_0) + f(x_\mathrm{f}, u_1)\big) = 0$ could be solved by either choosing $u_0 = u_1 = 0$ or $u_0 = -u_1$ with u_1 arbitrarily. □

Proof of Proposition 5.1.1. The proof is based on the proof of Theorem 5.1.1. Since the minimum grid size is set to $N_{\min} = 1$, recursive feasibility is guaranteed until $x_\mu(t_{\mu,n}) \in \mathbb{X} \setminus P$ with $P := P_c\big(x_\mathrm{f}, N_{\min}, (N_{\min} - 1)\Delta t^*\big)$. Note, $P = \{x_\mathrm{f}\}$ holds in this case. Furthermore, the dynamic programming principle ensures that the system actually reaches x_f: As soon as the grid reduces to $N = N_{\min}$ for some $x_\mu(t_{\mu,n})$ the successor state is $x_\mu(t_{\mu,n+1}) = x_\mathrm{f}$. The Lyapunov conditions are ensured up to only $x_\mu(t_{\mu,n}) \in P_{\Delta t_{\min}}(x_\mathrm{f}, N_{\min})$ (Lemma B.2.2). Due to $N_{\min} = 1$ and condition $\Delta t_{\min} < \Delta t^*$, the system reaches x_f before the optimal time interval reduces to $\Delta t^* = \Delta t_{\min}$.
Since $\Delta t_{\min} > 0$, Lemma B.2.3 ensures recursive feasibility and asymptotic stability beyond x_f. □

Proof of Theorem 9.1.1. The solution to the first optimal control problem exists and is feasible according to Assumption 9.1.1. Constraints are invariant with respect to the non-uniform grid points according to Assumption 9.1.2. Since the grid is non-uniform, $\Delta t_{\min} = 0$ and the principle of optimality holds, applying control law (5.1.1) with any step size $t_{\mu,n+1} - t_{\mu,n}$ ensures recursive feasibility until reaching $\mathbb{X}_\mathrm{f} = \{x_\mathrm{f}\}$. As soon as x_f is reached, the optimal control problem could choose $u(t) \in \mathbb{U}$ arbitrarily as the transition time is $\Delta t_k = 0$ for all k. Accordingly, $\mathbb{X}_\mathrm{f}$ is excluded in the first part of the theorem. On the other hand, if the control law ensures Remark 3.2.1, $t_\mathrm{f} = \sum_k^{N-1} \Delta t_k = 0$ implies that the control law is $\mu(x_\mathrm{f}) = u_\mathrm{f}$ which in turn results in $f(x_\mathrm{f}, u_\mathrm{f}) = 0$. Consequently, the closed-loop system renders the set $\mathbb{X} \setminus \mathbb{X}_\mathrm{f}$ respectively $\mathbb{X}$ forward invariant.
It is further required to show that the optimal cost function value $V^*_\mathrm{dyn} = \sum_{k=0}^{N-1} \Delta t^*_k$ constitutes a Lyapunov function according to conditions (B.1.1) and (B.1.2). However, the proofs for both conditions are analogue to the proofs of Lemma B.2.1 and Theorem 5.1.1 by considering $\sum_{k=0}^{N-1} \Delta t^*_k$ rather than $N\Delta t^*$ and $\Delta t_{\min} = 0$.

□

C

Time-Optimal Control Methods – Extensions and Related Work

C.1. Related Methods from the Literature

This section provides a brief overview on the mathematical formulation of related work.

C.1.1. Time-Optimal Control via Time Transformation

Consider the continuous-time time-optimal control problem (3.2.2) defined on the interval $[t_0, t_f]$ with $t_0 = 0$ and varying final time t_f. The time transformation approach maps this interval to the unit interval [0,1]. However, the system dynamics equation is scaled by t_f to still account for the varying final time. Correspondingly, the realization of (3.2.2) with transformed time $\bar{t} \in [0, 1]$ is given by:

$$\min_{u(\bar{t}), t_f} \; 1 \tag{C.1.1}$$

subject to

$$\begin{aligned} &x_u(0) = x_\mu(t_\mu), \quad x_u(\bar{t}) \in \mathbb{X}, \quad u(\bar{t}) \in \mathbb{U}, \quad x_u(1) \in \mathbb{X}_f, \\ &\dot{x}_u(\bar{t}) = t_f f\big(x_u(\bar{t}), u(\bar{t})\big). \end{aligned}$$

Any indirect or direct method can be applied to solve (C.1.1). For example, direct methods discretize the interval $[0, 1]$ with respect to a fixed grid. For any solution to (C.1.1), the original time is recovered by setting $t = t_f \bar{t}$. More details are provided in [QD73; Jen+91; Teo+91; Zha+04].

C.1.2. Time-Optimal Model Predictive Control (TOMPC)

TOMPC solves the time-optimal control problem in terms of a two-stage optimization problem. The original approach is described for discrete-time systems [Van+11b; Van+11a]. However, it is common in MPC to replace the discrete-time formulation by sampled-data systems. With grid size N, let $u_k \in \mathcal{U}$ with $k = 0, 1, \ldots, N-1$ denote the discrete controls and $x_k \in \mathcal{X}$ with $k = 0, 1, \ldots, N$ the states, respectively.

The following quadratic form optimal control problem constitutes the inner nonlinear program with initial state $x_s \in \mathbb{X}$:

$$\min_{\substack{u_0,u_1,\dots,u_{N-1},\\ x_0,x_1,\dots,x_N}} \sum_{k=0}^{N-1} \left((x_k - x_f)^\top Q(x_k - x_f) + (u_k - u_f)^\top R(u_k - u_f)\right) \tag{C.1.2}$$

subject to

$$\begin{aligned} &x_0 = x_s, \quad x_N = x_f, \quad x_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \\ &x_{k+1} = f_d(x_k, u_k), \\ &k = 0, 1, \dots, N-1. \end{aligned}$$

Hereby, $x_f \in \mathbb{X}$ and $u_f \in \mathbb{U}$ denote the final state and final control, respectively. Matrices $Q \in \mathbb{R}^{p\times p}$ and $R \in \mathbb{R}^{q\times q}$ are positive definite weighting matrices. The sampled-data system $f_d(x_k, u_k)$ is obtained solving initial value problem $\varphi(\Delta t, x_k, u_k)$ for some desired time step $\Delta t \in \mathbb{N}$ numerically, for example, according to Section A.3. The set $\mathbb{X}_{\text{feas}}(N) = \{x_s \mid \text{nonlinear program (C.1.2) is feasible for } N \text{ and } x_s\}$ includes all states for which (C.1.2) with horizon N is feasible.
The outer optimization loop adapts the horizon N with respect to bounds $N_{\min} \in \mathbb{N}$ and $N_{\max} \in \mathbb{N}$:

$$\min_{N\in\mathbb{N}} N \tag{C.1.3}$$

subject to

$$N_{\min} \leq N \leq N_{\max}, \quad x_s \in \mathbb{X}_{\text{feas}}(N).$$

The resulting horizon N^* corresponds to the dead-beat solution, in particular the minimum number of steps to reach x_f.

C.1.3. Stabilizing Time-Optimal Model Predictive Control

A stabilizing time-optimal MPC scheme based on a weighted ℓ_1-norm is proposed in [Ver+17]. The nonlinear problem for the optimal control problem is given by:

$$\min_{\substack{u_0,u_1,\dots,u_{N-1},\\ x_0,x_1,\dots,x_N}} \sum_{k=0}^{N-1} \theta^k \|x_k - x_f\|_1 \tag{C.1.4}$$

subject to

$$\begin{aligned} &x_0 = x_s, \quad x_N = x_f, \quad x_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \\ &x_{k+1} = f_d(x_k, u_k), \\ &k = 0, 1, \dots, N-1. \end{aligned}$$

Hereby, $f_d(x_k, u_k)$ denotes a sampled-data system as described in the previous section. Horizon N is fixed but must be large enough to allow the system to transit from initial state $x_s \in \mathbb{X}$ to final state x_f, so at least $N \geq N^*$ where N^* denotes the dead-beat solution. Weighting parameter $\theta \in \mathbb{R}$ must be chosen appropriately [Ver+17].

The ℓ_1-norm cost function is non-smooth. By introducing slack variables $\bar{s}_k \in \mathbb{R}^p$ with state dimension p, $k = 0, 1, \ldots, N-1$ and two additional constraints per state vector, the problem is rewritten to:

$$\min_{\substack{u_0, u_1, \ldots, u_{N-1}, \\ x_0, x_1, \ldots, x_N, \\ \bar{s}_0, \bar{s}_1, \ldots, \bar{s}_N}} \sum_{k=0}^{N-1} \theta^k 1_p^\intercal \bar{s}_k \tag{C.1.5}$$

subject to

$$\begin{aligned} & x_0 = x_\mathrm{s}, \quad x_N = x_\mathrm{f}, \quad x_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \\ & x_{k+1} = f_\mathrm{d}(x_k, u_k), \quad x_k - x_\mathrm{f} \leq \bar{s}_k, \quad x_\mathrm{f} - x_k \leq \bar{s}_k, \\ & k = 0, 1, \ldots, N-1. \end{aligned}$$

This nonlinear program is smooth and $1_p = (1, 1, \ldots, 1)^\intercal$ denotes the vector of ones with dimension p.

C.2. Types of Hermite-Simpson Collocation

This sections provides further formulations of Hermite-Simpson collocation. As introduced in Section 4.1.3, Hermit-Simpson realizations differ in terms of compressed and uncompressed form as well as in terms of the control parameterization. In compressed form, the definition of the intermediate state $x_{k+0.5}$ according to (4.1.15) is substituted in the overall collocation constraint (4.1.14). Instead, the uncompressed form adds (4.1.15) as separate equality constraint to the nonlinear program. The control parameterizations are defined in Section 4.1.3 and Section A.2 (constant, mean, linear, quadratic). Note, the nonlinear program definitions for both the linear and the quadratic spline with $u_{k+0.5}$ as additional optimization parameters are identical, since the actual interpolation is performed after the solution is obtained from the nonlinear program. Figure C.1 shows an example of the different control representations and their corresponding state trajectories for an integrator system. The state trajectories for both quadratic and linear control splines are identical as their nonlinear program definitions coincide.

C.2.1. Hermite-Simpson Collocation with Intermediate Controls

Section 4.1.3 already describes the uncompressed form with $u_{k+0.5}$ as additional parameters. The corresponding compressed form is given by:

$$\min_{\substack{u_0, u_{0.5}, u_1, \ldots, u_N, \\ x_0, x_1, \ldots, x_N, \\ \Delta t}} N \Delta t \tag{C.2.1}$$

subject to

$$\begin{aligned} & x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_\mathrm{f}, \quad u_N \in \mathbb{U}, \\ & x_k \in \mathbb{X}, \quad x_{k+0.5} \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad u_{k+0.5} \in \mathbb{U}, \quad \Delta t \geq \Delta t_\mathrm{min}, \\ & x_{k+1} - x_k = \phi_\mathrm{hs}\big(x_k, x_{k+1}, u_k, u_{k+0.5}, u_{k+1}, \Delta t\big), \\ & k = 0, 1, \ldots, N-1. \end{aligned}$$

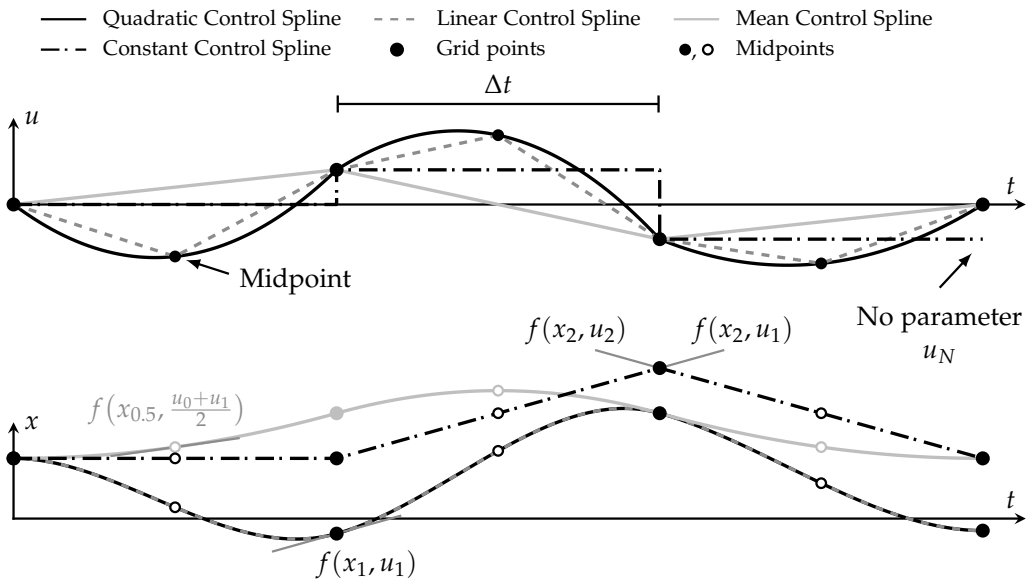


Figure C.1.: Example trajectories for different control representations of Hermite-Simpson collocation. The corresponding state trajectory is evaluated for an integrator system. The large filled circles are regular grid points whereas small circles represent midpoints. Non-filled circles are only subject to optimization in case of the uncompressed form.

Note, argument $x_{k+0.5}$ is omitted in $\phi_{\text{hs}}(\cdot)$ due to evaluation and substitution inside $\phi_{\text{hs}}(\cdot)$. The same applies to the other compressed forms defined below. Caching midpoint $x_{k+0.5}$ while evaluating $\phi_{\text{hs}}(\cdot)$ in a practical implementation increases efficiency.

The control trajectory on the interval $t \in [t_k, t_{k+1}]$ is either defined by a quadratic polynomial according to (4.1.16) or by the following linear representation:

$$u(t) := \begin{cases} u_k + (t - t_k)\big(u_{k+0.5} - u_k\big) & \text{for } t_k \leq t < t_{k+0.5} \\ u_{k+0.5} + (t - t_{k+0.5})\big(u_{k+1} - u_{k+0.5}\big) & \text{for } t_{k+0.5} \leq t \leq t_{k+1} \end{cases}. \tag{C.2.2}$$

C.2.2. Hermite-Simpson Collocation with Mean Control Spline

Another linear control spline representation follows from substitution of $u_{k+0.5} = 0.5(u_k + u_{k+1})$ in (4.1.14). To avoid any confusion with the previous linear spline (C.2.2), this representation is referred to as mean control spline. The control trajectory on the interval $t \in [t_k, t_{k+1}]$ is defined by:

$$u(t) := u_k + (t - t_k)\big(u_{k+1} - u_k\big). \tag{C.2.3}$$

The corresponding nonlinear program in compressed form is:

$$\min_{\substack{u_0,u_1,\ldots,u_N,\\ x_0,x_1,\ldots,x_N,\\ \Delta t}} N\Delta t \tag{C.2.4}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_\mathrm{f}, \quad u_N \in \mathbb{U},\\
&x_k \in \mathbb{X}, \quad x_{k+0.5} \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad u_{k+0.5} \in \mathbb{U}, \quad \Delta t \geq \Delta t_\mathrm{min},\\
&x_{k+1} - x_k = \phi_\mathrm{hs}\big(x_k, x_{k+1}, u_k, 0.5(u_k + u_{k+1}), u_{k+1}, \Delta t\big),\\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

Accordingly, the uncompressed form is given by:

$$\min_{\substack{u_0,u_1,\ldots,u_N,\\ x_0,x_{0.5},x_1,\ldots,x_N,\\ \Delta t}} N\Delta t \tag{C.2.5}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_\mathrm{f}, \quad u_N \in \mathbb{U},\\
&x_k \in \mathbb{X}, \quad x_{k+0.5} \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad u_{k+0.5} \in \mathbb{U}, \quad \Delta t \geq \Delta t_\mathrm{min},\\
&x_{k+1} - x_k = \phi_\mathrm{hs}(x_k, x_{k+0.5}, x_{k+1}, u_k, u_{k+0.5}, u_{k+1}, \Delta t),\\
&x_{k+0.5} = 0.5(x_k + x_{k+1}) + \Delta t/8\big(f(x_k, u_k) - f(x_{k+1}, u_{k+1})\big),\\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

C.2.3. Hermite-Simpson Collocation with Constant Controls

Another representation defines the control trajectory as piecewise constant. In particular, the control on the interval $t \in [t_k, t_{k+1}]$ is given by:

$$u(t) := u_k = \text{constant} \tag{C.2.6}$$

which results in the following compressed form:

$$\min_{\substack{u_0,u_1,\ldots,u_{N-1},\\ x_0,x_1,\ldots,x_N,\\ \Delta t}} N\Delta t \tag{C.2.7}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_\mathrm{f},\\
&x_k \in \mathbb{X}, \quad x_{k+0.5} \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad \Delta t \geq \Delta t_\mathrm{min},\\
&x_{k+1} - x_k = \phi_\mathrm{hs}(x_k, x_{k+1}, u_k, u_k, u_k, \Delta t),\\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

Finally, the uncompressed form for piecewise constant controls is given by:

$$\min_{\substack{u_0,u_1,\dots,u_{N-1},\\ x_0,x_{0.5},x_1,\dots,x_N,\\ \Delta t}} \quad N\Delta t \tag{C.2.8}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_\mathrm{f},\\
&x_k \in \mathbb{X}, \quad x_{k+0.5} \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad \Delta t \geq \Delta t_\mathrm{min},\\
&x_{k+1} - x_k = \phi_\mathrm{hs}(x_k, x_{k+0.5}, x_{k+1}, u_k, u_k, u_k, \Delta t),\\
&x_{k+0.5} = 0.5(x_k + x_{k+1}) + \Delta t/8\big(f(x_k, u_k) - f(x_{k+1}, u_k)\big),\\
&k = 0, 1, \dots, N-1.
\end{aligned}$$

C.3. Minimum Number of Control Interventions

Section 9.2 proposes an algorithm to find the minimum number of control interventions in terms of the non-uniform grid. A control intervention is referred to as shooting interval with a single constant control. Ideally, for bang-singular-bang systems these control interventions N correspond to the number of required switching points respectively the minimum number of intervals. The proposed algorithm erases similar consecutive controls up to convergence.
An alternative strategy consists of reducing the overall cost by testing $N+1$ and $N-1$ in every interval (multi-stage optimization) [Rös+17i]. The key idea in Algorithm C.1 is to reduce grid points which do not decrease the overall cost (transition time).
Let N^* denote the grid size related to the minimum number of required control interventions initial state $x_\mu(t_{\mu,n})$ and final state state x_f. The minimal cost according to (9.1.1) is denoted by $V^*_{\mathrm{dyn},N}$. Here, subscript N emphasizes the dependency on the grid size N. Other arguments (states, control, time) are omitted for a better readability. The cost $V^*_{\mathrm{dyn},N}$ corresponds to the optimal transition time. Furthermore, the grid size to provide a (sub-) optimal but still feasible trajectory is denoted by $N_\mathrm{crit} \leq N^*$ and leads to $V^*_{\mathrm{dyn},N_\mathrm{crit}} \geq V^*_{\mathrm{dyn},N^*}$. Consequently, for any $N < N_\mathrm{crit}$ problem (9.1.1) lacks sufficient degrees of freedom in control and is infeasible. The optimal cost for an infeasible solution is set to $V^*_{\mathrm{dyn},N} := \infty$. On the other hand, problem (9.1.1) is overdetermined for $N > N^*$ but still exhibits the same optimal cost $V^*_{\mathrm{dyn},N} = V^*_{\mathrm{dyn},N^*}$ (assuming proper convergence). This observation suggests adjusting the length iteratively by comparing $V^*_{\mathrm{dyn},N-1}$, $V^*_{\mathrm{dyn},N}$ and $V^*_{\mathrm{dyn},N+1}$ for a given N. The following algorithm does not include an outer optimization loop as in Section 9.2 as the multi-optimization is costly. However, it is often favorable to run the algorithm repeatedly in advance (offline) to determine the minimum number of control interventions online. As defined in Section 9.2, $\mathcal{P}_{\mathrm{local},N}$ denotes the current parameter set subject to optimization with N intervals. Inserting new parameters to the set is achieved by inserting a new tuple of time interval and control in between interval $\max\{\Delta t_k | \forall k\}$. Time intervals are interpolated with first-order hold and controls with zero-order hold. As noted before, the procedure is either applied online during closed-loop control, or offline by the repeated invocation up to convergence of N. Note, the infeasibility check in line 12 does not

Algorithm C.1.: Alternative method for adaptation of the number of control interventions.

procedure MULTISTAGESTEP($\mathcal{P}_{\text{local},N}, x_\mu(t_{\mu,n}), x_\text{f}$)
  $\{V^*_{\text{dyn},N}, \mathcal{P}_{\text{local},N}\} \leftarrow$ Solve nonlinear program w.r.t. $\mathcal{P}_{\text{local},N}$ ▷ solve (9.1.1)
  if $V^*_{\text{dyn},N}$ is feasible **then**
    $\mathcal{P}_{\text{local},N-1} \leftarrow$ Resize $\mathcal{P}_{\text{local},N}$ to size $N-1$
    $\{V^*_{\text{dyn},N-1}, \mathcal{P}_{\text{local},N-1}\} \leftarrow$ Solve nonlinear program w.r.t. $\mathcal{P}_{\text{local},N-1}$
    if $V^*_{\text{dyn},N-1} \leq V^*_{\text{dyn},N}$ **then**
      return $\mathcal{P}_{\text{local},N-1}$
  $\mathcal{P}_{\text{local},N+1} \leftarrow$ Resize $\mathcal{P}_{\text{local},N}$ to size $N+1$
  $\{V^*_{\text{dyn},N+1}, \mathcal{P}_{\text{local},N+1}\} \leftarrow$ Solve nonlinear program w.r.t. $\mathcal{P}_{\text{local},N+1}$
  if $V^*_{\text{dyn},N+1} < V^*_{\text{dyn},N}$ **then**
    return $\mathcal{P}_{\text{local},N+1}$
  if $V^*_{\text{dyn},N}$ is infeasible **then**
    Repeat procedure with significantly larger N
  return $\mathcal{P}_{\text{local},N}$

trigger in case the solution is already feasible, in particular, if $N \geq N_\text{crit} - 1$. Otherwise, a linear ($N \leftarrow N+1$) or more advanced search strategy increases N adequately. Generally, but especially for online usage, the algorithm of Section 9.2 is preferred, since it does not require the nonlinear program solver to solve up to convergence before intervals can be eliminated.

C.4. Unconstrained Nonlinear Least-Squares Approximation

This section investigates the application of unconstrained optimization techniques to time-optimal model predictive control with time-variant discretization grids. The benefit of utilizing unconstrained optimization techniques is that its efficient implementation is usually simple to achieve and that a wide range of general purpose optimization software is available. Especially, if the unconstrained optimization problem is formulated in terms of a nonlinear least-squares cost function, the Hessian can be efficiently approximated from first-order derivatives. Parts of this section have been published in [Rös+14a; Rös+16a].

C.4.1. Approximative Time-Optimal Control

The least-squares formulations in this chapter address the uniform and quasi-uniform grids as introduced in chapters 4 and 6. Note, the time-optimal control specialization for bang-singular-bang control systems according to Chapter 9 cannot be transformed to a least-squares problem since squaring its cost function would immediately lead to the quasi-uniform grid formulation. The description and investigation focus on the finite difference variants for both the global uniform and quasi-uniform grid.

Quasi-Uniform Grid

First, nonlinear program (6.2.4) with grid (6.1.1) is approximated by an unconstrained least-squares optimization problem. Transforming nonlinear program (6.2.4) to standard form leads to:

$$\min_{\substack{u_0,u_1,\dots,u_{N-1},\\ x_0,x_1,\dots,x_N,\\ \Delta t_0,\Delta t_1,\dots,\Delta t_{N-1}}} \sum_{k=0}^{N-1} \Delta t_k^2 \tag{C.4.1}$$

subject to

$$\begin{aligned} & x_0 = x_\mu(t_\mu), \quad x_N = x_\mathrm{f}, \quad \Delta t_k > 0, \\ & g\big(x_k, u_k\big) \leq 0, \\ & h\big(x_k, u_k, \Delta t_k, x_{k+1}\big) = 0, \\ & k = 0, 1, \dots, N-1. \end{aligned}$$

The system dynamics equality constraint is transformed by $h : \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X} \mapsto \mathcal{X}$ into standard form:

$$h\big(x_k, u_k, \Delta t_k, x_{k+1}\big) = \frac{x_{k+1} - x_k}{\Delta t_k} - \phi_\mathrm{fd}\big(x_k, x_{k+1}, u_k\big). \tag{C.4.2}$$

State constraints $x_k \in \mathbb{X}$ and control constraints $u_k \in \mathbb{U}$ are combined into the vector valued inequality term $g : \mathcal{X} \times \mathcal{U} \mapsto \mathbb{R}^R$. In case additional equality constraints are required, they can be added to $h(\cdot)$ which is omitted here for the sake of briefness and thus $\mathbb{R}^S := \mathcal{X}$.
The cost function in (C.4.1) already constitutes a least-squares objective. Quadratic penalty functions approximate constraints according to [NW06]. Notice, other approximations like barrier, augmented Lagrangian or exact penalty methods [KM00] contain terms that are not destined to be squared. This description should not be confused with constrained least-squares optimization problems in which only the cost function is of least-squares type but not the constraints.
The equality constraint is expressed by a quadratic penalty function $\psi : \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X} \times \mathbb{R}^+ \mapsto \mathbb{R}_0^+$. For the sake of readability, arguments of $h(\cdot)$ are omitted in the following:

$$\psi\big(x_k, u_k, \Delta t_k, x_{k+1}, \sigma_1\big) = \sigma_1 h(\cdot)^\intercal h(\cdot) = \sigma_1 ||h(\cdot)||_2^2. \tag{C.4.3}$$

Hereby, $\sigma_1 \in \mathbb{R}^+$ denotes a scalar weight. The inequality constraints are approximated by the following one-sided quadratic penalty function $\chi : \mathcal{X} \times \mathcal{U} \times \mathbb{R}^+ \mapsto \mathbb{R}_0^+$:

$$\chi\big(x_k, u_k, \sigma_2\big) = \sigma_2 \Big\| \underbrace{\max\Big(0, g\big(x_k, u_k\big)\Big)}_{\tilde{g}\left(x_k, u_k\right)} \Big\|_2^2. \tag{C.4.4}$$

The max-operator applies row-wise and $\sigma_2 \in \mathbb{R}^+$ denotes a scalar weight. Constraints $x_0 = x_\mu(t_\mu)$ and $x_N = x_\mathrm{f}$ are eliminated by substitution as explained in Chapter 7. Note, an important implication of this substitution is that reaching the final state is not affected by the choice of a penalty weight, for example σ_1. Only the accuracy of

the system dynamics approximation during the transition is affected by σ_2. Another interesting observation is that even though $\Delta t_k > 0$ could be included as a penalty, the difference quotient in (C.4.2) already bounds the domain to $\mathbb{R}^+$ implicitly (as long as $\Delta t_k > 0$ holds at initialization). This is enforced by the Levenberg-Marquardt algorithm as described in Section C.4.2 or any other trust-region solver.
With (C.4.1), (C.4.3) and (C.4.4) the approximative, unconstrained optimization problem is defined as follows:

$$\min_{\substack{u_0,u_1,\dots,u_{N-1},\\ x_1,x_2,\dots,x_{N-1},\\ \Delta t_0,\Delta t_1,\dots,\Delta t_{N-1}}} \sum_{k=0}^{N-1} \big(\Delta t_k^2 + \psi(x_k, u_k, \Delta t_k, x_{k+1}, \sigma_1) + \chi(x_k, u_k, \sigma_2)\big). \tag{C.4.5}$$

Global Uniform Grid

This section approximates nonlinear program (4.1.8) with grid (4.1.1) by an unconstrained least-squares optimization problem. Most of the steps of the previous section are analogue. Nonlinear program (4.1.8) is similar to form (C.4.1) in case $\Delta t_k = \Delta t$ for all $k = 0, 1, \dots, N-1$. Only the cost function cannot be inherited due to the squared approximation provided in Chapter 6. The uniform grid cost is $N\Delta t$ which is not a least-squares function. Squaring $N\Delta t$ does not change the minimizer of the nonlinear program, since Δt is strictly positive by definition. With previously defined penalty terms and constraint elimination, the unconstrained optimization problem is given by:

$$\min_{\substack{u_0,u_1,\dots,u_{N-1},\\ x_1,x_2,\dots,x_{N-1},\\ \Delta t}} (N\Delta t)^2 + \sum_{k=0}^{N-1} \big(\psi(x_k, u_k, \Delta t, x_{k+1}, \sigma_1) + \chi(x_k, u_k, \sigma_2)\big). \tag{C.4.6}$$

Remark C.4.1. Note, keeping N in the cost function results in an implicit weight on the transition time. Usually, Δt is small and since only a single Δt is defined for the whole trajectory, the presence of N balances the overall cost value with respect to σ_1 and σ_2. However, without loss of generality, N could be set to $N = 1$ but (initial) weights σ_1 and σ_2 are possibly required to be much smaller to still ensure a quasi time-optimal transition.

C.4.2. Solution to the Nonlinear Least-Squares Problem

This section addresses the solution to the nonlinear least-squares problem.

Remark C.4.2. Transforming nonlinear programs into equivalent unconstrained optimization problems is a common procedure and the minimizer of (C.4.5) coincides with the actual minimizer of (C.4.1) only if $\sigma_1 \to \infty$ and $\sigma_2 \to \infty$ [NW99]. However, large weights introduce ill-conditioned properties and therefore the underlying solver does not accept adequate step sizes. The problem might not converge properly. A common remedy is to increase σ_1 and σ_2 successively with each iteration. This way, the intermediate solutions first converge towards a solution that violates the constraints which is then refined afterwards.

According to their nonlinear program counterparts, the approximative least-squares optimal control problems are realized for closed-loop control with grid adaptation according to Algorithm 5.2 (global uniform grid) respectively 6.1 (quasi-uniform grid). Hereby, the weight adaptation scheme seamlessly integrates into the superimposed grid adaptation algorithm by synchronization with iteration i. Within each iteration i, the weights are updated according to $\sigma^{(i+1)} = \eta_{\mathrm{LS}}\sigma^{(i)}$ for both σ_1 and σ_2 with growth rate $\eta_{\mathrm{LS}} \geq 1$. In the remainder, initial weights are set to $\sigma_1^{\{0\}} = \sigma_2^{\{0\}} = 2$ and $\eta_{\mathrm{LS}} = 1.2$. Particular choices for the initial weights and the growth rate depend on the application. Obviously, σ is adapted only I_{adapt} times, but the closed-loop realization refines the solution during runtime and hence the actual minimizer for $\sigma_1 \to \infty$ and $\sigma_2 \to \infty$ might be waived in exchange for a suboptimal but more efficient solution.

For ease of notation, both unconstrained optimization problems (C.4.5) and (C.4.6) are assumed to be in a common standard least-squares form:

$$\min_z \|F(z)\|_2^2 \tag{C.4.7}$$

In case of the local grid (C.4.5), the optimization parameter vector $z \in \mathbb{R}^{n_z}$ with $n_z = (N-2)p + (N-1)(q+1)$ and cost vector $F(z)$ are defined as

$$z := \left(u_0^\mathsf{T}, \Delta t_0, x_1^\mathsf{T}, u_1^\mathsf{T}, \Delta t_1, \ldots, x_{N-1}^\mathsf{T}, u_{N-1}^\mathsf{T}, \Delta t_{N-1}\right)^\mathsf{T}, \tag{C.4.8}$$

$$F(z) := \left(\Delta t_0, \tilde{\sigma}_1 h^\mathsf{T}, \tilde{\sigma}_2 \tilde{g}^\mathsf{T}, \Delta t_1, \tilde{\sigma}_1 h^\mathsf{T}, \tilde{\sigma}_2 \tilde{g}^\mathsf{T}, \ldots, \Delta t_{N-1}, \tilde{\sigma}_1 h^\mathsf{T}, \tilde{\sigma}_2 \tilde{g}^\mathsf{T}\right)^\mathsf{T} \tag{C.4.9}$$

and in case of the global grid (C.4.6) as

$$z := \left(u_0^\mathsf{T}, x_1^\mathsf{T}, u_1^\mathsf{T}, \ldots, x_{N-1}^\mathsf{T}, u_{N-1}^\mathsf{T}, \Delta t\right)^\mathsf{T}, \tag{C.4.10}$$

$$F(z) := \left(N\Delta t, \tilde{\sigma}_1 h^\mathsf{T}, \tilde{\sigma}_2 \tilde{g}^\mathsf{T}, \ldots, \tilde{\sigma}_1 h^\mathsf{T}, \tilde{\sigma}_2 \tilde{g}^\mathsf{T}\right)^\mathsf{T} \tag{C.4.11}$$

with $n_z = (N-2)p + (N-1)q + 1$. Arguments are omitted for the sake of readability and recapitulate that initial and final state constraints (parameters) are eliminated by substitution. The repetition of $h(\cdot)$ and $\tilde{g}(\cdot)$ in (C.4.11) is performed $N-1$ times for the related variables at grid points $k = 0, 1, \ldots, N-1$. Furthermore, $\tilde{\sigma}_1 = \sqrt{\sigma_1}$ and $\tilde{\sigma}_2 = \sqrt{\sigma_2}$ holds.

The literature reports many iterative methods for solving nonlinear least-squares problem (C.4.7) such as the popular Gauss-Newton, Levenberg-Marquardt algorithm or trust-region strategies like Dogleg, two-dimensional subspace method and algorithms based on conjugate gradients [NW06].

In the remainder, the Levenberg-Marquardt algorithm is utilized due to its proper balance between robustness and efficiency. It is also well established in efficient implementations for sparse bundle adjustment and simultaneous localizing and mapping [LA09; Küm+11]. Levenberg-Marquardt is often referred to as progenitor of trust-region methods. Similar to trust-region methods and in contrast to Gauss-Newton, the algorithm enforces new parameter updates to enforce a reduction of the overall cost in every iteration.

In order to solve (C.4.7), the nonlinear cost function vector $F(z)$ is replaced by a linear approximation, for example, by utilizing the first two terms of the Taylor series expansion:

$$F(z + \Delta z) \approx F(z) + \mathrm{D}\,F(z). \tag{C.4.12}$$

Hereby, Δz denotes the increment of the optimization vector at the current point z and $\mathrm{D}\,F(z)$ constitutes the Jacobian matrix of $F(z)$ with respect to z. The following auxiliary optimization problem seeks the optimal increment Δz to approximate (C.4.7) in the vicinity of a given z:

$$\min_{\Delta z} \underbrace{\|F(z) + \mathrm{D}\,F(z)\Delta z\|_2^2 + \lambda_{\mathrm{LM}}\|\Delta z\|_2^2}_{\tilde{F}(\Delta z)}. \tag{C.4.13}$$

The term $\lambda_{\mathrm{LM}}\|\Delta z\|_2^2$ with damping factor $\lambda_{\mathrm{LM}} \in \mathbb{R}_0^+$ constitutes a regularization term with respect to the step Δz. Small $\lambda_{\mathrm{LM}} \approx 0$ favor the actual cost functions, whereas larger λ_{LM} prefer small steps Δz. The method migrates to the Gauss-Newton approach for the special choice $\lambda_{\mathrm{LM}} = 0$. The Levenberg-Marquardt algorithm adapts λ_{LM} to reduce the step size Δz until the overall cost $\|F(z)\|_2^2$ is decreased. Resolving the squared terms in (C.4.13) leads to:

$$\begin{aligned}
\tilde{F}(\Delta z) &= \big(F(z) + \mathrm{D}\,F(z)\Delta z\big)^\intercal \big(F(z) + \mathrm{D}\,F(z)\Delta z\big) + \lambda_{\mathrm{LM}}\Delta z^\intercal \Delta z \\
&= \underbrace{F(z)^\intercal F(z)}_{\tilde{d}} + 2\underbrace{F(z)^\intercal \mathrm{D}\,F(z)}_{\tilde{p}}\Delta z + \Delta z^\intercal \underbrace{\mathrm{D}\,F(z)^\intercal \mathrm{D}\,F(z)}_{H}\Delta z + \lambda_{\mathrm{LM}}\Delta z^\intercal \Delta z \\
&= \tilde{d} + 2\tilde{p}\Delta z + \Delta z^\intercal (H + \lambda_{\mathrm{LM}} I_{n_z})\Delta z.
\end{aligned} \tag{C.4.14}$$

Hereby, I_{n_z} denotes the $n_z \times n_z$ identity matrix. The first-order necessary conditions for an unconstrained optimization problem require

$$\nabla\,\tilde{F} = 2\tilde{p} + 2(H + \lambda_{\mathrm{LM}} I_{n_z})\Delta z \stackrel{!}{=} 0 \tag{C.4.15}$$

which leads to the following linear system:

$$(H + \lambda_{\mathrm{LM}} I_{n_z})\Delta z = -\tilde{p}. \tag{C.4.16}$$

Here, H denotes the Hessian that itself depends on the Jacobian $\mathrm{D}\,F(z)$. The Jacobian $\mathrm{D}\,F(z)$ is represented as a sparse matrix and is efficiently computed using the hypergraph as described in Chapter 7. Consequently, the Hessian H is sparse as well. Several methods exists to solve linear system (C.4.16) efficiently. In the remainder, linear system (C.4.16) is solved by a direct *LLt* cholesky factorization with fill-in reducing. The algorithm is available as part of the C++ math library *Eigen* [GJ+10]. As indicated before, the auxiliary optimization problem (C.4.13), in particular linear system (C.4.16), must be solved in every iteration. By denoting the solution to (C.4.16) by Δz^* and by adding the iteration index κ as superscript to z, the iterative refinement of the optimal parameter vector proceeds as follows:

$$z^{\{\kappa+1\}} = z^{\{\kappa\}} + \Delta z^* \tag{C.4.17}$$

until absolute and relative tolerance thresholds are satisfied. However, before a certain Δz^* is considered as solution, the damping factor in (C.4.13) is adapted until $z^{\{\kappa\}} + \Delta z^*$ actually reduces the overall cost. The entire algorithm is listed in Appendix E.2.3. Levenberg-Marquardt locally migrates to the Gauss-Newton algorithm and hence the algorithm converges locally linearly [NW06].
Note, lower and upper bounds on the optimization parameters are included in $g(\cdot)$. It is also possible to explicitly consider bounds by augmenting Levenberg-Marquardt with an additional line search as proposed in [Kan+05].

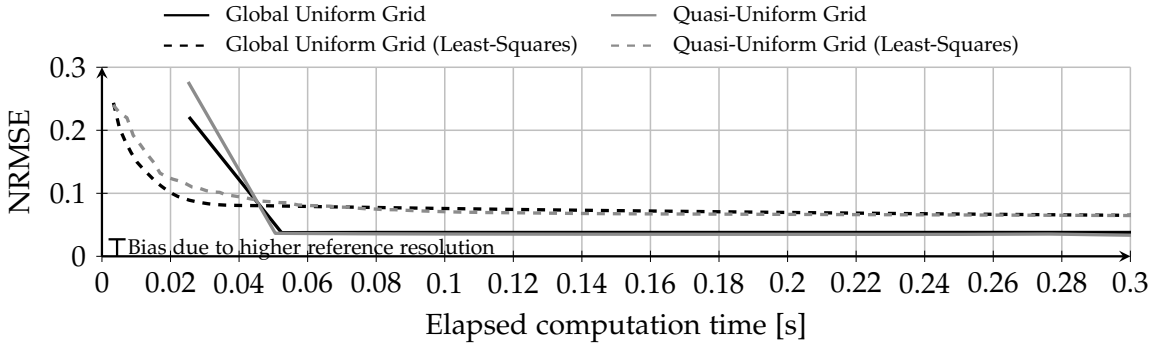


Figure C.2.: Convergence of the open-loop solution to the coupled Van der Pol oscillator without limited computation time.

C.4.3. Examples

Detailed examples and evaluations are provided in [Rös+14a; Rös+16a]. This section provides a brief summary of the results.
According to Remark C.4.2, the quadratic penalties cannot enforce constraints well. Even if the weight adaptation provides a remedy, the overall procedure is rather limited to small- to mid-scale problems. To this end, the model of the Van der Pol oscillator (see Section 3.3.1) is extended to increase the complexity for the following example.
The so-called coupled Van der Pol oscillator is composed of two individual oscillators and a coupling term:

$$\ddot{y}_1(t) - \big(1 - y_1(t)^2\big)\dot{y}_1(t) + y_1(t) - \big(y_2(t) - y_1(t)\big) = u(t), \tag{C.4.18}$$
$$\ddot{y}_2(t) - \big(1 - y_2(t)^2\big)\dot{y}_2(t) + y_2(t) - \big(y_1(t) - y_2(t)\big) = 0 \tag{C.4.19}$$

with $y_1(t), y_2(t) \in \mathbb{R}$ and $u(t) \in \mathcal{U} = \mathbb{R}$.
The state space model with state $x(t) := \big(x_1(t), x_2(t), x_3(t), x_4(t)\big)^\intercal \in \mathcal{X} = \mathbb{R}^4$ is defined by:

$$\dot{x}(t) = f\big(x(t), u(t)\big) = \begin{pmatrix} x_2(t) \\ \big(1 - x_1(t)^2\big)x_2(t) - 2x_1(t) + x_3(t) \\ x_4(t) \\ \big(1 - x_3(t)^2\big)x_4(t) - 2x_3(t) + x_1(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} u(t),$$
$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} x(t). \tag{C.4.20}$$

The control task is to guide the system from the origin to $x_\mathrm{f} = (1, 0, 0.5, 0)^\intercal$ in minimum time while adhering to control constraints $\mathbb{U} := \{u \in \mathcal{U} \mid |u| \leq 1.6\}$. Collocation with Crank-Nicolson differences (4.1.7) transforms the optimal control problem into a nonlinear program. In this example, both the exact formulation according to Section 6 with IPOPT (explicit Hessian) and the least-squares approximation with Levenberg-Marquardt are taken into account.

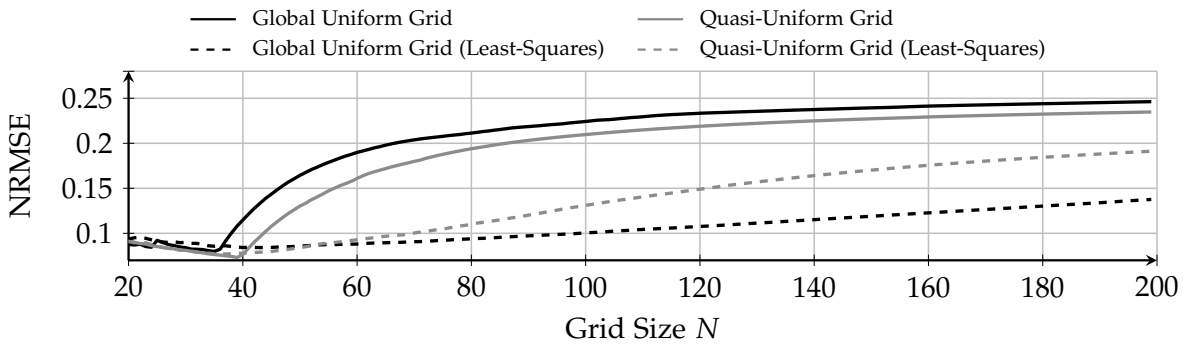


Figure C.3.: Open-loop control of the coupled Van der Pol olscillator with fixed computation time to 25 ms and increasing grid size N.

For real-time applications it is often interesting to analyze the performance under limited computational resources. To this end, Figure C.3 provides the normalized root-mean-square error (NRMSE) of the state and control trajectories with respect to the known optimal solution ($N = 200$, solved up to convergence) for varying grid sizes N. The computation time is bounded to 25 ms. Normalization for the NRMSE is performed based on the range of data.
The resulting NRMSE for the different approaches with respect to the reference state and control sequences is depicted in Figure C.3. Two different phases are apparent. The error decreases in the first phase ($N < 50$) with respect to the known reference and the solvers converge properly.
All approaches converge within the first 25 ms. The least-squares solutions perform similar as the nonlinear program solutions. In the second phase ($N > 40$), the solvers are no longer able to converge within the first 25 ms. As the computational burden increases the convergence deteriorates for increasing grid sizes. The least-squares realizations generally exhibit lower errors, whereas the global grid performs best in this particular example.
Figure C.2 shows the NRMSE for $N = 70$ with respect to the elapsed computation time. The restriction to 25 ms is inactive. The results indicate that the least-squares realizations converge faster to a suboptimal solution. However, beyond 50 ms, the actual nonlinear programs outperform the least-squares solutions that converge more slowly to the optimal solution. Notice, the NRMSE for the converged solutions is still biased, since the reference solution has a higher resolution ($N = 200$).
Figure C.4 shows an example of the closed-loop control achieved by the quasi-uniform grid least-squares approximation. The plant model is simulated by a 5^{th}-order Runge-Kutta method (see Section A.3) and hence the controller copes with a slight model mismatch. Figure C.4 also depicts the reference state and control trajectories for the open-loop solution with $N = 200$.

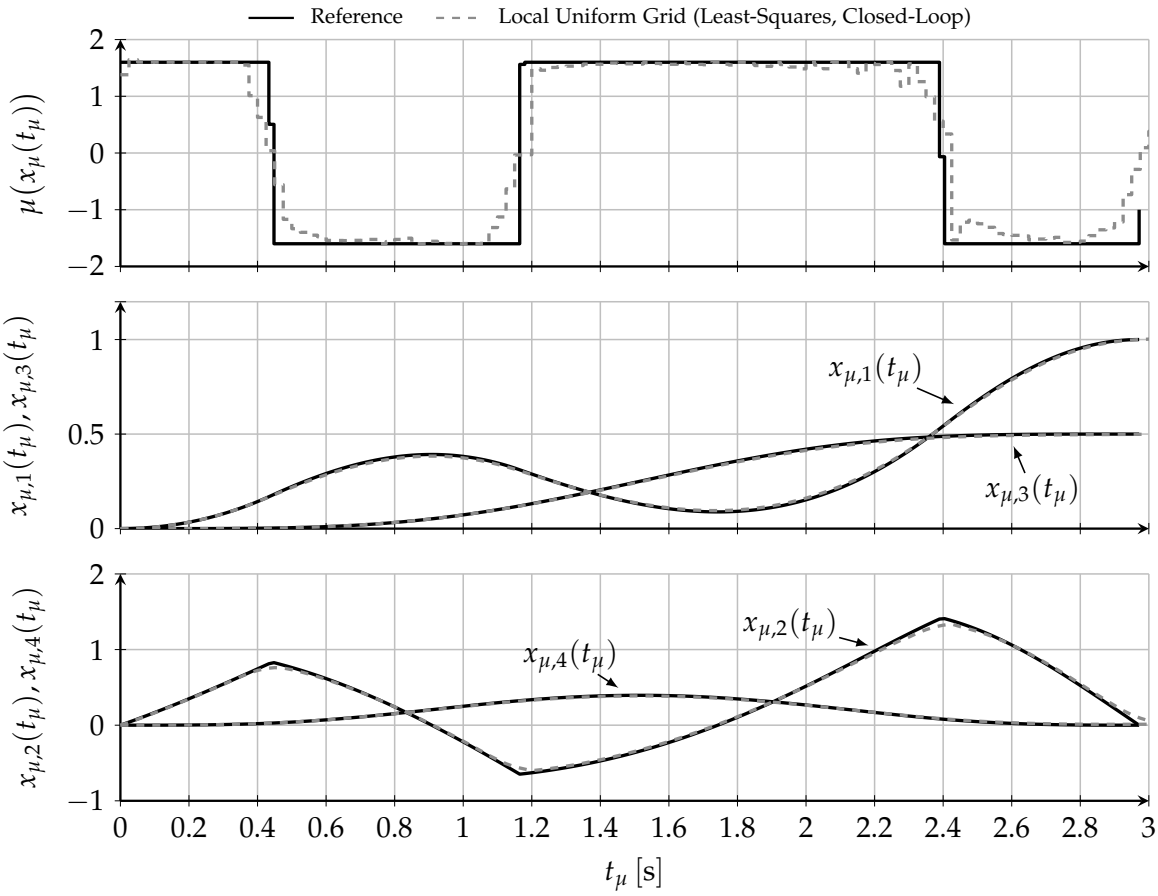


Figure C.4.: Closed-loop example of the coupled Van der Pol oscillator and the quasi-uniform grid approach (least-squares approximation). The open-loop reference $N = 200$ is shown as well.

D

Hypergraph for Nonlinear Model Predictive Control

This chapter introduces the hypergraph for standard model predictive control formulations. The general hypergraph is introduced in Chapter 7 and particularly for general nonlinear programs in Section 7.1. In the context of continuous-time systems, single shooting, multiple shooting and collocation apply to the general optimal control problem (3.2.1). However, formulating the hypergraph for simultaneous and sequential methods in terms of discrete-time systems is analogue. This chapter summarizes the main results presented in [Rös+18a].

D.1. Structural Sparsity Exploitation with Hypergraphs

Since multiple shooting and collocation are already described in Chapter 4 for variable discretization grids, this section briefly lists the nonlinear programs for fixed discretization grids including single shooting.

D.1.1. Single Shooting

Let $t_0 < t_1 < \ldots < t_k < \ldots < t_N = t_\mathrm{f}$ define a grid with N partitions. The control input trajectory is represented by piecewise constant controls, in particular $u(t) := u_k = \text{constant for } t \in [t_k, t_{k+1})$. The state trajectory emerges as the solution of the initial value problem according to (3.1.2) with initial state $x_\mathrm{s} \in \mathcal{X}$: $x_u(t) := \varphi\big(t - t_0, x_\mathrm{s}, u(t)\big)$. Applying single shooting to the continuous-time optimal control problem (3.2.1) results in:

$$\min_{u_0, u_1, \ldots, u_{N-1}} \Big[V_\mathrm{f}\big(x_u(t_N)\big) + \int_{t_0}^{t_\mathrm{f}} \ell\big(x_u(t), u(t)\big)\, \mathrm{d}t \Big] \tag{D.1.1}$$

subject to

$$x_u(t_0) = x_\mathrm{s}, \quad x_u(t_k) \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad x_u(t_N) \in \mathbb{X}_\mathrm{f}, \quad k = 0, 1, \ldots, N-1.$$

Hereby, $x_u(t_k)$ denotes the state at grid time instance t_k obtained from the initial value problem. The remaining parameters are introduced in Chapter 3. For the sake

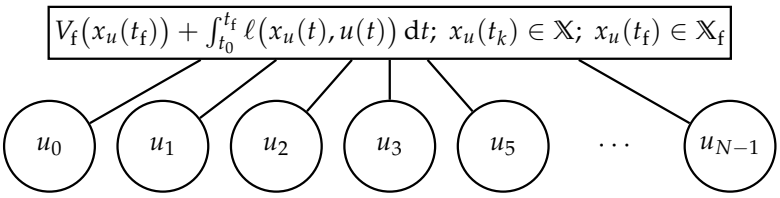


Figure D.1.: Single shooting hypergraph example

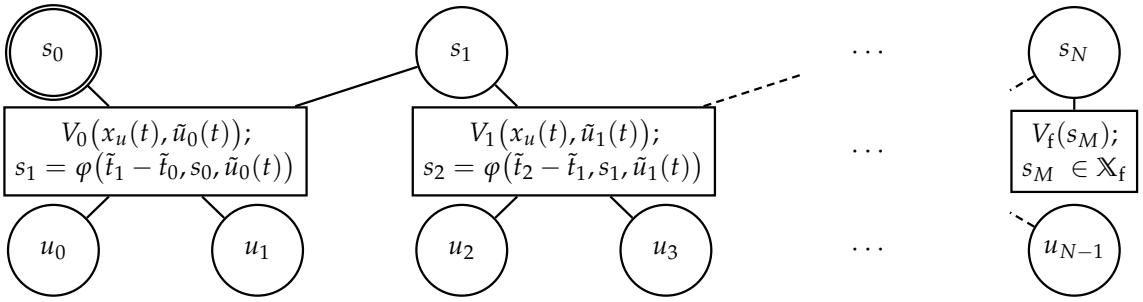


Figure D.2.: Multiple shooting hypergraph with two controls per shooting interval. A double circle indicates a fixed vertex.

of readability, the cost terms and the initial value problem employ the notation $u(t)$ which in turn relates to the sequence of optimization parameters u_k in interval $[t_0, t_\mathrm{f}]$. Figure D.1 shows an example hypergraph for single shooting. Since single shooting (D.1.1) contains no isolated summands in its cost function, the corresponding hypergraph is composed of a single global edge. In this example, the control constraints $u_k \in \mathbb{U}$ are considered as box constraints and hence are included in the vertex description (see Section 7.1). Obviously, the single shooting hypergraph reveals no sparse structure.

D.1.2. Multiple Shooting

The multiple shooting derivation follows Section 4.1.2 by further introducing a grid for shooting nodes $s_i \in \mathcal{X}$, in particular $\tilde{t}_0 < \tilde{t}_1 < \ldots < \tilde{t}_i < \ldots < \tilde{t}_M = t_\mathrm{f}$. For details and definitions refer to Section 4.1.2.
The nonlinear program with general cost terms is defined as follows:

$$\min_{\substack{u_0,u_1,\ldots,u_{N-1},\\ s_0,s_1,\ldots,s_M}} \Big[V_\mathrm{f}(s_N) + \sum_{i=0}^{M-1} \underbrace{\int_{\tilde{t}_i}^{\tilde{t}_{i+1}} \ell\big(x_u(t), \tilde{u}_i(t)\big)\,\mathrm{d}t}_{V_i\left(x_u(t),\tilde{u}_i(t)\right)} \Big] \tag{D.1.2}$$

subject to

$$\begin{aligned}
&s_0 = x_\mathrm{s}, \quad s_M \in \mathbb{X}_\mathrm{f}\\
&s_i \in \mathbb{X}, \quad u_k \in \mathbb{U} \text{ for all } k = 0, 1, \ldots, N-1,\\
&s_{i+1} = \varphi\big(\tilde{m}_i \Delta t, s_i, \tilde{u}_i(t)\big),\\
&\varphi\big(\varpi \Delta t, s_i, \tilde{u}_i(t)\big) \in \mathbb{X} \text{ for all } \varpi = 1, 2, \ldots, \tilde{m}_i - 1\\
&i = 0, 1, \ldots, M-1.
\end{aligned}$$

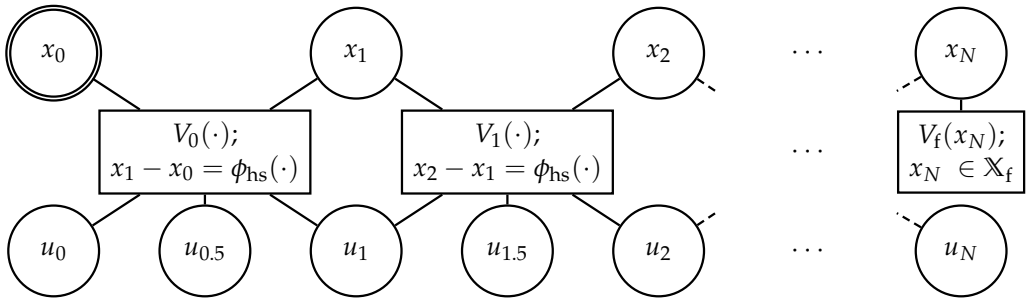


Figure D.3.: Example Hypergraph of compressed form Hermite-Simpson collocation with controls at midpoints.

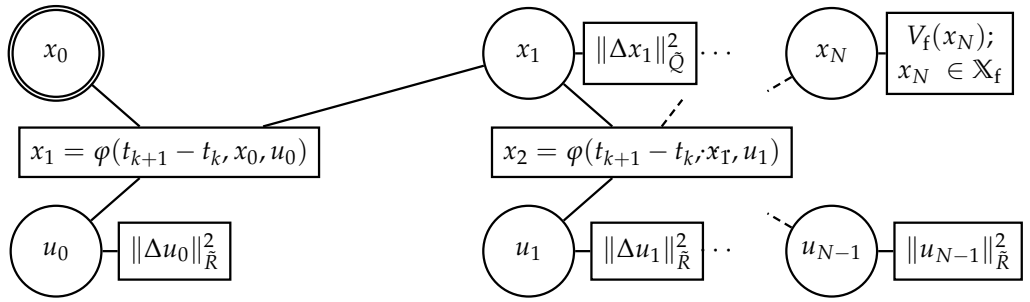


Figure D.4.: Full discretization hypergraph with quadratic cost.

Note, cost term $V_i(\cdot)$ only depends on s_i and $\tilde{u}_i(t)$ whereas the latter includes the subset of u_k that fall belong to the shooting interval $[t_i, t_{i+1}]$. This already indicates a sparser structure and is later exploited by proper edges in the hypergraph. Figure D.2 shows an example hypergraph with two controls per shooting interval ($\tilde{m}_0 = \tilde{m}_1 = 2$). In contrast to single shooting, the hypergraph exhibits several edges, each corresponding to a particular shooting interval. For the sake of simplicity, state constraints are only evaluated at shooting nodes s_i. Box constraints for shooting nodes and controls are included in the vertices.

D.1.3. Hermite-Simpson Collocation

Direct collocation discretizes both the state and the control trajectory according to grid $t_0 < t_1 < \ldots < t_k < \ldots < t_N = t_{\text{f}}$. This thesis focuses on Hermite-Simpson collocation as introduced in Section 4.1.3.

This section describes the nonlinear program and hypergraph for the compressed form with intermediate controls $u_{k+0.5}$. However, hypergraph formulations for the other types are straightforward.

In addition to the time-optimal control problem in Section 4.1.3, the integral running cost is approximated by Simpson quadrature:

$$V_k(x_k, x_{k+1}, u_k, u_{k+0.5}, u_{k+1}) = \frac{t_{k+1} - t_k}{6}\big(\ell(x_k, u_k) + 4\ell(x_{k+0.5}, u_{k+0.5}) + \ell(x_{k+1}, u_{k+1})\big). \tag{D.1.3}$$

Due to the compressed form, $x_{k+0.5}$ is substituted by (4.1.15) and is therefore omitted from the argument list. Note, a practical implementation computes the interpolant

$x_{k+0.5}$ once for evaluating the system dynamics constraint, the cost term and intermediate state constraints (even in compressed form). The resulting nonlinear problem is defined by:

$$\min_{\substack{u_0,u_{0.5},u_1,\ldots,u_N,\\ x_0,x_1,\ldots,x_N}} \left[V_\mathrm{f}(x_N) + \sum_{k=0}^{N-1} V_k(x_k, x_{k+1}, u_k, u_{k+0.5}, u_{k+1})\right] \tag{D.1.4}$$

subject to

$$\begin{aligned}
&x_0 = x_\mu(t_\mu), \quad x_N \in \mathbb{X}_\mathrm{f}, \quad u_N \in \mathbb{U},\\
&x_k \in \mathbb{X}, \quad x_{k+0.5} \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad u_{k+0.5} \in \mathbb{U}\\
&x_{k+1} - x_k = \phi_\mathrm{hs}(x_k, x_{k+1}, u_k, u_{0.5}, u_{k+1}, t_{k+1} - t_k),\\
&k = 0, 1, \ldots, N-1.
\end{aligned}$$

Figure D.3 depicts an example hypergraph for compressed form Hermite-Simpson collocation with intermediate controls. Hereby, each edge depends on two consecutive states and three controls.

D.1.4. Full Discretization

Collocation via finite differences and multiple shooting with $M = N$ ($i = k$, $s_i = x_k$) and constant controls are sometimes referred to as full discretization approach. In case the grid resolution is high, simple one-step integration schemes or finite differences are often sufficient. Also in full discretization, the integral of the running cost is often approximated by the Riemann sum, in particular $V_k(\cdot) \approx \ell(\cdot)(t_{k+1} - t_k)$.
An example hypergraph for full discretization with quadratic form cost functions is shown in Figure D.4. With references $x_\mathrm{f} \in \mathbb{X}_\mathrm{f}$ and $u_\mathrm{f} \in \mathbb{U}$, the running cost is set to

$$V_k(\cdot) = \Delta x_k^\intercal \tilde{Q} \Delta x_k + \Delta u_k^\intercal \tilde{R} \Delta u_k \tag{D.1.5}$$

with $\Delta x_k = x_k - x_\mathrm{f}$, $\Delta u_k = u_k - u_\mathrm{f}$, $\tilde{Q} = (t_{k+1} - t_k)Q$, $\tilde{R} = (t_{k+1} - t_k)R$ and positive definite weighting matrices $Q \in \mathbb{R}^{p \times p}$ as well as $R \in \mathbb{R}^{q \times q}$. Figure D.4 abbreviates the above quadratic state error by $\|\Delta x_k\|_{\tilde{Q}}^2 = \Delta x_k^\intercal \tilde{Q} \Delta x_k$. The same applies to the control effort. Each edge is associated with cost terms that are independent of intermediate solutions to the systems dynamics. The hypergraph reveals the most sparse structure for full discretization. On the other hand, the number of optimization parameters must be large for a sufficient system dynamics approximation accuracy.

D.2. Comparative Analysis and Benchmark Results

This section summarizes the performance results presented in [Rös+18a]. A comparison with the automatic differentiation framework CasADi (v. 3.3.0) [And13] is carried out (see introduction of Chapter 7). Rather than comparing individual gradient and Jacobian computation times, computation times include the entire solution of the nonlinear program up to convergence for identical solver parameters. The analysis takes

into account that CasADi provides exact derivatives, whereas the proposed hypergraph approaches operate with approximate derivatives (see Chapter 7). The optimal control problems are solved with IPOPT (refer to Section E.2.1) and the BFGS Hessian approximation. IPOPT's relative convergence threshold is set to 10^{-4}. CasADi provides two modes, SX and MX respectively. SX is intended for fast computation whereas MX focuses on memory efficiency.
The first benchmark system is the Van der Pol oscillator which is described in Section 3.3.1. The control task consists of controlling the system from the origin to $x_\mathrm{f} = (1,0)^\intercal$ while adhering to $\mathbb{U}$ and minimizing a quadratic cost function with $Q = \operatorname{diag}\big((1.5,0.5)\big)$ and $R = 0.1$. The temporal grid is defined uniformly with $t_{k+1} - t_k = 0.1\,\mathrm{s}$.
The second benchmark problem is to balance the cart-pole system (time arguments are omitted for a better readability):

$$\begin{aligned}\ddot{x}_\mathrm{c} &= \frac{l m_p \sin(\phi_\mathrm{c})\dot{\phi}_\mathrm{c}^2 + u + m_p g \cos(\phi_\mathrm{c})\sin(\phi_\mathrm{c})}{m_c + m_p\big(1-\cos^2(\phi_\mathrm{c})\big)},\\ \ddot{\phi}_\mathrm{c} &= -\frac{l m_p \cos(\phi_\mathrm{c})\sin(\phi_\mathrm{c})\dot{\phi}_\mathrm{c}^2 + u\cos(\phi_\mathrm{c}) + (m_c+m_p)g\sin(\phi_\mathrm{c})}{l m_c + l m_p\big(1-\cos^2(\phi_\mathrm{c})\big)}.\end{aligned} \tag{D.2.1}$$

The state vector is set to $x(t) = \big(x_\mathrm{c}, \phi_\mathrm{c}, \dot{x}_\mathrm{c}, \dot{\phi}_\mathrm{c}\big)^\intercal$ and the task is to upswing the pendulum from the lower stable steady state $x_\mathrm{s} = (0,0,0,0)^\intercal$ to the upper steady state $x_\mathrm{f} = (-0.5,\pi,0,0)^\intercal$ and to stabilize it there. Weighting matrices of the quadratic form cost function are set to $Q = \operatorname{diag}\big((2,10,0.25,0.5)\big)$ and $R = 0.1$. The temporal grid resolution is $t_{k+1} - t_k t = 0.01\,\mathrm{s}$. Dynamic parameters in (D.2.1) are taken from [Kel17]. For both benchmarks, the terminal state cost is set to $V_\mathrm{f}(x_N) = \|\Delta x_N\|_Q^2$ and the terminal constraint is omitted, in particular $\mathbb{X}_\mathrm{f} = \mathbb{X}$.
For the analysis, the computation time required to solve the benchmark problems is partitioned into the preparation time $\bar{T}_\mathrm{p}$ and the solving time $\bar{T}_\mathrm{s}$. Both CasADi and the hypergraph approach prepare the optimization problem. The hypergraph approach sets up the hypergraph structure, establishes edge connectivity and collects strategy-related information. For CasADi, the preparation time includes the graph generation, the solving time involves the numerical evaluation, in particular the invocation of IPOPT. Table D.1 shows the mean computation times over 100 repetitions for both benchmarks with increasing horizon lengths N. Obviously, methods which exploit sparsity such as the hypergraph (except for single shooting) and automatic differentiation clearly outperform the dense computation. Note, since IPOPT is a sparse solver, dense matrix operations automatically imply an additional overhead. For single shooting, the hypergraph offers no advantages but automatic differentiation already exploits the structure on the function graph level. CasADi in SX mode is in general slightly faster in terms of the actual solving time, but requires a substantial and non-negligible preparation time. For the full discretization formulation, the hypergraph performs even better in these benchmarks. Figure D.5 compares the preparation and solving times with respect to the horizon length N. Whereas computation times are similar for low and mid horizon lengths, the setup time of the hypergraph is two orders of magnitude lower.
Concludingly, the overall benchmark results favor automatic differentiation in case

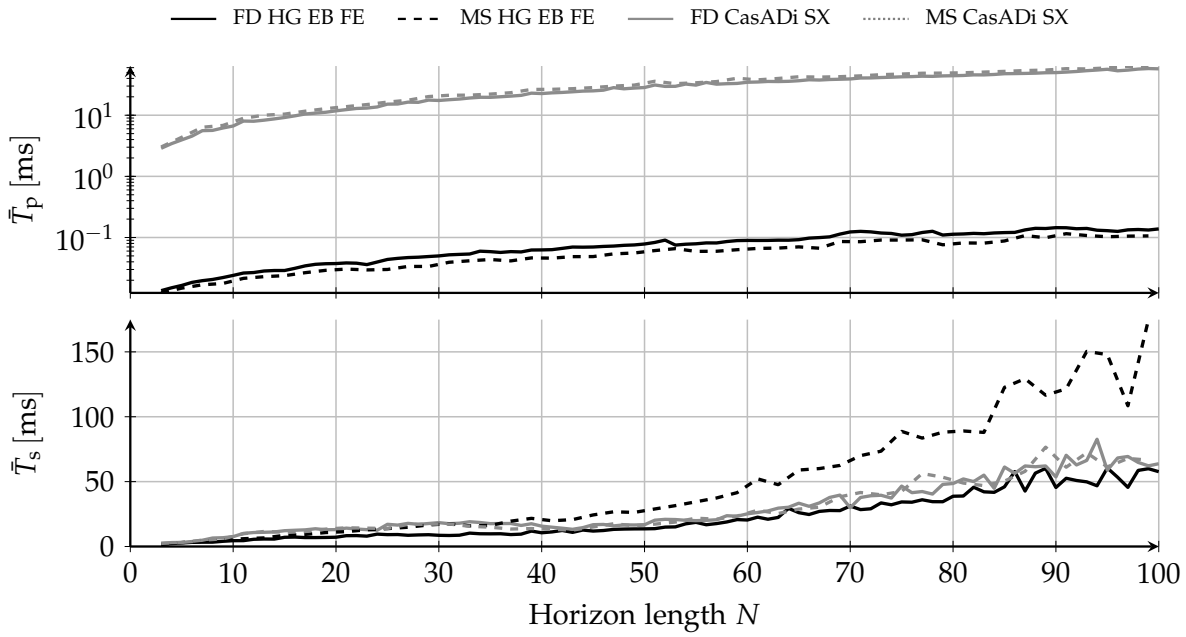


Figure D.5.: Cart-pole preparation times $\bar{T}_p$ and solving times $\bar{T}_s$ for the hypergraph (HG) formulation with edge-based (EB) iterations according to Algorithm 7.1 and CasADi with SX mode. Direct transcriptions is realized with both full discretization (FD) and multiple shooting (MS) with two controls per interval. The integration scheme is forward Euler (FE).

of MPC problems with static structure. The overhead of setting up the structure is more than compensated by the efficiency of solving the problem of identical structure repeatedly in closed-loop control. On the other hand, the hypergraph with its negligible preparation time is preferred for varying problem structures. For example, the grid adaptation schemes for time-optimal control as proposed in this thesis change the dimensions in every sampling interval. Also other MPC applications in which grid sizes or constraint sets change frequently benefit from the hypergraph formulation. For example, robot motion planning in dynamic environments involves changing costs and constraint sets during runtime (as dynamic obstacles enter the workspace).

Table D.1.: Benchmark results for single shooting (SS), multiple shooting with 2 controls per interval (MS), full-discretization (FD) and Hermite-Simpson collocation (HS); shooting with forward Euler (FE) and 5$^{\text{th}}$-order Runge-Kutta (RK5) integration. The hypergraph (HG) performs with either edge-based (EB) iterations according to Algorithm 7.1 or vertex-based iterations according to Algorithm 7.2.

		Van der Pol Oscillator						Cart-Pole System					
		$N = 5$		$N = 45$		$N = 85$		$N = 5$		$N = 45$		$N = 85$	
		$\bar{T}_\text{p}$ [ms]	$\bar{T}_\text{s}$ [ms]	$\bar{T}_\text{p}$ [ms]	$\bar{T}_\text{s}$ [ms]	$\bar{T}_\text{p}$ [ms]	$\bar{T}_\text{s}$ [ms]	$\bar{T}_\text{p}$ [ms]	$\bar{T}_\text{s}$ [ms]	$\bar{T}_\text{p}$ [ms]	$\bar{T}_\text{s}$ [ms]	$\bar{T}_\text{p}$ [ms]	$\bar{T}_\text{s}$ [ms]
SS FE	Dense	–	2.1	–	42.8	–	283.1	–	1.3	–	27.5	–	397.5
	HG EB	**0.01**	2.1	**0.02**	37.5	**0.03**	250.4	**0.01**	1.3	**0.02**	25.4	**0.03**	362.5
	HG VB	0.01	**2.1**	0.03	38.3	0.04	258.1	0.01	**1.3**	0.02	26.5	0.04	378.8
	CasADi SX	2.47	2.9	10.70	**13.0**	19.36	**34.2**	2.98	3.0	19.76	**6.4**	37.36	**40.7**
	CasADi MX	4.10	3.2	26.94	22.9	50.71	77.0	6.90	3.7	63.90	15.0	121.90	116.4
SS RK5	Dense	–	3.5	–	220.6	–	1561.8	–	2.2	–	140.3	–	2037.3
	HG EB	**0.01**	3.4	**0.02**	221.0	**0.03**	1561.9	**0.01**	2.1	**0.02**	139.0	**0.03**	2015.0
	HG VB	0.01	3.4	0.02	223.1	0.04	1574.5	0.01	**2.1**	0.02	140.3	0.03	2036.1
	CasADi SX	2.24	**3.0**	10.03	**13.0**	17.61	**34.2**	2.96	3.0	19.25	**6.3**	35.87	**39.6**
	CasADi MX	4.33	3.2	26.92	23.1	50.45	77.2	7.16	3.7	64.97	15.0	122.33	116.4
MS FE	Dense	–	2.9	–	127.6	–	615.3	–	2.9	–	278.8	–	2681.3
	HG EB	**0.02**	**2.7**	**0.06**	13.6	**0.10**	29.2	**0.01**	3.0	**0.05**	25.2	**0.08**	123.5
	HG VB	0.02	2.8	0.08	14.1	0.14	29.6	0.02	**2.6**	0.06	25.5	0.10	125.6
	CasADi SX	3.05	3.3	14.34	**11.7**	24.77	**14.2**	4.36	3.1	29.33	**15.1**	51.93	**50.5**
	CasADi MX	8.27	4.2	71.27	30.8	145.24	50.3	18.49	4.9	192.44	60.3	407.96	284.0
MS RK5	Dense	–	5.5	–	650.1	–	2506.5	–	7.5	–	1326.2	–	13 810.8
	HG EB	**0.01**	4.2	**0.05**	53.3	**0.08**	102.3	**0.01**	5.2	**0.05**	106.8	**0.08**	600.4
	HG VB	0.02	4.3	0.06	54.0	0.10	103.3	0.02	5.2	0.06	107.4	0.10	592.0
	CasADi SX	2.78	**3.2**	12.80	**11.2**	22.76	**14.3**	4.25	**3.0**	27.42	**15.0**	50.64	**49.7**
	CasADi MX	8.42	4.2	71.43	31.1	145.43	50.5	18.77	4.9	193.54	60.3	408.30	284.3
FD	Dense	–	2.8	–	106.7	–	489.4	–	3.5	–	402.6	–	5136.1
	HG EB	**0.02**	2.5	**0.09**	**8.6**	**0.14**	**11.6**	**0.02**	2.7	**0.07**	**12.3**	**0.11**	**46.8**
	HG VB	0.02	**2.5**	0.12	8.6	0.17	11.8	0.02	**2.7**	0.09	12.8	0.16	48.4
	CasADi SX	2.91	3.4	12.46	10.0	22.45	15.4	4.06	3.1	25.06	17.0	47.80	60.5
	CasADi MX	6.09	3.8	51.53	18.2	112.31	35.5	11.94	3.9	144.46	40.7	345.94	197.0
HS	Dense	–	5.4	–	482.1	–	2294.9	–	7.8	–	1258.1	–	17 234.4
	HG EB	**0.02**	3.8	**0.10**	25.7	**0.16**	52.9	**0.02**	5.1	**0.08**	49.0	**0.14**	243.8
	HG VB	0.02	**3.7**	0.12	25.4	0.19	53.3	0.02	**4.9**	0.10	48.8	0.19	251.8
	CasADi SX	4.87	4.7	34.90	**17.8**	62.74	**33.5**	9.54	5.9	83.76	**29.1**	157.89	**138.7**
	CasADi MX	16.71	7.1	186.23	55.7	419.43	130.7	44.28	11.5	626.33	136.2	1526.34	797.3

E

Implementation Details and Algorithms

This chapter provides information on the developed and utilized software tools that has been used to create and evaluate the results in this thesis.

E.1. Software Framework

The theoretical results and examples in this thesis are accompanied by a C++ software framework for general control systems. A central focus lies on the efficient implementation of optimization based approaches like MPC and optimal control. The framework relies on sparse algebra via *Eigen* [GJ+10] and provides many specialized interfaces to other software packages to facilitate efficient memory sharing. Core functionalities and a collection of numeric subroutines like solving Riccati, Lyapunov and Sylvester equations in either continuous-time or discrete-time as well as numerical integration and differentiation schemes are provided with own library-free implementations to facilitate its usage on several platforms. The package also includes the numerical simulation of dynamic systems and supports multi-threaded controller and plant simulations.

Core features

- Modular class hierarchy (system, plant, controller, observer, solver, task).
- Fully generic and template based hypergraph implementation for general nonlinear programs.
- Default implementations for common MPC and optimal control tasks.
- Dynamically configured graphical user interface (GUI).
- Network-based client-server communication (visualization and controllers can run across multiple machines).
- Language and platform independent parameter and data serialization using Google's *Protobuf*.
- Export of measured data (Matlab, YAML).
- Unit tests for core functionalities.
- Native support of several operating systems including Windows, Ubuntu, MacOS.

E.2. Optimization Algorithms

E.2.1. Interior-Point Method

IPOPT (**I**nterior **P**oint **Opt**imizer) constitutes an established open-source nonlinear programming solver written in C++. Especially large-scale optimization problems handles IPOPT well and efficiently. IPOPT implements a sparse primal-dual interior-point algorithm with a filter line-search strategy [WB06]. Inequality constraints are converted to cost terms by barrier functions. The primal-dual equations are solved using a damped Newton method which in turn leads to the solution to a linear system. IPOPT provides interfaces to established linear solvers. Well-known for efficient sparse linear solvers is the *Harwell Subroutine Library* (HSL), especially algorithm HSL-MA57 for small and medium sized optimization problems [Com].

Table E.1 lists the default solver settings defined for the simulations, experiments and benchmarks in this thesis. Whether the Hessian is approximated by BFGS (limited memory) or finite differences is stated explicitly in the particular experiment or simulation section. The software framework described in Section E.1 seamlessly integrates with IPOPT. IPOPT requires sparse matrices in triplet format and supports the Hessian to be defined only in terms of the lower triangular part. The hypergraph implementation creates data structures and allocates memory once whenever the structure of the nonlinear program changes. During solving stage, IPOPT queries the values for the cost gradient, constraint Jacobians and the Hessian of the Lagrangian (if explicit Hessian computation is active). These values are computed directly and stored in IPOPT's memory format to avoid additional copies and conversions.

Table E.1.: Solver settings for IPOPT version 3.0.0. All other parameters are set to default.

Parameter	IPOPT Name	Value
Linear Solver	*linear_solver*	HSL-MA57
Maximum Iterations	*max_iter*	1000 (if not stated otherwise)
Relative Tolerance	*tol*	$1 \cdot 10^{-4}$
NLP Scaling	*nlp_scaling_method*	gradient-based
Mehrotra Algorithm	*mehrotra_algorithm*	No
(Barrier) Mu Strategy	*mu_strategy*	monotone
Hessian Approximation	*hessian_approximation*	Exact / Limited Memory
Warm-Start Init Point	*warm_start_init_point*	Yes

E.2.2. Sequential Quadratic Programming

This section briefly describes the sequential quadratic programming (SQP) algorithm conducted for simulations and benchmarks in this thesis.

The nominal nonlinear program is defined as:

$$\begin{aligned} \min_{z} \quad & J(z) \\ \text{subject to} \quad & \\ & h(z) = 0, \\ & g(z) \leq 0 \end{aligned} \tag{E.2.1}$$

with $z \in \mathbb{R}^{n_z}$, $J : \mathbb{R}^{n_z} \mapsto \mathbb{R}$, $g : \mathbb{R}^{n_z} \mapsto \mathbb{R}^R$, and $h : \mathbb{R}^{n_z} \mapsto \mathbb{R}^S$ as introduced in Section A.4. The key for any SQP algorithm is to apply the Taylor series expansion to obtain a quadratic subproblem which is repeated iteratively. Superscript $\{\kappa\}$ emphasizes the relation to iteration $\kappa \in \mathbb{N}_0$:

$$\begin{aligned} \min_{\Delta z} \quad & \frac{1}{2} \Delta z^\intercal H^{\{\kappa\}} \Delta z + \nabla_z J(z^{\{\kappa\}})^\intercal \Delta z + J(z^{\{\kappa\}}) \\ \text{subject to} \quad & \\ & \mathrm{D}_z h(z^{\{\kappa\}}) \Delta z + h(z^{\{\kappa\}}) = 0, \\ & \mathrm{D}_z g(z^{\{\kappa\}}) \Delta z + g(z^{\{\kappa\}}) \leq 0. \end{aligned} \tag{E.2.2}$$

Hereby, $\nabla_z J(z^{\{\kappa\}})$ denotes the gradient of the cost function. $\mathrm{D}_z h(z^{\{\kappa\}})$ and $\mathrm{D}_z g(z^{\{\kappa\}})$ are the constraint Jacobians. Optimization parameter $\Delta z \in \mathbb{R}^{n_z}$ defines the increment at the current point $z^{\{\kappa\}}$ with $\kappa \in \mathbb{N}_0$ such that $z^{\{\kappa+1\}} = z^{\{\kappa\}} + \Delta z^*$ defines the parameter update. SQP algorithms repeatedly solve (E.2.2) to iteratively refine the parameter z up to convergence. Note, $H^{\{\kappa\}}$ is often set to the Hessian of the Lagrangian with multipliers $\lambda^{\{\kappa\}} \in \mathbb{R}^L$ and $\mu^{\{\kappa\}} \in \mathbb{R}^S$ respectively, in particular $H^{\{\kappa\}} := \nabla^2_{zz} \mathcal{L}(z^{\{\kappa\}}, \lambda^{\{\kappa\}}, \mu^{\{\kappa\}})$. According to Remark 4.2.2, the Hessian of the Lagrangian is often indefinite for time-optimal control problems and hence the implementation in Section E.1 favors the Hessian of the cost function $H^{\{\kappa\}} := \nabla^2_{zz} J(z^{\{\kappa\}})$.

Remark E.2.1. In case of the global and local uniform grid, the Hessian of the cost function is zero and causes problems with the underlying quadratic programm (QP) solver. As a remedy, the cost $N\Delta t$ for the global uniform grid is changed to $(N\Delta t)^2$. The local uniform grid also squares each time interval Δt_k similar to the quasi-uniform grid but keeps the uniformity constraints. This modification is not suitable for the non-uniform grid, since its successful solution heavily depends on the inertia correction as introduced below. As a side effect, computing the Hessian of the cost function is cheap as it does not involve any system dynamics constraints and could be provided analytically. The sparse Hessian reveals a single non-zero for the global grid, $N - 1$ non-zeros for the local/quasi-uniform grid and no non-zeros for the non-uniform grid.

Solving (E.2.2) repeatedly and taking the full step Δt each time does not account for the actual nonlinearities of the optimization problem and hence a globalization strategy is inevitable in any practical algorithm. Merit functions measure progress to obtain a proper successor point $z^{\{\kappa+1\}}$ in terms of both cost values and constraint satisfaction. This work utilizes the ℓ_1 exact penalty function [Mor+12]:

$$\Phi(z, \mu_\mathrm{m}) = J(z) + \mu_\mathrm{m} \|h(z)\|_1 + \mu_\mathrm{m} \|\max(0, g(z))\|_1 \tag{E.2.3}$$

Algorithm E.1.: Simplified sequential quadratic programming algorithm without watchdog.

1: Choose parameters $0 < \eta_{\text{sqp}} < 0.5$ and $0 < \tau_{\text{sqp}} < 1$.
2: Initialize $z^{\{0\}}$, $\lambda^{\{0\}}$, $\mu^{\{0\}}$
3: Evaluate $J(z^{\{0\}}), h(z^{\{0\}}), g(z^{\{0\}}), \nabla_z J(z^{\{0\}}), H^{\{0\}}, \mathrm{D}_z h(z^{\{0\}}), \mathrm{D}_z g(z^{\{0\}})$
4: **for** $\kappa = 0, 1, \ldots, I_{\text{SQP}}$ **or** until convergence **do**
5: Levenberg modification
6: $\{\Delta z^*, \lambda^*_{\text{qp}}, \mu^*_{\text{qp}}\} \leftarrow$ Solve (E.2.2) ▷ Also retrieve multipliers $\lambda^*_{\text{qp}}, \mu^*_{\text{qp}}$
7: Inertia control
8: $\Delta\lambda \leftarrow \lambda^*_{\text{qp}} - \lambda^{\{\kappa\}}$
9: $\Delta\mu \leftarrow \mu^*_{\text{qp}} - \mu^{\{\kappa\}}$
10: Choose μ_{m} properly
11: $\alpha \leftarrow 1$
12: **while** $\Phi(z^{\{\kappa\}} + \alpha\Delta z^*, \mu_{\text{m}}) > \Phi(z^{\{\kappa\}}, \mu_{\text{m}}) + \alpha\eta_{\text{sqp}} D\big(\Phi(z^{\{\kappa\}}, \mu_{\text{m}}), \Delta z^*\big)$ **do**
13: $\alpha \leftarrow \alpha\tau_{\text{sqp}}$
14: $z^{\{\kappa+1\}} \leftarrow z^{\{\kappa\}} + \alpha\Delta z^*$
15: $\lambda^{\{\kappa+1\}} \leftarrow \lambda^{\{\kappa\}} + \alpha\Delta\lambda$
16: $\mu^{\{\kappa+1\}} \leftarrow \mu^{\{\kappa\}} + \alpha\Delta\mu$
17: Evaluate $J(z^{\{\kappa+1\}}), h(z^{\{\kappa+1\}}), g(z^{\{\kappa+1\}})$
18: Evaluate $\nabla_z J(z^{\{\kappa+1\}}), H^{\{\kappa+1\}}, \mathrm{D}_z h(z^{\{\kappa+1\}}), \mathrm{D}_z g(z^{\{\kappa+1\}})$

with penalty parameter $\mu_{\text{m}} \in \mathbb{R}^+$. The advantage of exact penalty functions like (E.2.3) is that constraint satisfaction is achieved for a finite penalty parameter μ_{m}. A suitable update scheme for μ_{m} is provided in [Mor+12] or in [NW06] for the equality-constrained-only version (however, the scheme is similar).
A straightforward approach to choose a sufficient step $\alpha\Delta z$ with $0 < \alpha \leq 1$ is to decrease the merit function by backtracking line-search based on the Armijo condition [NW06]. The Armijo condition requires the directional derivative of $\Phi(z, \mu_{\text{m}})$ with respect to Δz [Mor+12]:

$$D\big(\Phi(z^{\{\kappa\}}, \mu_{\text{m}}), \Delta z\big) = \nabla_z J(z^{\{\kappa\}})^\intercal \Delta z - \mu_{\text{m}} \Big(\|h(z^{\{\kappa\}})\|_1 + \|\max\big(0, g(z^{\{\kappa\}})\big)\|_1 \Big). \quad \text{(E.2.4)}$$

Algorithm E.1 shows a simplified version of the implemented SQP algorithm and is based on the line-search SQP algorithm from [NW06] with extensions from [Bet10]. Initialization of primal parameters $z^{\{0\}}$ in line 2 for MPC tasks proceeds according to Chapter 5. The dual parameters (multipliers) are initialized either by vectors of zeros or by solving an auxiliary quadratic problem (E.2.2) with $H^{\{0\}}$ as identity matrix [Bet10]. The Levenberg modification in line 5 adapts the diagonal values of the Hessian for $\kappa > 0$ in order to try to maintain positive definiteness. Details on this strategy are provided in [Bet10].
Line 6 invokes the quadratic program solver for (E.2.2) and retrieves the optimal increment as well as the multipliers of the quadratic program. As quadratic program solver serves the open-source software package OSQP (Operator Splitting Quadratic Program) [Ste+17]. The solver with sparse linear algebra routines outperforms most of the commercial and academic quadratic program solvers [Ste+17]. Table E.2 provides

Table E.2.: Solver settings for OSQP version 0.4.1. All other parameters are set to default.

Parameter	OSQP Name	Value
Linear Solver	*linsys_solver*	Internal QDLDL
Maximum Iterations	*max_iter*	200
Absolute Tolerance	*eps_abs*	$1 \cdot 10^{-5}$
Relative Tolerance	*eps_rel*	$1 \cdot 10^{-5}$
Primal Infeas. Tol.	*eps_prim_inf*	$1 \cdot 10^{-4}$
Dual Infeas. Tol	*eps_dual_inf*	$1 \cdot 10^{-4}$
Alpha	*alpha*	1.6
Adaptive Rho	*adaptive_rho*	False
Warm-start	*warm_start*	False

Table E.3.: SQP solver settings.

Parameter	Value
QP Solver	OSQP
Maximum Iterations I_{SQP}	200
Relative Tolerance	$1 \cdot 10^{-5}$
η_{sqp}	$1 \cdot 10^{-5}$
τ_{sqp}	0.9

the settings specified for simulations and benchmarks in this thesis.
Whenever the inertia of the underlying KKT system is incorrect, respectively, whenever the underlying linear solver in OSQP fails, the inertia control in line 7 of Algorithm E.1 tries to fix the Hessian $H^{\{\kappa\}}$ more aggressively than the Levenberg modification. For details on the inertia control scheme refer to [Bet10].
As mentioned before, a suitable update scheme for the penalty parameter in line 10 is provided in [Mor+12] or in [NW06]. Line 12 starts the backtracking line-search scheme. The actual implementation in this thesis further rests upon a watchdog strategy to deal with the Maratos effect as described in [NW06]. Furthermore, the hypergraph seamlessly integrates into the algorithm and matches the same sparsity data structures. Also derivative information is updated only when necessary.
The algorithm terminates if the maximum number of iterations is exceeded or if the convergence thresholds for the KKT conditions are satisfied. Convergence is tested similar to IPOPT [WB06]. In any subsequent invocation of Algorithm E.1 with similar structure, primal and dual variables are warm-started rather than subject to the complete reinitialization. Table E.3 summarizes the SQP solver settings.

E.2.3. Levenberg-Marquardt Algorithm

The implemented algorithm is based on [LA09] and [Küm+11]. Section C.4.2 already provides the mathematical foundation for Algorithm E.2. The implementation utilizes the software package *Eigen* [GJ+10] for sparse linear algebra. Gradient information

and Jacobian matrices are efficiently computed by the hypergraph. The linear system in line 8 is solved by *Eigen*'s direct *LLt* cholesky factorization with fill-in reducing.

Algorithm E.2.: Levenberg-Marquardt optimization routine.

1: Initialize $z^{\{0\}}$
2: Evaluate H and p ▷ See (C.4.14), hypergraph-based
3: $\lambda_{\mathrm{LM}} \leftarrow 10^{-5} \max(\operatorname{diag} H)$
4: $\rho \leftarrow 0$
5: $\nu \leftarrow 2$
6: **for** $\kappa = 1, 2, \ldots, I_{\mathrm{LM}}$ **or** until convergence **do**
7: **repeat**
8: $\Delta z^* \leftarrow$ Solve $(H + \lambda_{\mathrm{LM}} I_{n_z}) \Delta z = -\tilde{p}$ ▷ See (C.4.16)
9: $\hat{\rho} \leftarrow F(z^{\{\kappa\}}) - F(z^{\{\kappa\}} + \Delta z^*)$
10: $\rho \leftarrow \hat{\rho} \left(\Delta z^{*\intercal} \left(\lambda_{\mathrm{LM}} \Delta z^{*\intercal} + \tilde{p} \right) \right)^{-1}$
11: **if** $\rho > 0$ **then**
12: $z^{\{\kappa+1\}} \leftarrow z^{\{\kappa\}} + \Delta z^*$
13: Evaluate H and $\tilde{p}$ ▷ Hypergraph-based
14: $\alpha \leftarrow \max(\min(2/3, 1 - (2\rho - 1)^3), 1/3)$
15: $\lambda_{\mathrm{LM}} \leftarrow \lambda_{\mathrm{LM}} \alpha$
16: $\nu \leftarrow 2$
17: **else**
18: $\lambda_{\mathrm{LM}} \leftarrow \nu \lambda_{\mathrm{LM}}$
19: $\nu \leftarrow 2\nu$
20: **until** $\rho > 0$

F

Supplemental Results

In addition to the benchmarks and examples provided in the main part of this thesis, this chapter provides results for further scenarios and other configurations. Furthermore, results on the model identification for the *ECP Industrial Plant Emulator Model 220* are provided.

F.1. ECP Industrial Plant Emulator Model 220

The *ECP Industrial Plant Emulator Model 220* and its schematic overview are described in Section 3.3.3. The particular model structure is provided by *ECP* itself but the parameters are identified to improve the model accuracy for the testbed at hand. Since transmission belts are almost stiff, individual load plates are substituted by a single one for the model. Individual motor dynamics are neglected. Let $y : \mathbb{R} \mapsto \mathbb{R}$ denote the position of the plate corresponding to the second encoder. Section 3.3.3 defines the motor inputs $u_1(t)$ and $u_2(t)$. A model which takes the moment of inertia, viscous friction, Coulomb friction and both input torques into account is given by:

$$J_{\text{ecp}}\ddot{y}(t) + \tilde{c}_1\dot{y}(t) + \tilde{c}_2 \operatorname{sign}(\dot{y}(t)) = \tilde{k}_1 u_1(t) - \tilde{k}_2 u_2(t) \tag{F.1.1}$$

with moment of inertia J_{ecp}, friction coefficients $\tilde{c}_1$ and $\tilde{c}_2$ as well as motor specific constants $\tilde{k}_1$ and $\tilde{k}_2$. Note, (F.1.1) is normalized by dividing by the strictly positive J_{ecp}. For the identification respectively optimization, the parameter J_{ecp} is redundant and to avoid the inherent null space, model (F.1.1) with physical parameters is replaced by a mathematical model which is given in state space form in (3.3.7). For local optimization purposes, the mathematical model approximates the Coulomb friction by a smooth $\tanh(\cdot)$ function.
Figure F.1 shows the amplitude modulated random binary sequences for both motors which are applied to the plant for model identification. Also, the measured states and the corresponding identification result are depicted. The identification is performed by global nonlinear optimization (pattern search) which minimizes the mean square error between position measurements and simulated model output (3.3.7). Measurements are recorded at 100 Hz which is slightly below the limit of the underlying digital signal processor. The fit is calculated by $100(1 - \text{NRMSE})$ %.

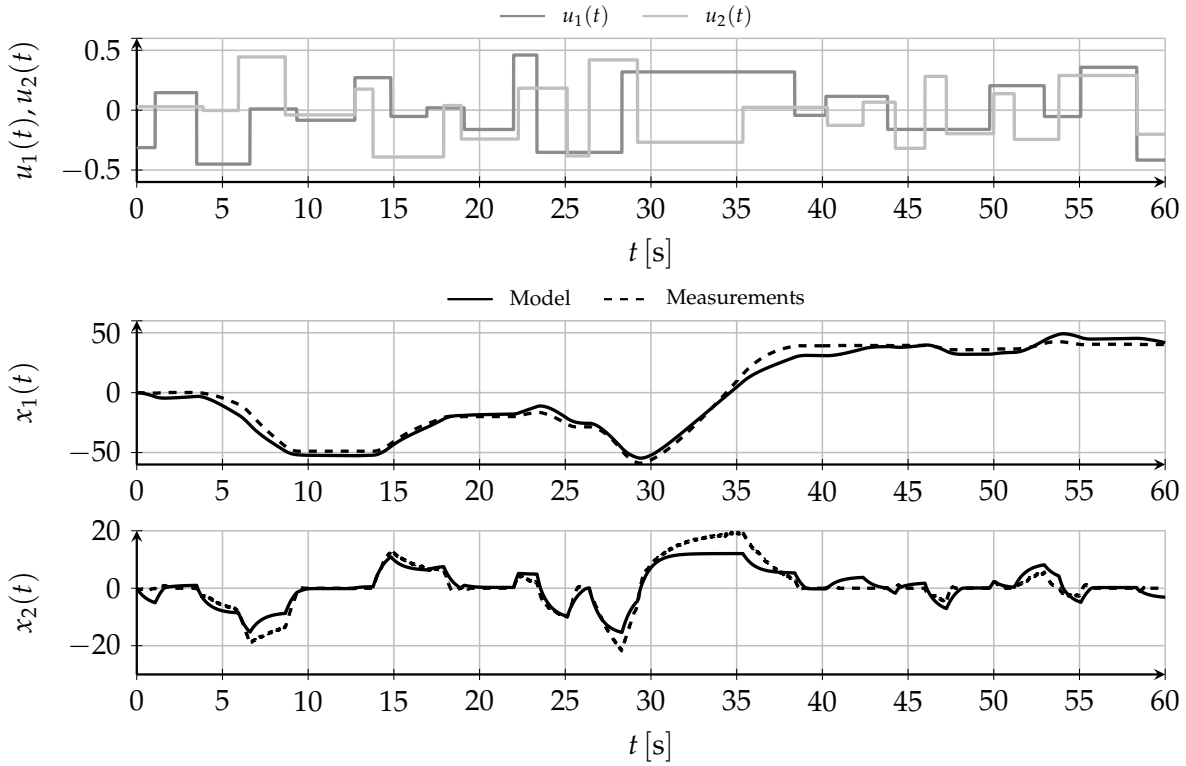


Figure F.1.: Measurements and identification results of the plant emulator ECP Model 220. The fit for the position $x_1(t)$ is 95.49 % and for the velocity $x_2(t)$ 90.39 % respectively.

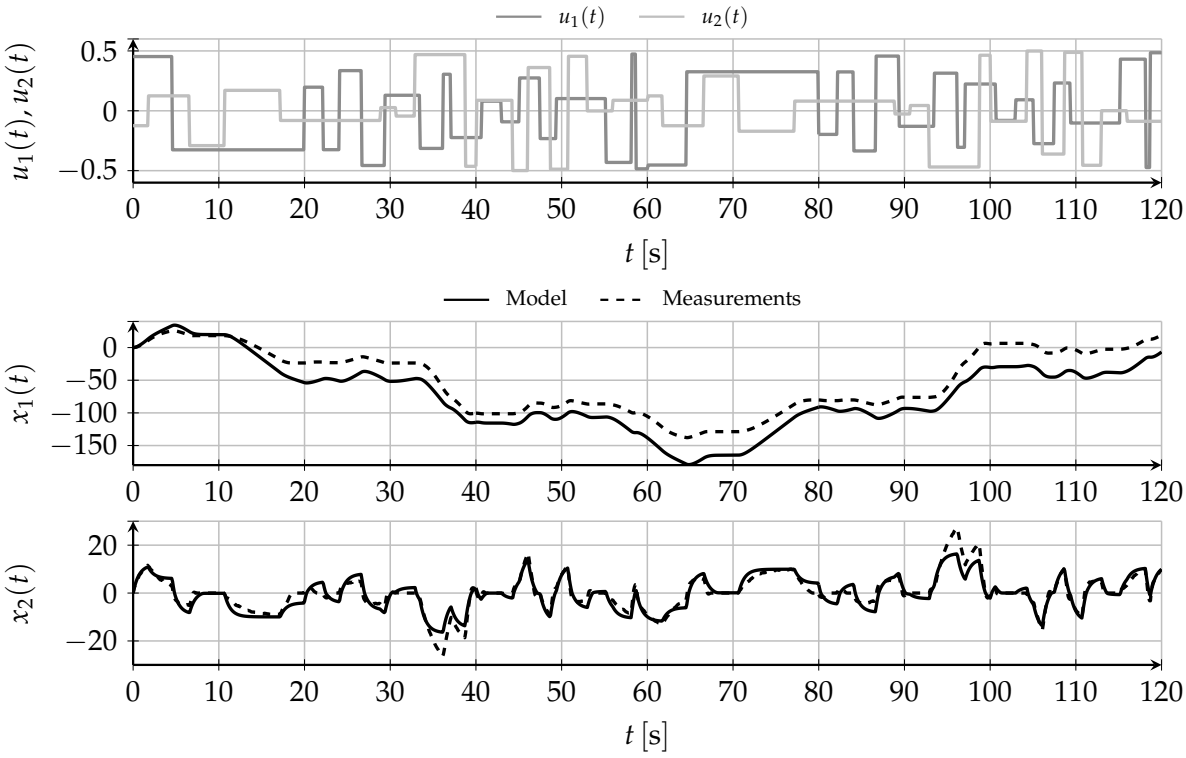
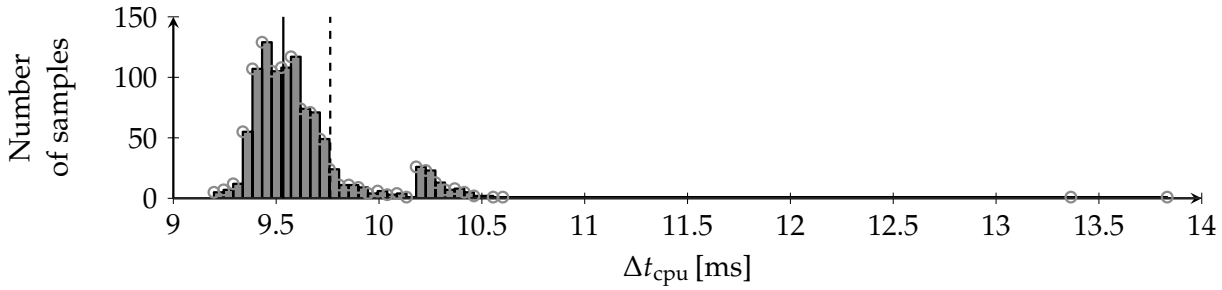
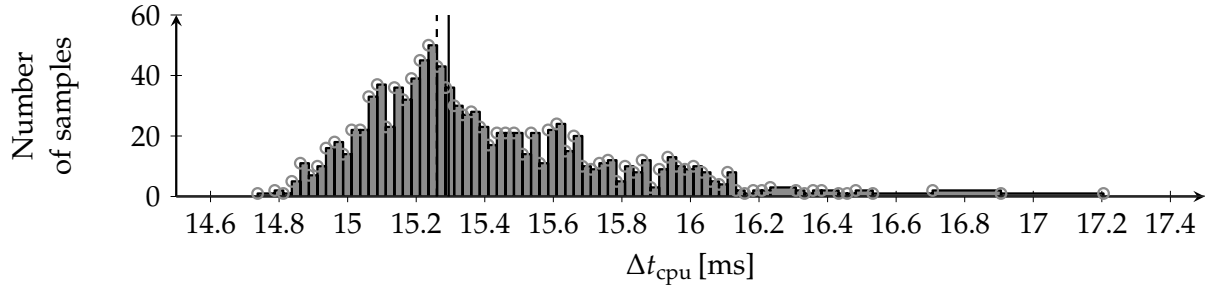


Figure F.2.: Validation of the identified model for the ECP Model 220. The fit for the position $x_1(t)$ is 88.21 % and for the velocity $x_2(t)$ 92 % respectively.



(a) Van der Pol oscillator with multiple shooting (forward Euler) and IPOPT with explicit Hessian.



(b) Rocket system with collocation via Crank-Nicolson differences and IPOPT with explicit Hessian.

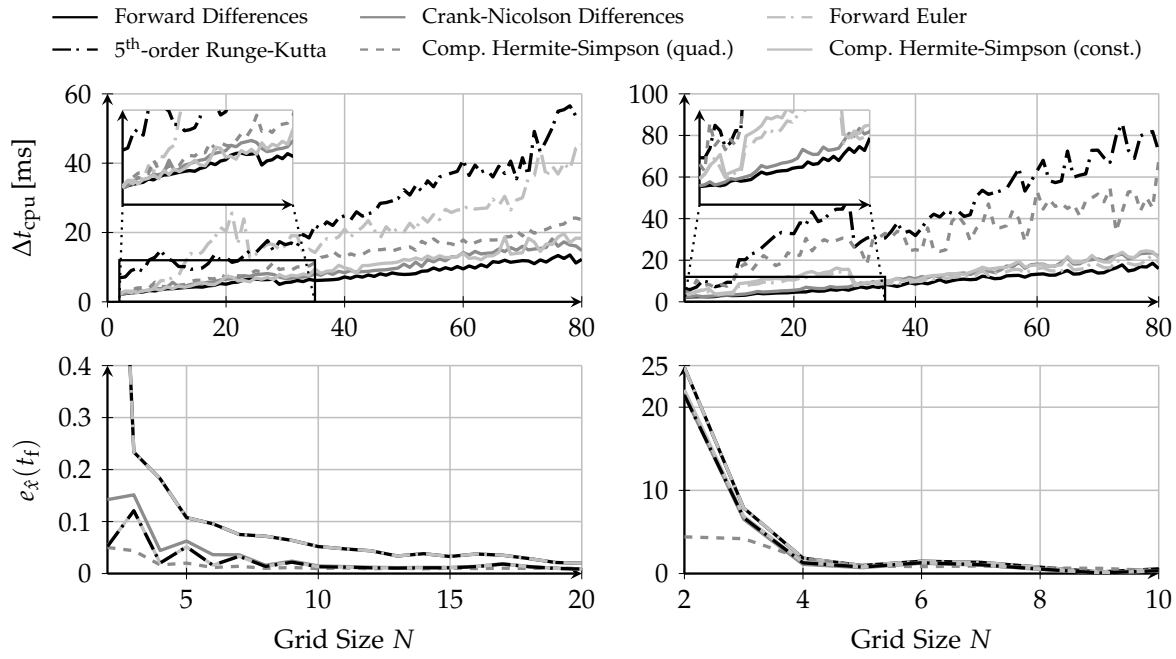
Figure F.3.: Histogram of measured computation times (1000 samples) on two scenarios with grid size $N = 50$. The black vertical line represents the median of the whole data and the dotted line represents the median of the first 20 samples. Circles indicate bars with a non-zero number of samples.

Validation is performed by sampling a new control sequence for both motors. In addition, the sequences are replicated beyond 60 s with opposite sign. Figure F.2 shows the results. Even though the fit is close to 88.21 %, the model mismatch especially for the position is not negligible.

F.2. Benchmark Hardware and Results

F.2.1. Computation Time Measurements

Chapter 8 provides benchmark results for which the measurement of computation times is of particular importance.
All benchmarks that involve the measurement of computation times are performed on a PC with an Intel Core i7-4770 CPU at 3.4 GHz and 8 GB RAM. The operating system is Ubuntu 16.04 LTS (64 Bit).
In the following, two examples demonstrate how the measured computation times are distributed. Figure F.3a shows the distribution for solving the multiple shooting nonlinear program (forward Euler) for time-optimal control of the Van der Pol Oscillator with IPOPT (explicit Hessian). Correspondingly, Figure F.3b depicts the distribution for solving the Crank-Nicolson collocation nonlinear program for the rocket system (also IPOPT with explicit Hessian computations). Both distributions reveal a larger number of samples around 9.5 ms and 15.25 ms respectively which indicate a rather local



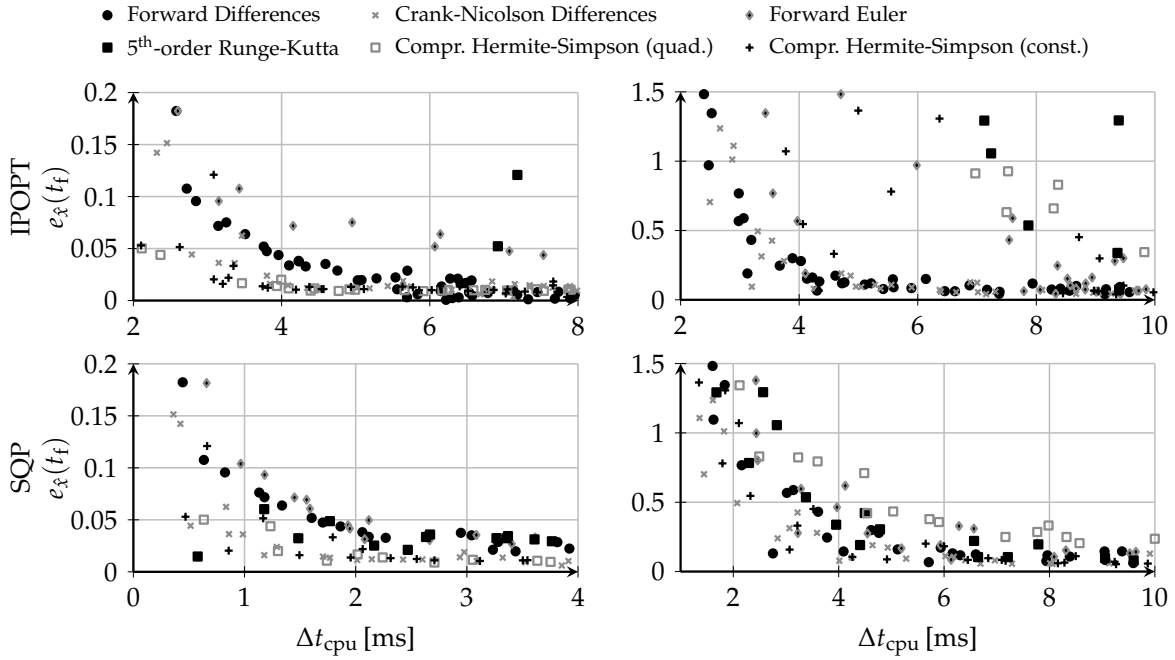
(a) Van der Pol oscillator (b) Rocket system

Figure F.4.: Computation times and integral errors for several direct transcription methods and the local uniform grid. Multiple shooting is configured with $M = N$. The solver is IPOPT with explicit Hessian computation.

normal distribution. However, several samples are distributed for a number of larger computation times (especially for the rocket system). The established Shapiro-Wilk test fails even for large $N \approx 1000$. Note, the benchmark PC is not a real-time capable system. Even though benchmarks are performed without any user interaction, the actual performance still depends on the operating system and its scheduler. However, the median is preferred over the mean for measurements to avoid large distortions caused by any operating system interruption. In order to further reduce the execution time required for all benchmarks, the number of samples is limited to 20. The corresponding median is represented as a dashed line and its deviation from the median of 1000 samples is less than half a millisecond.

F.2.2. Performance Comparison between Transcription Methods

Figure F.4 depicts the comparison between several transcription methods for the local uniform grid in addition to the global uniform grid results in Section 8.1.1. Forward Euler is slightly faster than the other methods, followed by forward differences. However, they also reveal the largest integral errors. On the other hand, Crank-Nicolson and Hermite-Simpson with constant controls perform almost comparable at lower integral errors $e_{\hat{x}}(t_{\mathrm{f}})$. The 5$^{\mathrm{th}}$-order Runge-Kutta method and Hermite-Simpson with quadratic control splines reveal much larger computation times. The results, also confirm Remark 8.1.2 which states that the dynamics of the rocket system are much simpler to approximate (also in Figure F.5).



(a) Van der Pol oscillator

(b) Rocket system

Figure F.5.: Scatter plot of integral errors with respect to the computation time (local uniform grid). IPOPT with explicit Hessian computation is utilized at the top and SQP at the bottom. Multiple shooting is configured with $M = N$ for IPOPT and $M = \lceil N/4 \rceil$ for SQP.

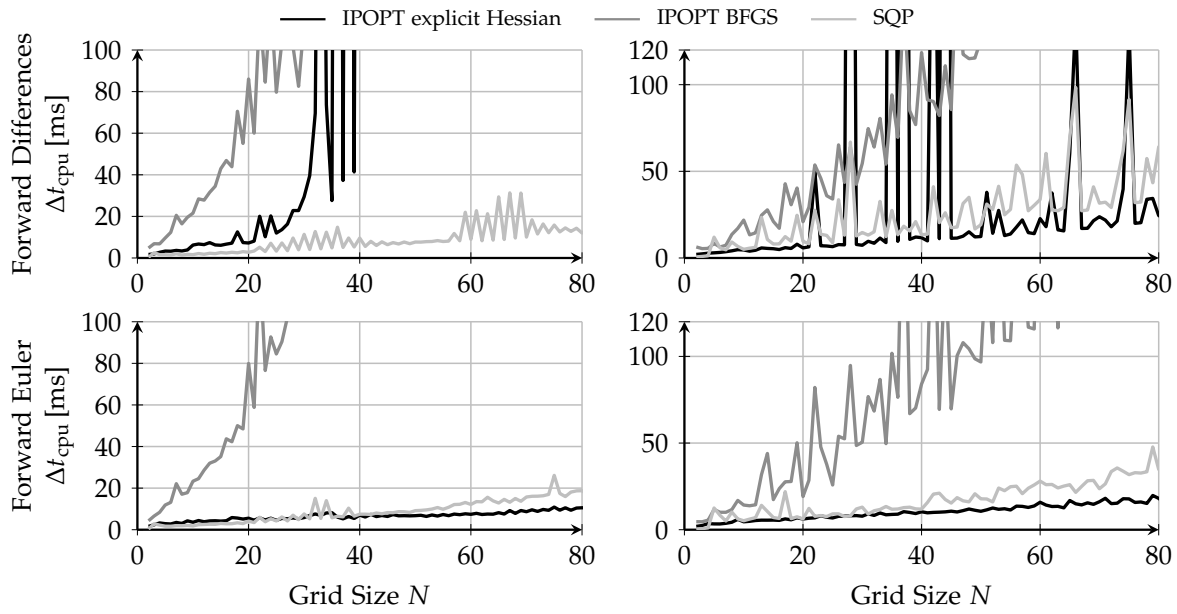
Figure F.5 shows the relation between the integral error and the computation time. Similar to the global uniform grid in Section 8.1.1, Crank-Nicolson differences provides a reasonable tradeoff in terms of computation time and accuracy such that they are well suited for both benchmark systems.

F.2.3. Comparison between Solvers

In addition to the global and local uniform grid in Section 8.1.3, Figure F.6 compares IPOPT with explicit Hessian computation, IPOPT with BFGS and the SQP solver with respect to the grid size on the quasi-uniform grid. As already pointed out in Section 8.1, the quasi-uniform grid does not perform well with IPOPT (explicit Hessian) and forward differences. Even though IPOPT converges to the optimal solution, the corresponding computation is large and exhibits large peaks which depend on how accurately the current grid size N matches the actual switching points. IPOPT with BFGS performs worse for both the two benchmark systems and two transcription methods.

F.2.4. Convergence Analysis

The benchmarks in this thesis show that the quasi-uniform grid performs worse in terms of computation times with IPOPT and collocation via finite differences. In addition, IPOPT performs generally better than SQP on larger problem sizes, while



(a) Van der Pol oscillator

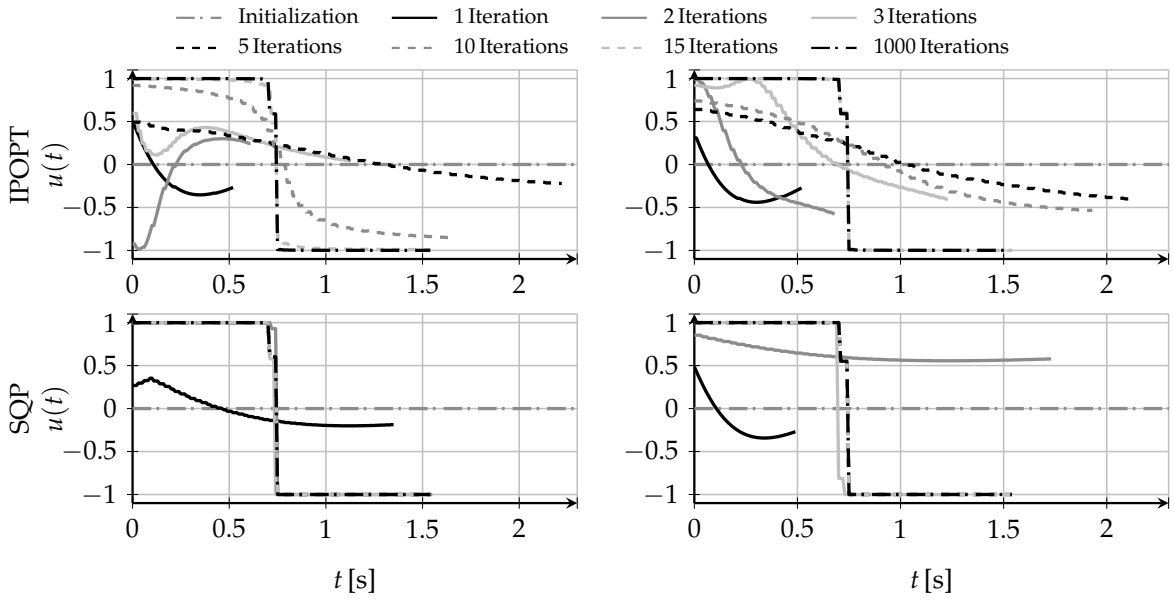
(b) Rocket system

Figure F.6.: Comparison of the solver configurations on the quasi-uniform grid. Forward differences are utilized at the top and forward Euler with $M = N$ at the bottom.

SQP is still faster for the global uniform grid with finite differences. To this end, this section examines the convergence behavior of both IPOPT with explicit Hessian and SQP for selected configurations. The control task considers the Van der Pol oscillator as in the previous benchmarks with $N = 50$ for which the quasi-uniform grid already exhibits large computation times.

Figure F.7 and F.8 show intermediate solutions of the control trajectory in case either IPOPT or SQP is terminated after a specified number of solver iterations. IPOPT is configured with the explicit Hessian computation. The initialization is chosen as before, in particular a straight line in the state space and zero controls. Temporal variables are initialized according to a total transition time of 3 s. Note that in every SQP iteration, the quadratic subproblem is optimized until a termination criteria as specified in Section E.2.2 is fulfilled. Every SQP iteration is therefore computationally more expensive than an iteration in IPOPT. The figures show that SQP is able to get close to the optimal solution in only $1 - 2$ SQP iterations. IPOPT requires much more iterations. The results show that the nonlinearities that come from the Van der Pol oscillator and the finite difference collocation method are relatively small and can be resolved with only a few quadratic cost respectively linear constraint approximations. This could also be interesting for other problems where the system dynamics are almost linear. Here it is expected that the finite difference term (division of the state difference by the time interval) by this observation does not require many SQP iterations.

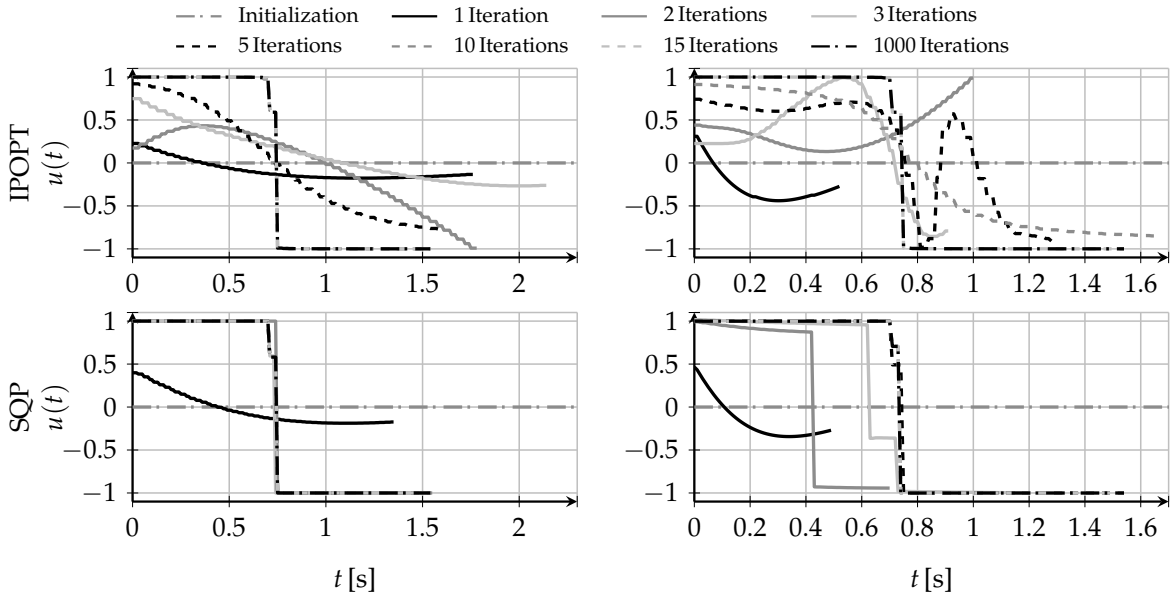
As mentioned before, the quasi-uniform grid performs worse with IPOPT and finite differences. Figure F.9 shows the results for intermediate control trajectories for the same configurations as for the local and global uniform grid. Hereby, IPOPT converged to a sigmoidal approximation of the bang-bang control trajectory in 10 iterations. But



(a) Forward Differences

(b) Forward Euler

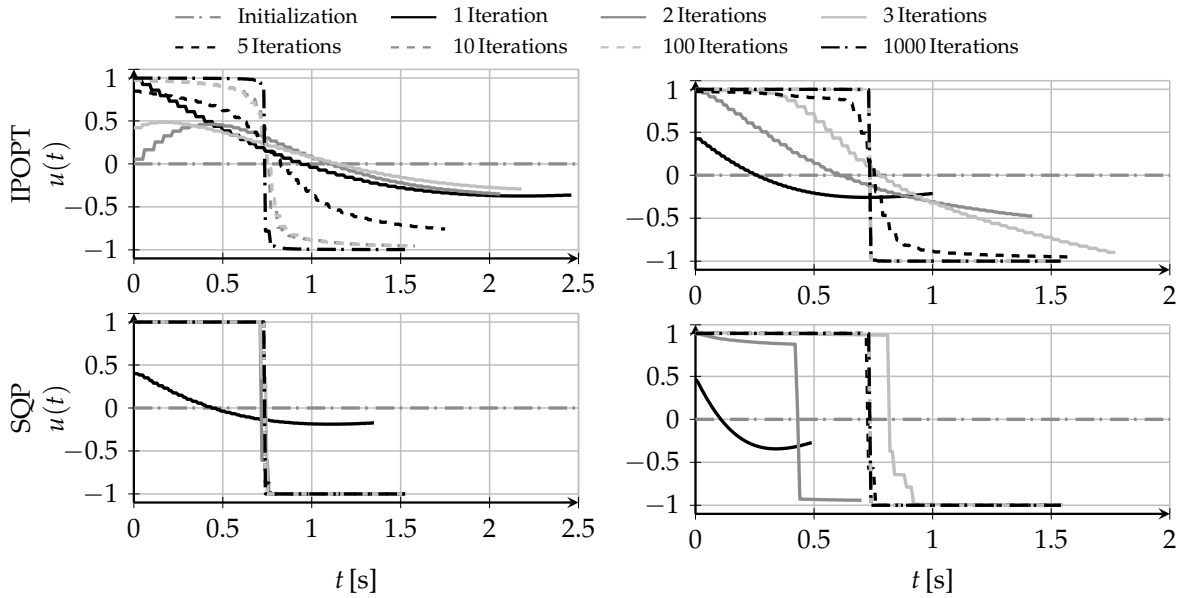
Figure F.7.: This figure shows control trajectories obtained from intermediate SQP respectively IPOPT solver iterations. Results for the Van der Pol oscillator with both collocation (forward differences) and multiple shooting (forward Euler) are shown. The underlying grid is the global uniform one with $N = 50$. The initialization is a straight line in the state space and zero controls up to 3 s. IPOPT uses the explicit Hessian computation.



(a) Forward Differences

(b) Forward Euler

Figure F.8.: Control trajectories obtained from intermediate SQP respectively IPOPT solver iterations. The configuration is similar to Figure F.7 but with the local uniform grid.



(a) Forward Differences

(b) Forward Euler

Figure F.9.: Control trajectories obtained from intermediate SQP respectively IPOPT solver iterations. The configuration is similar to Figure F.7 but with the quasi-uniform grid.

IPOPT cannot improve this solution significantly withen the next iterations. Note that the solution after 100 iterations does not deviate much from the solution after 10 iterations. However, the result after 1000 iterations shows that the IPOPT is able to converge properly. Figure F.10 illustrates this even better by showing the final time (identical respectively proportional to the cost function value) with respect to the number of solver iterations. In case of forward differences, IPOPT stagnates with the quasi-uniform grid after 7 iterations. However, the internal line search and solver heuristics are able to restore the proper convergence after about 110 iterations. On the other hand, IPOPT converges with the quasi-uniform grid and forward Euler properly.

F.2.5. Closed-Loop Performance

Section 8.3 presents closed-loop results for the local uniform grid including a dual-mode and hybrid cost realization as well as the ℓ_1-norm approach for the *ECP industrial plant emulator 220*.

In addition to Figure 8.10, Figure F.11 includes the input sequences and grid size evolution (N) for the same scenario but with the global uniform grid approach rather than the local uniform grid. Both the local and the global uniform grid perform comparable, as the solution to the individual optimal control problems are the same. Also, the computation times are similar to Figure 8.10. Note, the grid size N for the dual-mode realization remains constant as soon as the local controller is active (also visible in the intervals of negligible computation times). In this case, invoking the predictive controller is skipped at all. The ℓ_1-norm approach is initialized with a grid size of $N = 40$ to ensure a feasible solution for all changes in the reference x_f. It is also visible, that the initial uniform grid size does not match the desired temporal

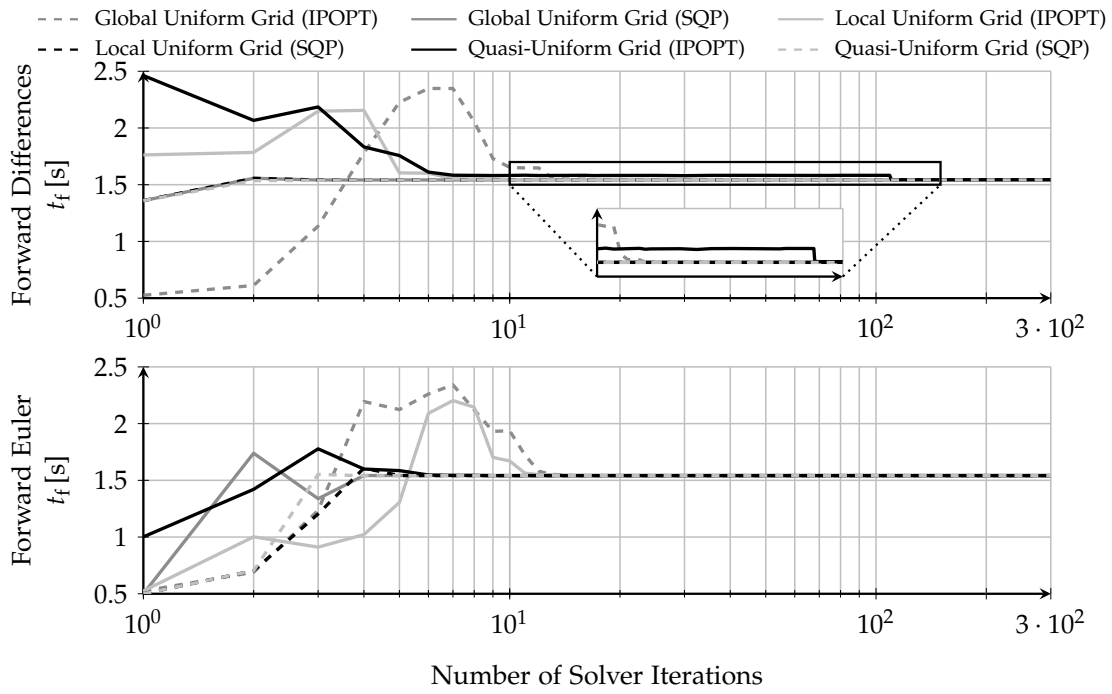


Figure F.10.: Convergence of the optimal final time respect to the number of IPOPT respectively SQP solver iterations for different discretization grids. The control task and initialization are similar to Figure F.7.

resolution such that further grid points are inserted.
Another experiment in Section 8.3 compares the performance in case of additional disturbances, in particular for a sinusoidal disturbance profile. In the following, other closed-loop results with different disturbance profiles are shown. They all demonstrate the effectiveness of the approach and that the time-optimal variable grid solution coincides with the time-optimal ℓ_1-norm formulation which also confirms validity of the time-optimal solution. The general control task is to drive the system from the origin to $x_f = (8, 0)^\intercal$. Note, the accumulated signal commanded to motor 2 (including the actual control plus the disturbance) is limited to ± 0.8 to not harm the real system. Figure F.12 considers a linearly increasing disturbance profile $u_{\text{dist}}(t_\mu)$ which is bounded to 0.5. The sweep remains active upon arrival at x_f (but not violating ± 0.8). Figure F.13 and Figure F.14 show examples in which a step disturbance is active in the interval $[1\,\text{s}, 1.5\,\text{s}]$. In contrast, Figure F.15 and Figure F.16 keep the step disturbance active beyond $t = 1\,\text{s}$.

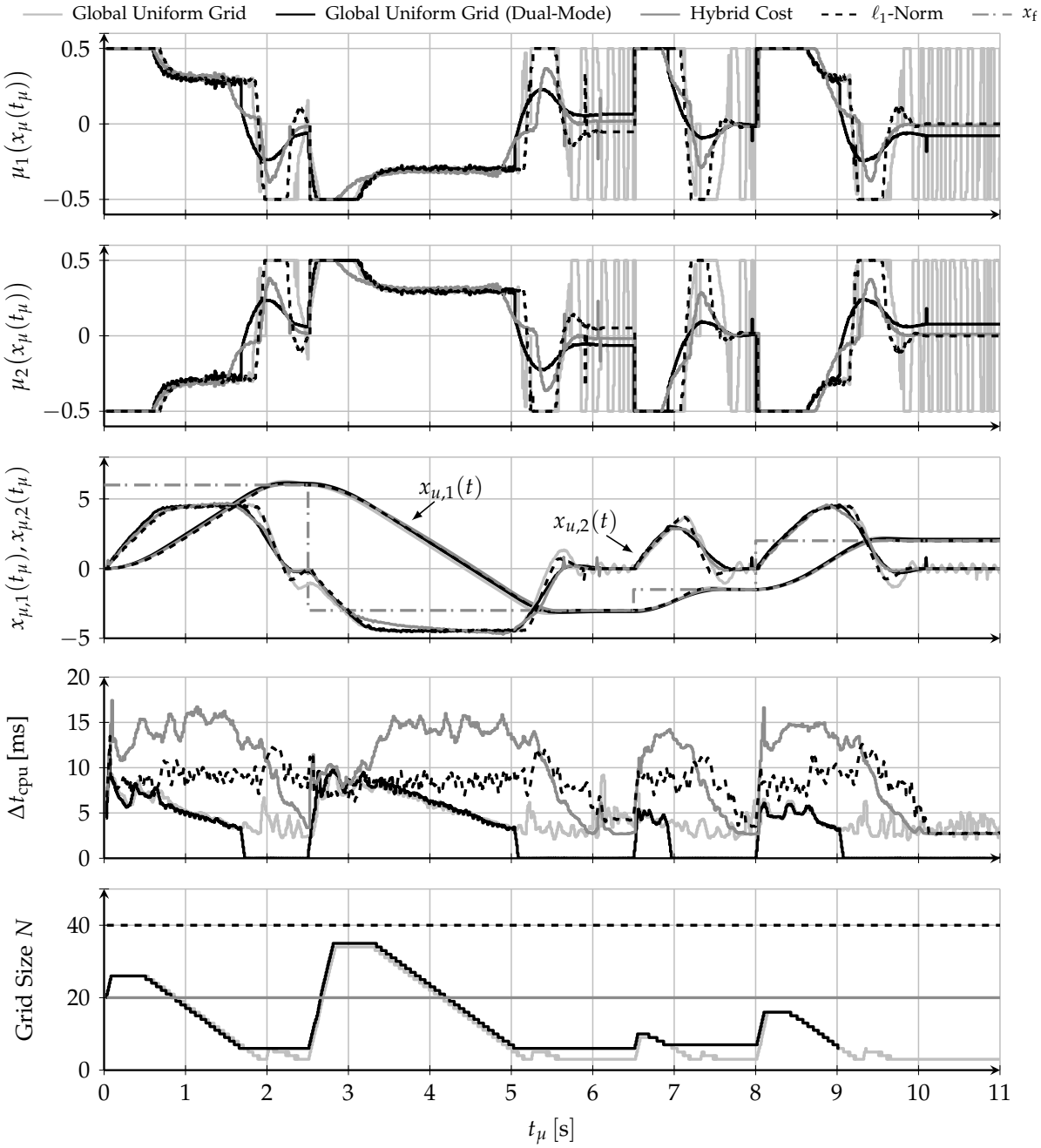


Figure F.11.: Closed-loop control of the ECP Model 220 with varying final position $x_{f,1}$ and several methods. These are the global uniform grid, a dual-mode and hybrid cost realization and the ℓ_1-norm approach.

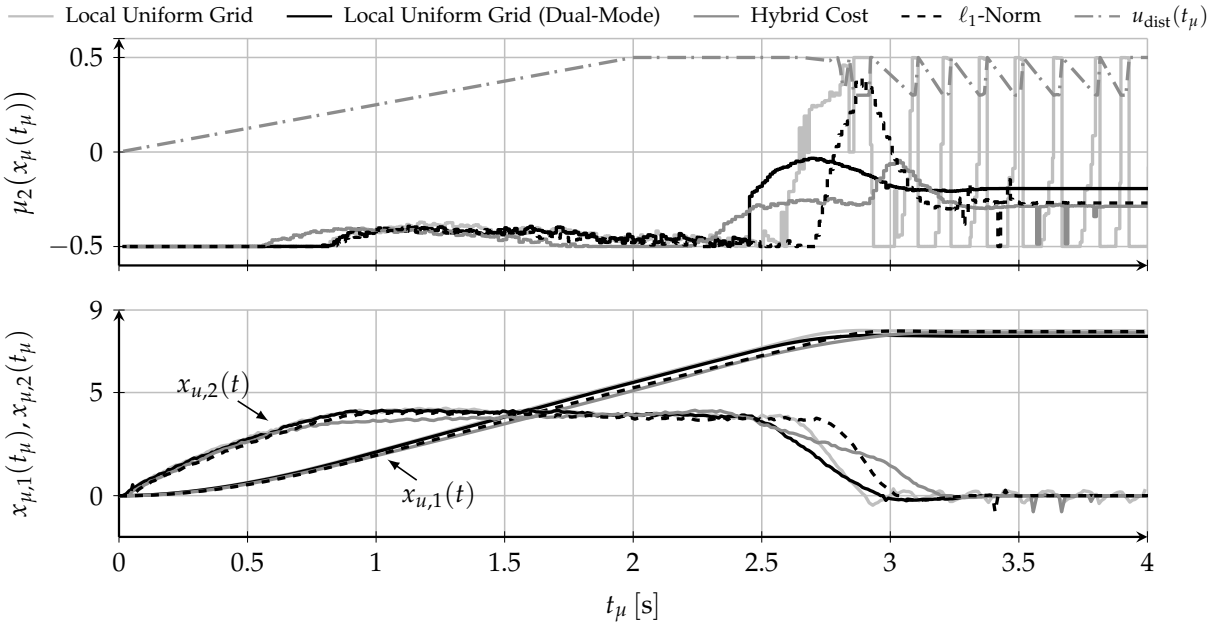


Figure F.12.: Closed-loop control of the ECP Model 220 with a linearly increasing input disturbance at motor 2. The disturbance signal is bounded and remains active upon arrival at x_f.

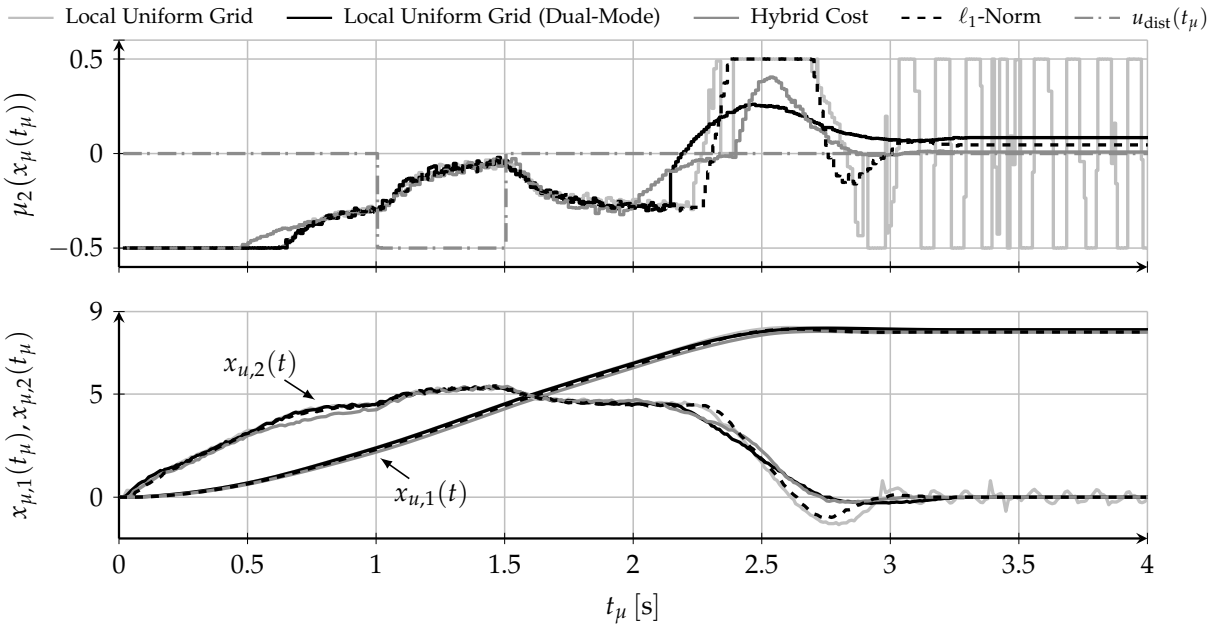


Figure F.13.: Closed-loop control of the ECP Model 220 with a negative disturbance at motor 2 in the interval $[1\,\mathrm{s}, 1.5\,\mathrm{s}]$.

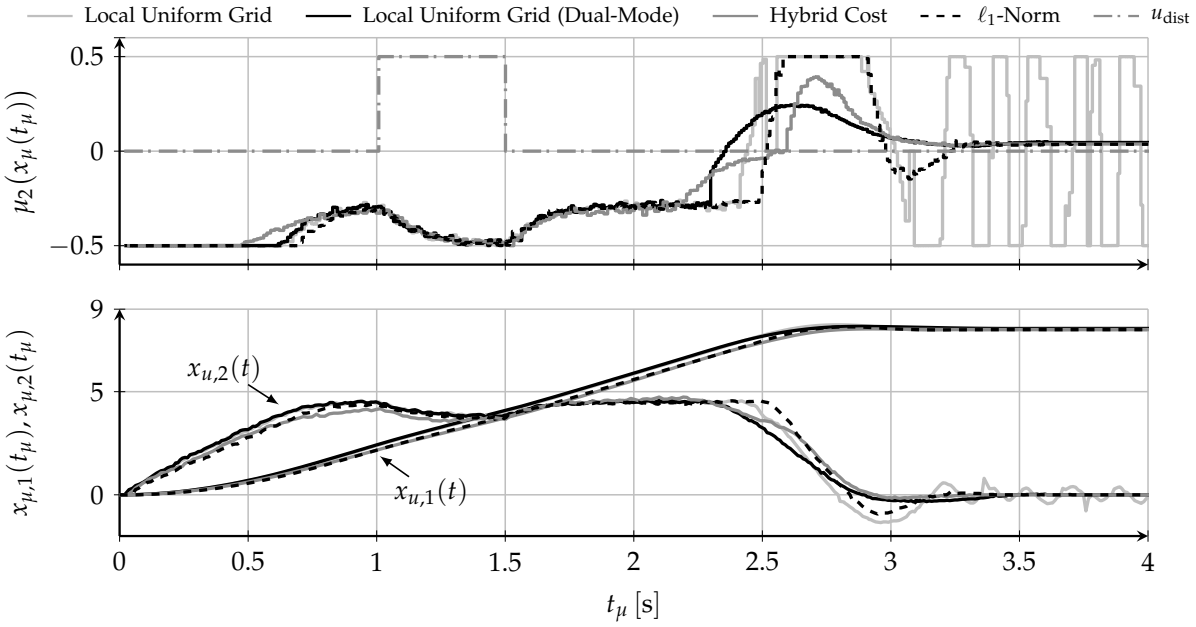


Figure F.14.: Closed-loop control of the ECP Model 220 with a positive disturbance at motor 2 in the interval $[1\,\text{s}, 1.5\,\text{s}]$.

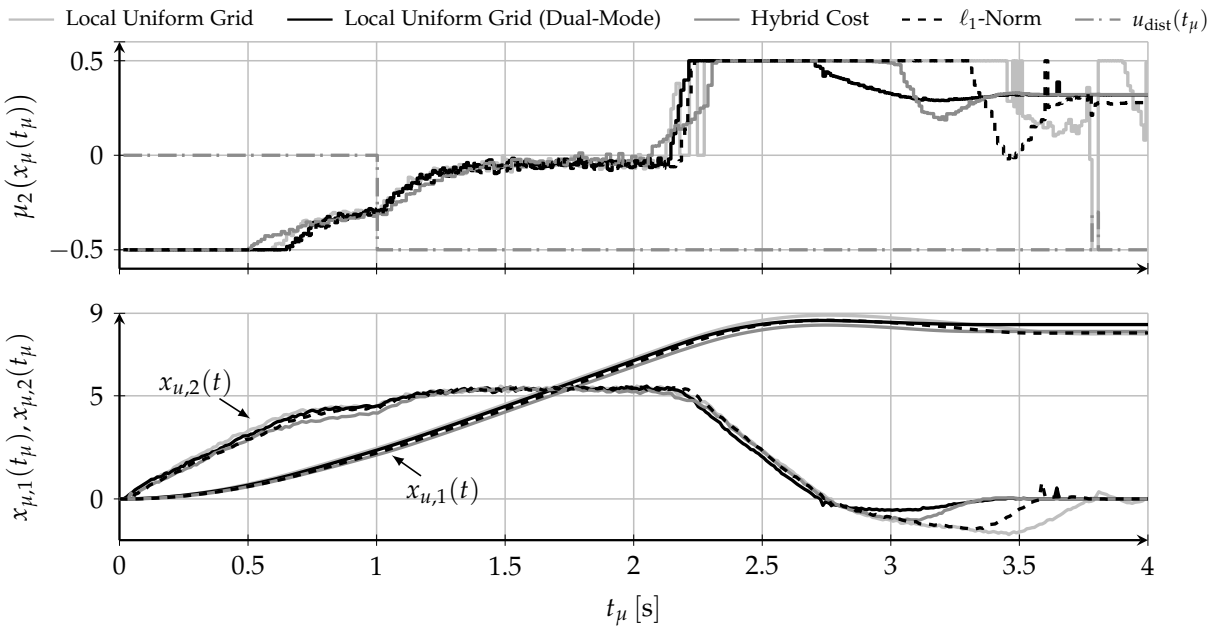


Figure F.15.: Closed-loop control of the ECP Model 220 with a negative step disturbance at motor 2 at $t = 1\,\text{s}$. The disturbance remains active.

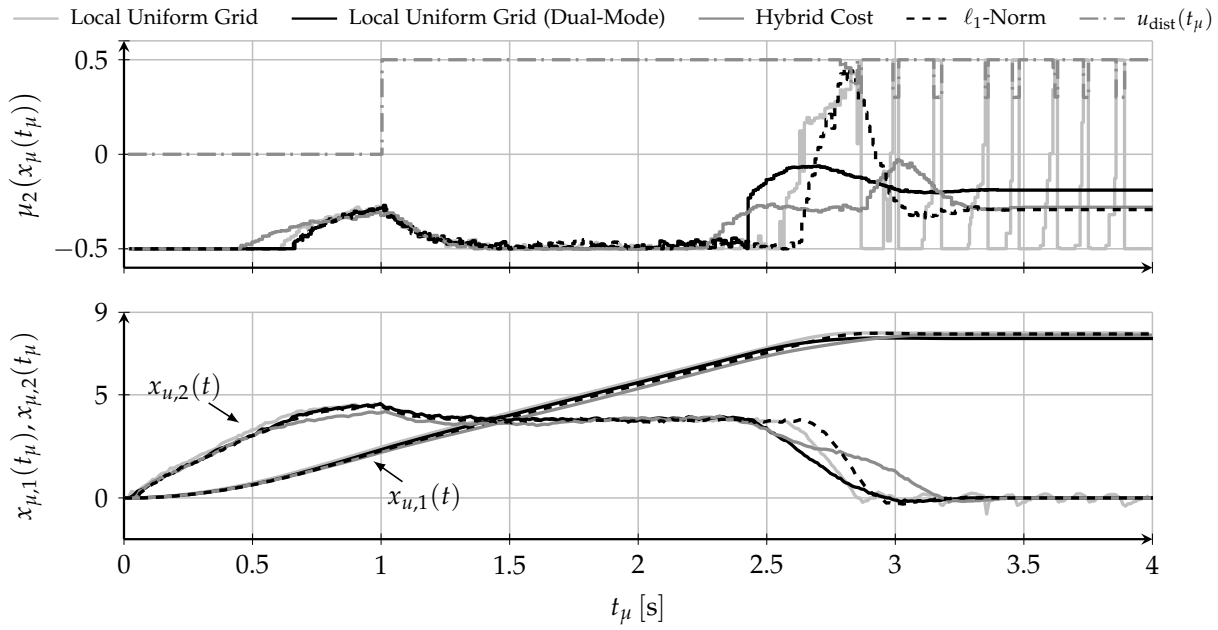


Figure F.16.: Closed-loop control of the ECP Model 220 with a positive step disturbance at motor 2 at $t = 1\,\text{s}$. The disturbance remains active.

Bibliography

[AK04] S. N. Avvakumov and Y. N. Kiselev. "Boundary Value Problem for Ordinary Differential Equations with Applications to Optimal Control". In: *World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*. 2004.

[Alb02] B. A. Albassam. "Optimal Near-Minimum-Time Control Design for Flexible Structures". In: *Journal of Guidance, Control and Dynamics* 25.4 (2002), pp. 618–625.

[And13] J. Andersson. "A General-Purpose Software Framework for Dynamic Optimization". PhD thesis. KU Leuven, 2013.

[Ath71] M. Athans. "The role and use of the stochastic linear-quadratic-Gaussian problem in control system design". In: *IEEE Transactions on Automatic Control* 16.6 (1971), pp. 529–552.

[Bel54] R. Bellmann. "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60 (1954), pp. 503–515.

[Bel57] R. Bellmann. "Dynamic Programming". PhD Thesis. Princeton University Press, 1957.

[Ber95] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[Bes+09] T. Besselmann, J. Lofberg, and M. Morari. "Constrained time-optimal control of linear parameter-varying systems". In: *Joint 48th IEEE Conference on Decision and Control (CDC) and 28th Chinese Control Conference (CCC)*. 2009, pp. 6923–6928.

[Bet10] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2nd ed. Advances in Design and Control. Society for Industrial and Applied Mathematics, 2010.

[Bet98] J. T. Betts. "Survey of Numerical Methods for Trajectory Optimization". In: *Journal of Guidance, Control, and Dynamics* 21.2 (1998), pp. 193–207.

[Bie84] L. T. Biegler. "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation". In: *Computers & Chemical Engineering* 8.3 (1984), pp. 243–247.

[BM15] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. 2nd ed. Birkhäuser Basel, 2015.

[BP84] H. G. Bock and K. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: *IFAC World Congress*. 1984, pp. 242–247.

[But16] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. 3rd ed. John Wiley & Sons, 2016.

[CA98] H. Chen and F. Allgöwer. "A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability". In: *Automatica* 34.10 (1998), pp. 1205–1217.

[Cag+04] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari. "Move blocking strategies in receding horizon control". In: *IEEE Conference on Decision and Control (CDC)*. 2004, pp. 2023–2028.

[Cag+07] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari. "Move blocking strategies in receding horizon control". In: *Journal of Process Control* 17.6 (2007), pp. 563–570.

[Chi+96] L. Chisci, A. Lombardi, and E. Mosca. "Dual-Receding Horizon Control of Constrained Discrete Time Systems". In: *European Journal of Control* 2.4 (1996), pp. 278–285.

[CL47] M. L. Cartwright and J. E. Littlewood. "On Non-Linear Differential Equations of the Second Order". In: *Annals of Mathematics* 48.2 (1947), pp. 472–494.

[Cla83] F. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, 1983.

[Com] Computational Mathematics Group. *HSL. A collection of Fortran codes for large scale scientific computation*. URL: `http://www.hsl.rl.ac.uk/`. Visited on 2019-01-13.

[CR80] C. R. Cutler and B. L. Ramaker. "Dynamic Matrix Control – A computer control algorithm". In: *Joint Automatic Control Conference*. 1980.

[CS82] C. C. Chen and L. Shaw. "On receding horizon feedback control". In: *Automatica* 18.3 (1982), pp. 349–352.

[Die+02] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations". In: *Journal of Process Control* 12.4 (2002), pp. 577–585.

[Die+05] M. Diehl, H. G. Bock, and J. P. Schlöder. "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control". In: *SIAM Journal on Control and Optimization* 43.5 (2005), pp. 1714–1736.

[ECP] ECP. *User Manual: Model 220 Industrial Plant Emulator*. URL: `http://www.ecpsystems.com/controls_emulator.htm`. Visited on 2019-01-13.

[Eri+04] K. Eriksson, D. Estep, and C. Johnson. *Applied Mathematics: Body and Soul, Volume 1: Derivatives and Geometry in IR3*. Berlin, Heidelberg: Springer, 2004.

[FA03] R. Findeisen and F. Allgöwer. "The quasi-infinite horizon approach to nonlinear model predictive control". In: *Nonlinear and Adaptive Control*. Ed. by A. Zinober and D. Owens. Vol. 281. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, 2003, pp. 89–108.

[Fat68] A. F. Fath. "Approximation to the Time-Optimal Control of Linear State-Constrained Systems". In: *Joint Automatic Control Conference*. 1968, pp. 962–969.

[Fri+16] G. Frison, D. Kouzoupis, J. B. Jørgensen, and M. Diehl. "An Efficient Implementation of Partial Condensing for Nonlinear Model Predictive Control". In: *IEEE Conference on Decision and Control (CDC)*. 2016, pp. 4457–4462.

[GI10] R. Gondhalekar and J.-i. Imura. "Least-restrictive move-blocking model predictive control". In: *Automatica* 46.7 (2010), pp. 1234–1240.

[GJ+10] G. Guennebaud, B. Jacob, et al. *Eigen v3*. 2010. URL: `http://eigen.tuxfamily.org`. Visited on 2019-01-13.

[GK12] K. Graichen and B. Käpernick. "A Real-Time Gradient Method for Nonlinear Model Predictive Control". In: *Frontiers of Model Predictive Control*. InTech, 2012.

[GL12] J.-M. Ginoux and C. Letellier. "Van der Pol and the history of relaxation oscillations: Toward the emergence of a concept". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.2 (2012), p. 023120.

[GM82] C. E. Garcia and M. Morari. "Internal model control. A unifying review and some new results". In: *Industrial & Engineering Chemistry Process Design and Development* 21.2 (1982), pp. 308–323.

[GP17] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. 2nd ed. Communications and Control Engineering. Springer, 2017.

[Gri+05] P. Grieder, M. Kvasnica, M. Baotić, and M. Morari. "Stabilizing low complexity feedback control of constrained piecewise affine systems". In: *Automatica* 41.10 (2005), pp. 1683–1694.

[Gro+16] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl. "From Linear to Nonlinear MPC: bridging the gap via the Real-Time Iteration". In: *International Journal of Control* (2016), pp. 1–19.

[Grü02] L. Grüne. *Asymptotic Behavior of Dynamical and Control Systems under Pertubation and Discretization*. Lecture Notes in Mathematics. Berlin, Heidelberg: Springer, 2002.

[Grü97] L. Grüne. "An adaptive grid scheme for the discrete Hamilton-Jacobi-Bellman equation". In: *Numerische Mathematik* 75.3 (1997), pp. 319–337.

[Hal80] J. K. Hale. *Ordinary Differential Equations*. 2nd ed. Krieger Publishing Company, 1980.

[Has76] L. Hasdorff. *Gradient Optimization and Nonlinear Control*. Krieger Pub Co, 1976.

[Hel+98] A. Helbig, O. Abel, and W. Marquardt. "Model predictive control for on-line optimization of semi-batch reactors". In: *American Control Conference (ACC)*. 1998, pp. 1695–1699.

[Hes+07] J. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time-dependent problems*. Cambridge University Press, 2007.

[HL02] B. Hu and A. Linnemann. "Toward infinite-horizon optimality in nonlinear model predictive control". In: *IEEE Transactions on Automatic Control* 47.4 (2002), pp. 679–682.

[Hom16] S. A. Homsi. "Online generation of time-optimal trajectories for industrial robots in dynamic environments". PhD thesis. Université Grenoble Alpes, 2016.

[Hor15] S. Horiuchi. "Evaluation of chassis control algorithms using controllability region analysis". In: *The Dynamics of Verhicles on Roads and Tracks*. Ed. by M. Rosenberger, M. Plöchl, K. Six, and J. Edelmann. CRC Press, Taylor & Francis Group, 2015, pp. 35–44.

[Hou+11a] B. Houska, H. J. Ferreau, and M. Diehl. "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization". In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312.

[Hou+11b] B. Houska, H. J. Ferreau, and M. Diehl. "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range". In: *Automatica* 47.10 (2011), pp. 2279–2285.

[II02] J. Imae and K. Inoue. "A unified approach to computational methods of nonlinear optimal control problems with possible jumps in states". In: *American Control Conference (ACC)*. Vol. 6. 2002, pp. 4571–4576.

[Jam74] E. M. James. "Time Optimal Control and the Van der Pol Oscillator". In: *IMA Journal of Applied Mathematics* 13.1 (1974), pp. 67–81.

[Jen+91] L. S. Jennings, M. E. Fisher, K. L. Teo, and C. J. Goh. "MISER3: Solving optimal control problems – an update". In: *Advances in Engineering Software and Workstations* 13.4 (1991), pp. 190–196.

[Jer+11] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides. "A condensed and sparse QP formulation for predictive control". In: *IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC)*. 2011, pp. 5217–5222.

[JM13] M. Jost and M. Mönnigmann. "Accelerating model predictive control by online constraint removal". In: *IEEE Conference on Decision and Control (CDC)*. 2013, pp. 5764–5769.

[Joh04] T. A. Johansen. "Approximate explicit receding horizon control of constrained nonlinear systems". In: *Automatica* 40.2 (2004), pp. 293–300.

[Jos+15] M. Jost, G. Pannocchia, and M. Mönnigmann. "Online constraint removal: Accelerating MPC with a Lyapunov function". In: *Automatica* 57 (2015), pp. 164–169.

[Jos+17] M. Jost, G. Pannocchia, and M. Mönnigmann. "Accelerating linear model predictive control by constraint removal". In: *European Journal of Control* 35 (2017), pp. 42–49.

[Kai80] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[Kal60] R. E. Kalman. "Contributions to the Theory of Optimal Control". In: *Matematica Mexicana* 5 (1960), pp. 102–119.

[Kan+05] C. Kanzow, N. Yamashita, and M. Fukushima. "Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints". In: *Journal of Computational and Applied Mathematics* 177.2 (2005), pp. 375–397.

[Kas+11] N. Kashiri, M. Ghasemi, and M. Dardel. "An iterative method for time optimal control of dynamic systems". In: *Archives of Control Sciences* 21.1 (2011), pp. 5–23.

[Kay03] C. Y. Kaya. "Computational Method for Time-Optimal Switching Control". In: *Journal of Optimization Theory and Applications* 117.1 (2003), pp. 69–92.

[Kel17] M. Kelly. "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation". In: *SIAM Rev.* 59.4 (2017), pp. 849–904.

[KG16] B. Käpernick and K. Graichen. "Nonlinear model predictive control based on constraint transformation". In: *Optimal Control Applications and Methods* 37.4 (2016), pp. 807–828.

[KM00] E. C. Kerrigan and J. M. Maciejowski. "Soft Constraints and Exact Penalty Functions in Model Predictive Control". In: *UKACC International Conference*. Cambridge and UK, 2000.

[KN96] C. Y. Kaya and J. L. Noakes. "Computations and time-optimal controls". In: *Optimal Control Applications and Methods* 17.3 (1996), pp. 171–185.

[Kou+11] K. I. Kouramas, N. P. Faísca, C. Panos, and E. N. Pistikopoulos. "Explicit/multi-parametric model predictive control (MPC) of linear discrete-time systems by dynamic and multi-parametric programming". In: *Automatica* 47.8 (2011), pp. 1638–1645.

[Kou+15a] D. Kouzoupis, H. J. Ferreau, and M. Diehl. "First-Order Methods in Embedded Nonlinear Model Predictive Control". In: *European Control Conference (ECC)*. 2015, pp. 2617–2622.

[Kou+15b] D. Kouzoupis, R. Quirynen, J. Frasch, and M. Diehl. "Block Condensing for Fast Nonlinear MPC with the Dual Newton Strategy". In: *IFAC Nonlinear Model Predictive Control Conference (NMPC)*. 2015, pp. 26–31.

[KS10] D. P. Kelly and R. S. Sharp. "Time-optimal control of the race car: a numerical method to emulate the ideal driver". In: *Vehicle System Dynamics* 48.12 (2010), pp. 1461–1474.

[KS12] D. P. Kelly and R. S. Sharp. "Time-optimal control of the race car: influence of a thermodynamic tyre model". In: *Vehicle System Dynamics* 50.4 (2012), pp. 641–662.

[KS72] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems. First Edition*. Wiley-Interscience, 1972.

[Kuf+15] D. K. M. Kufoalor, B. J. T. Binder, H. J. Ferreau, L. Imsland, T. A. Johanse, and M. Diehl. "Automatic Deployment of Industrial Embedded Model Predictive Control using qpOASES". In: *European Control Conference (ECC)*. 2015, pp. 2601–2608.

[Küm+11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "G2o: A general framework for graph optimization". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 3607–3613.

[LA09] M. I. A. Lourakis and A. A. Argyros. "SBA: A Software Package for Generic Sparse Bundle Adjustment". In: *ACM Transactions on Mathematical Software* 36.1 (2009), pp. 1–30.

[Lam12] D. Lam. "A model predictive approach to optimal path-following and contouring control". PhD Thesis. The University of Melbourne, 2012.

[Lee+97] H. W. J. Lee, K. L. Teo, and C. J. Goh. "Control Parametrization Enhancing Technique for Time Optimal Control Problems". In: *Dynamical Systems and Applications* 6 (1997), pp. 243–261.

[Lee+99] H. W. J. Lee, K. L. Teo, V. Rehbock, and L. S. Jennings. "Control parametrization enhancing technique for optimal discrete-valued control problems". In: *Automatica* 35.8 (1999), pp. 1401–1407.

[Lei+03] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. "An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization. Part 1: Theoretical aspects, part 2: Software aspects and applications". In: *Computers & Chemical Engineering* 27.2 (2003), pp. 157–174.

[Lei81] G. Leitmann. *The Calculus of Variations and Optimal Control*. Plenum Press New York, 1981.

[Li+06] R. Li, K. L. Teo, K. H. Wong, and G. R. Duan. "Control parameterization enhancing transform for optimal control of switched systems". In: *Mathematical and Computer Modelling* 43.11-12 (2006), pp. 1393–1403.

[Lin+96] Y. Lin, E. Sontag, and Y. Wang. "A Smooth Converse Lyapunov Theorem for Robust Stability". In: *SIAM Journal on Control and Optimization* 34.1 (1996), pp. 124–160.

[LK65] H. A. Luther and H. P. Konen. "Some Fifth-Order Classical Runge-Kutta Formulas". In: *SIAM Review* 7.4 (1965), pp. 551–558.

[LM67] E. B. Lee and L. Markus. *Foundations of optimal control theory*. John Wiley & Sons, 1967.

[Luu10] R. Luus. *Iterative Dynamic Programming*. Chapman & Hall, 2010.

[Mak+18b] A. Makarow, M. Keller, C. Rösmann, and T. Bertram. "Model Predictive Trajectory Set Control with Adaptive Input Domain Discretization". In: *American Control Conference (ACC)*. 2018, pp. 3159–3164.

[May+00] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. Scokaert. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6 (2000), pp. 789–814.

[May14] D. Q. Mayne. "Model predictive control: Recent developments and future promise". In: *Automatica* 50.12 (2014), pp. 2967–2986.

[May95] D. Q. Mayne. "Optimization in Model Based Control". In: *IFAC Symposium on Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes*. 1995, pp. 229–242.

[MH99] M. Morari and J. H. Lee. "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667–682.

[MK11] M. Mönnigmann and M. Kastsian. "Fast explicit MPC with multiway trees". In: *IFAC World Congress*. 2011, pp. 1356–1361.

[MM93] H. Michalska and D. Q. Mayne. "Robust receding horizon control of constrained nonlinear systems". In: *IEEE Transactions on Automatic Control* 38.11 (1993), pp. 1623–1633.

[MO02] H. Maurer and H. J. Oberle. "Second Order Sufficient Conditions for Optimal Control Problems with Free Final Time: The Riccati Approach". In: *SIAM Journal on Control and Optimization* 41.2 (2002), pp. 380–403.

[Mor+12] J. L. Morales, J. Nocedal, and Y. Wu. "A Sequential Quadratic Programming Algorithm with an Additional Equality Constrained Phase". In: *IMA Journal of Numerical Analysis* 32.2 (2012), pp. 553–579.

[Mor83] M. Morari. "Internal Model Control - Theory and Applications". In: *International IFAC/IMEKO Conference on the Instrumentation and Automation in the Paper, Rubber, Plastics and Polymerisation Industries*. 1983, pp. 1–18.

[MS04] L. Magni and R. Scattolini. "Model Predictive Control of Continuous-Time Nonlinear Systems With Piecewise Constant Control". In: *IEEE Transactions on Automatic Control* 49.6 (2004), pp. 900–906.

[NA15] I. Nielsen and D. Axehill. "A parallel structure exploiting factorization algorithm with applications to Model Predictive Control". In: *IEEE Conference on Decision and Control (CDC)*. 2015, pp. 3932–3938.

[NT04] D. Nešić and A. R. Teel. "A framework for stabilization of nonlinear sampled-data systems based on their approximate discrete-time models". In: *IEEE Transactions on Automatic Control* 49.7 (2004), pp. 1103–1122.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd ed. Springer series in operations research. New York: Springer, 2006.

[NW99] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer series in operations research. New York: Springer, 1999.

[Pak+13] K. S. Pakazad, H. Ohlsson, and L. Ljung. "Sparse Control Using Sum-of-norms Regularized Model Predictive Control". In: *IEEE Conference on Decision and Control (CDC)*. 2013, pp. 5758–5763.

[Pas12] B. Passenberg. "Theory and Algorithms for Indirect Methods in Optimal Control of Hybrid Systems". PhD Thesis. Technische Universität München, 2012.

[Per91] L. Perko. *Differential Equations and Dynamical Systems*. 3rd ed. Berlin, Heidelberg: Springer, 1991.

[PH14] U. Piechottka and V. Hagenmeyer. "A discussion of the actual status of process control in theory and practice: a personal view from German process industry". In: *at – Automatisierungstechnik* 62.2 (2014).

[Pin+13] G. Pin, M. Filippo, F. A. Pellegrino, G. Fenu, and T. Parisini. "Approximate model predictive control laws for constrained nonlinear discrete-time systems: analysis and offline design". In: *International Journal of Control* 86.5 (2013), pp. 804–820.

[Pon87] L. S. Pontryagin. *Mathematical Theory of Optimal Processes*. CRC Press, 1987.

[Pow11] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. 2nd ed. John Wiley & Sons, 2011.

[PP14] G. Pin and T. Parisini. "On the Robustness of Nominal Nonlinear Minimum-Time Control and Extension to Non-Robustly Controllable Target Sets". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 863–875.

[QB03] S. J. Qin and T. A. Badgwell. "A survey of industrial model predictive control technology". In: *Control Engineering Practice* 11 (2003), pp. 733–764.

[QB97] S. J. Qin and T. A. Badgwell. "An Overview Of Industrial Model Predictive Control Technology". In: *International Conference on Chemical Process Control*. 1997, pp. 232–256.

[QD73] V. H. Quintana and E. J. Davison. "A numerical method for solving optimal control problems with unspecified terminal time". In: *International Journal of Control* 17.1 (1973), pp. 97–115.

[Qui+15a] R. Quirynen, S. Gros, and M. Diehl. "Inexact Newton based Lifted Implicit Integrators for fast Nonlinear MPC". In: *IFAC Nonlinear Model Predictive Control Conference (NMPC)*. 2015, pp. 32–38.

[Qui+15b] R. Quirynen, S. Gros, and M. Diehl. "Lifted implicit integrators for direct optimal control". In: *Conference on Decision and Control (CDC)*. 2015, pp. 3212–3217.

[Rai+12] D. M. Raimondo, O. Huber, M. Schulze Darup, M. Mönnigmann, and M. Morari. "Constrained time-optimal control for nonlinear systems: a fast explicit approximation". In: *IFAC Conference on Nonlinear Model Predictive Control (NMPC)*. 2012, pp. 113–118.

[Raw+17] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd ed. Nob Hill Publishing, 2017.

[Reh+99] V. Rehbock, K. L. Teo, L. S. Jennings, and H. W. J. Lee. "A Survey of the Control Parametrization and Control Parametrization Enhancing Methods for Constrained Optimal Control Problems". In: *Progress in Optimization: Contributions from Australasia*. Ed. by A. Eberhard, R. Hill, D. Ralph, and B. M. Glover. Boston, MA: Springer US, 1999, pp. 247–275.

[Rhe+14] S. Rhein, T. Utz, and K. Graichen. "Efficient state constraint handling for MPC of the heat equation". In: *UKACC International Conference on Control*. 2014, pp. 656–661.

[Ric+67] J. Richalet, A. Rault, J. Testud, and J. Papon. "Algorithmic control of industrial processes". In: *IFAC Symposium on Identification and System Parameter Estimation*. 1967, pp. 1119–1167.

[Ric+77] J. Richalet, A. Rault, J. Testud, and J. Papon. "Model algorithmic control of industrial processes". In: *IFAC/IFIP International Conference on Digital Computer Applications to Process Control*. 1977, pp. 103–120.

[Ric13] A. Richards. "Fast model predictive control with soft constraints". In: *European Control Conference (ECC)*. 2013, pp. 1–6.

[RK12] I. M. Ross and M. Karpenko. "A review of pseudospectral optimal control: From theory to flight". In: *Annual Reviews in Control* 36.2 (2012), pp. 182–197.

[Rös+13a] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. "Efficient trajectory optimization using a sparse model". In: *IEEE European Conference on Mobile Robots (ECMR)*. 2013, pp. 138–143.

[Rös+14a] C. Rösmann, F. Hoffmann, and T. Bertram. "Prädiktive Regelung mit Timed-Elastic-Bands". In: *at - Automatisierungstechnik* 62.10 (2014), pp. 720–731.

[Rös+15a] C. Rösmann, F. Hoffmann, and T. Bertram. "Planning of Multiple Robot Trajectories in Distinctive Topologies". In: *IEEE European Conference on Mobile Robots (ECMR)*. 2015, pp. 1–6.

[Rös+15c] C. Rösmann, F. Hoffmann, and T. Bertram. "Timed-Elastic-Bands for Time-Optimal Point-to-Point Nonlinear Model Predictive Control". In: *European Control Conference (ECC)*. 2015, pp. 3357–3362.

[Rös+16a] C. Rösmann, F. Hoffmann, and T. Bertram. "Convergence Analysis of Time-Optimal Model Predictive Control under Limited Computational Resources". In: *European Control Conference (ECC)*. 2016, pp. 465–570.

[Rös+17a] C. Rösmann, F. Hoffmann, and T. Bertram. "Integrated online trajectory planning and optimization in distinctive topologies". In: *Robotics and Autonomous Systems* 88 (2017), pp. 142–153.

[Rös+17c] C. Rösmann, F. Hoffmann, and T. Bertram. "Kinodynamic Trajectory Optimization and Control for Car-Like Robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 5681–5686.

[Rös+17g] C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. "Sparse shooting at adaptive temporal resolution for time-optimal model predictive control". In: *IEEE Conference on Decision and Control (CDC)*. 2017, pp. 5551–5556.

[Rös+17i] C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. "Time-Optimal Nonlinear Model Predictive Control with Minimal Control Interventions". In: *IEEE Conference on Control Technology and Applications (CCTA)*. 2017, pp. 19–24.

[Rös+18a] C. Rösmann, M. Krämer, A. Makarow, F. Hoffmann, and T. Bertram. "Exploiting Sparse Structures in Nonlinear Model Predictive Control with Hypergraphs". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2018, pp. 1332–1337.

[Sag06] S. Sager. "Numerical Methods for Mixed–Integer Optimal Control Problems". PhD Thesis. Universität Heidelberg, 2006.

[Sch+97] C. Scherer, P. Gahinet, and M. Chilali. "Multiobjective output-feedback control via LMI optimization". In: *IEEE Transactions on Automatic Control* 42.7 (1997), pp. 896–9119.

[SG79] L. F. Shampine and C. W. Gear. "A User's View of Solving Stiff Ordinary Differential Equations". In: *SIAM Review* 21.1 (1979), pp. 1–17.

[Ste+17] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. "OSQP: An Operator Splitting Solver for Quadratic Programs". In: *ArXiv e-prints* (Nov. 2017). arXiv: `1711.08013 [math.OC]`.

[Str93] O. von Stryk. "Numerical solution of optimal control problems by direct collocation". In: *Optimal Control*. Ed. by R. Bulirsch, A. Miele, J. Stoer, and K. Well. Vol. 111. ISNM International Series of Numerical Mathematics. Birhäuser Basel, 1993, pp. 129–143.

[Sus79] H. J. Sussmann. "A Bang-Bang Theorem with Bounds on the Number of Switchings". In: *SIAM Journal on Control and Optimization* 17.5 (1979), pp. 629–651.

[Sus87a] H. J. Sussmann. "Regular Synthesis for Time-Optimal Control of Single-Input Real Analytic Systems in the Plane". In: *SIAM Journal on Control and Optimization* 25.5 (1987), pp. 1145–1162.

[Sus87b] H. J. Sussmann. "The Structure of Time-Optimal Trajectories for Single-Input Systems in the Plane: The General Real Analytic Case". In: *SIAM Journal on Control and Optimization* 25.4 (1987), pp. 868–904.

[TC10] J. P. Timings and D. J. Cole. "Minimum manoeuvre time of a nonlinear vehicle at constant forward speed using convex optimisation". In: *International Symposium on Advanced Vehicle Control*. 2010.

[Teo+91] K. L. Teo, C. J. Goh, and K. H. Wong. *A unified computational approach to optimal control problems*. Pitman Monographs and Surveys in Pure and Applied Mathematics. Longman Scientific & Technicalc, 1991.

[Teo+99] K. L. Teo, L. S. Jennings, H. W. J. Lee, and V. Rehbock. "The control parameterization enhancing transform for constrained optimal control problems". In: *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics* 40.3 (1999), pp. 314–335.

[TK71] D. Tabak and B. C. Kuo. *Optimal Control by Mathematical Programming*. Instrumentation and Control Series. Prentice Hall, 1971.

[Tri+16] V.-V. Trinh, M. Alamir, P. Bonnay, and F. Bonne. "Explicit Model Predictive Control via Nonlinear Piecewise Approximations". In: *IFAC Symposium on Nonlinear Control Systems (NOLCOS)*. 2016, pp. 259–264.

[Tsa+75] T. H. Tsang, D. M. Himmelblau, and T. F. Edgar. "Optimal control via collocation and non-linear programming". In: *International Journal of Control* 21.5 (1975), pp. 763–768.

[Van+11a] L. Van den Broeck, M. Diehl, and J. Swevers. "A model predictive control approach for time optimal point-to-point motion control". In: *Mechatronics* 21.7 (2011), pp. 1203–1212.

[Van+11b] L. Van den Broeck, M. Diehl, and J. Swevers. "Model predictive control for time-optimal point-to-point motion control". In: *IFAC World Congress*. 2011, pp. 2458–2463.

[Van26] B. Van der Pol. "On relaxation-oscillations". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1926), pp. 978–992.

[Ver+14] R. Verschueren, S. D. Bruyne, M. Zanon, J. V. Frasch, and M. Diehl. "Towards Time-Optimal Race Car Driving using Nonlinear MPC in Real-Time". In: *IEEE Conference on Decision and Control (CDC)*. 2014, pp. 2505–2510.

[Ver+16a] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl. "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm". In: *European Control Conference (ECC)*. 2016, pp. 141–147.

[Ver+16b] R. Verschueren, N. van Duijkeren, J. Swevers, and M. Diehl. "Time-optimal Motion Planning for n-DOF Robot Manipulators using a Path-Parametric System Reformulation". In: *American Control Conference (ACC)*. 2016, pp. 2092–2097.

[Ver+17] R. Verschueren, H. J. Ferreau, A. Zanarini, M. Mercangöz, and M. Diehl. "A stabilizing nonlinear model predictive control scheme for time-optimal point-to-point motions". In: *IEEE Conference on Decision and Control (CDC)*. 2017, pp. 2525–2530.

[Vos10] G. Vossen. "Switching Time Optimization for Bang-Bang and Singular Controls". In: *Journal of Optimization Theory and Applications* 144.2 (2010), pp. 409–429.

[Vuk+13] 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: *IEEE Conference on Decision and Control (CDC)*. 2013, pp. 5113–5118.

[WB06] A. Wächter and L. T. Biegler. "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57.

[WB08] Y. Wang and S. P. Boyd. "Fast Model Predictive Control Using Online Optimization". In: *IFAC World Congress*. 2008, pp. 6974–6979.

[WB10] Y. Wang and S. P. Boyd. "Fast Model Predictive Control Using Online Optimization". In: *IEEE Transactions on Control Systems Technology* 18.2 (2010), pp. 267–278.

[Wil71] J. C. Willems. "Least Squares Stationary Optimal Control and the Algebraic Riccati Equation". In: *IEEE Transactions on Automatic Control* 16.6 (1971), pp. 621–634.

[Zan+17] A. Zanelli, R. Quirynen, G. Frison, and M. Diehl. "A partially tightened real-time iteration scheme for nonlinear model predictive control". In: *IEEE Conference on Decision and Control (CDC)*. 2017, pp. 4388–4393.

[Zei+14] M. N. Zeilinger, D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones. "On real-time robust model predictive control". In: *Automatica* 50.3 (2014), pp. 683–694.

[Zha+04] J. Zhao, M. Diehl, R. Longman, H. G. Bock, and J. P. Schlöder. "Nonlinear Model Predictive Control of Robots Using Real-time Optimization". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. 2004.

[Zho+96] K. Zhou, J. C. Doyle, and K. Glover. *Robust and optimal control*. Upper Saddle River and N.J: Prentice Hall, 1996.

Related Peer-Reviewed Publications

Publications in the context of time-optimal model predictive control and sparsity exploitation in model predictive control are listed below:

C. Rösmann, M. Krämer, A. Makarow, F. Hoffmann, and T. Bertram. "Exploiting Sparse Structures in Nonlinear Model Predictive Control with Hypergraphs". In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. 2018, pp. 1332–1337.

C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. "Gain-Scheduling zwischen zeitoptimaler und quadratischer modellprädiktiven Regelung". In: *International Federation for the Promotion of Mechanism and Machine Science D-A-CH (IFToMM D-A-CH)*. 2018.

C. Rösmann, F. Hoffmann, and T. Bertram. "Zeitoptimale Bewegungsplanung für Mobile Roboter mit Ackermann-Lenkung". In: *VDI Mechatroniktagung*. 2017, pp. 238–243.

C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. "Sparse shooting at adaptive temporal resolution for time-optimal model predictive control". In: *IEEE Conference on Decision and Control (CDC)*. 2017, pp. 5551–5556.

C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. "Time-Optimal Nonlinear Model Predictive Control with Minimal Control Interventions". In: *IEEE Conference on Control Technology and Applications (CCTA)*. 2017, pp. 19–24.

C. Rösmann, F. Hoffmann, and T. Bertram. "Convergence Analysis of Time-Optimal Model Predictive Control under Limited Computational Resources". In: *European Control Conference (ECC)*. 2016, pp. 465–570.

C. Rösmann, F. Hoffmann, and T. Bertram. "Timed-Elastic-Bands for Time-Optimal Point-to-Point Nonlinear Model Predictive Control". In: *European Control Conference (ECC)*. 2015, pp. 3357–3362.

C. Rösmann, F. Hoffmann, and T. Bertram. "Prädiktive Regelung mit Timed-Elastic-Bands". In: *at - Automatisierungstechnik* 62.10 (2014), pp. 720–731.

E. Mares, C. Rösmann, T. Bertram, F. Keller, M. Skutek, and T. Ottenhues. "Zeitoptimale Trajektorienplanung für automatisierte Parkvorgänge von Fahrzeugen". In: *VDI Mechatroniktagung*. 2019, pp. 115–120.

M. Krämer, C. Rösmann, F. I. John, and T. Bertram. "Zeitoptimaler Mastaufbau einer mobilen Autobetonpumpe unter Nebenbedingungen". In: *Forschung im Ingenieurwesen* 82.45 (2018), pp. 45–57.

M. Krämer, C. Rösmann, F. I. John, and T. Bertram. "Zeitoptimale Regelung des Mastaufbaus einer Autobetonpumpe unter statischen, kinodynamischen und geometrischen Nebenbedingungen". In: *International Federation for the Promotion of Mechanism and Machine Science D-A-CH (IFToMM D-A-CH)*. 2017.

C. Götte, M. Keller, C. Rösmann, C. Haß, K.-H. Glander, A. Seewald, and T. Bertram. "A Real-Time Capable Model Predictive Approach to Lateral Vehicle Guidance". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 1908–1913.

Additional Peer-Reviewed Publications

C. Rösmann, F. Hoffmann, and T. Bertram. "Integrated online trajectory planning and optimization in distinctive topologies". In: *Robotics and Autonomous Systems* 88 (2017), pp. 142–153.

C. Rösmann, F. Hoffmann, and T. Bertram. "Kinodynamic Trajectory Optimization and Control for Car-Like Robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 5681–5686.

C. Rösmann, F. Hoffmann, and T. Bertram. "Online Trajectory Planning in ROS under Kinodynamic Constraints with Timed-Elastic-Band". In: *Robot Operating System (ROS) - The Complete Reference 2*. Ed. by A. Koubaa. Vol. 707. Studies in Computational Intelligence. Springer International Publishing, 2017, pp. 231–261.

C. Rösmann, M. Oeljeklaus, F. Hoffmann, and T. Bertram. "Online Trajectory Prediction and Planning for Social Robot Navigation". In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. 2017, pp. 1255–1260.

C. Rösmann, F. Hoffmann, and T. Bertram. "Planning of Multiple Robot Trajectories in Distinctive Topologies". In: *IEEE European Conference on Mobile Robots (ECMR)*. 2015, pp. 1–6.

C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. "Efficient trajectory optimization using a sparse model". In: *IEEE European Conference on Mobile Robots (ECMR)*. 2013, pp. 138–143.

C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. "Trajectory modification considering dynamic constraints of autonomous robots". In: *7th German Conference on Robotics (ROBOTIK)*. 2012, pp. 74–79.

F. Albers, C. Rösmann, F. Hoffmann, and T. Bertram. "Online Trajectory Optimization and Navigation for Dynamic Environments in ROS". In: *Robot Operating System (ROS) - The Complete Reference 3*. Ed. by A. Koubaa. Studies in Computational Intelligence. Springer International Publishing, 2018, pp. 241–274.

A. Makarow, J. Braun, C. Rösmann, G. Schoppel, I. Glowatzky, and T. Bertram. "Introduction of Model Predictive Control for the System Optimization of a Proportional Directional Control Valve". In: *IEEE Conference on Control Technology and Applications (CCTA)*. 2018, pp. 921–926.

A. Makarow, M. Keller, C. Rösmann, and T. Bertram. "Model Predictive Trajectory Set Control with Adaptive Input Domain Discretization". In: *American Control Conference (ACC)*. 2018, pp. 3159–3164.

A. Makarow, C. Rösmann, F. Hoffmann, and T. Bertram. “Vergleich der modellprädiktiven Trajektorienscharregelung mit konventionellen Regelungsverfahren”. In: *International Federation for the Promotion of Mechanism and Machine Science D-A-CH (IFToMM D-A-CH)*. 2018.

A. Makarow, M. Keller, C. Rösmann, T. Bertram, G. Schoppel, and I. Glowatzky. “Model Predictive Trajectory Set Control for a Proportional Directional Control Valve”. In: *IEEE Conference on Control Technology and Applications (CCTA)*. 2017, pp. 1229–1234.

M. Alsayegh, C. Rösmann, F. Hoffmann, and T. Bertram. “Model Predictive Control for Learning from Demonstration”. In: *26. Workshop Computational Intelligence*. Dortmund, 2016, pp. 31–42.

C. Wissing, J. Braun, C. Rösmann, and T. Bertram. “Entwicklung eines Regelungskonzeptes für einen Knick-Arm-Roboter für das automatisierte Laden”. In: *VDI Mechatroniktagung*. 2015, pp. 31–36.

M. Oeljeklaus, C. Rösmann, F. Hoffmann, and T. Bertram. “Trajektorienplanung mit Timed-Elastic-Bands für die proxemische Interaktion zwischen Menschen und mobilen Robotern”. In: *23. Workshop Computational Intelligence*. Dortmund, 2013, pp. 385–400.

Patent Applications

T. Bertram, A. Makarow, C. Rösmann, and M. Keller. “Regulating unit, mechatronic system, and method for regulating a mechatronic system (EN), Regelungseinheit, mechatronisches System und Verfahren zum Regeln eines mechatronischen Systems (DE)”. Patent WO 2019002587 A1, DE 10 2017 114 731 A1. 2019.

T. Bertram, C. Rösmann, F. Hoffmann, F. I. John, J. Henikl, and M. Krämer. “Large manipulator with automated boom construction (EN), Grossmanipulator mit automatisiertem Mastaufbau (DE)”. Patent WO 002018115248 A1 , DE 10 2012 206 952 A1. 2018.

W. Feiten, M. Lipkowski, C. Rösmann, and T. Wösch. “Method and device for controlling the motion of a movable unit in space (EN), Verfahren und Vorrichtung zur Steuerung der Bewegung einer beweglichen Einheit im Raum (DE)”. Patent WO 2013160195 A1, DE 10 2012 206 952 A1. 2012.

M. Lipkowski, C. Rösmann, and T. Wösch. “Method and device for controlling a movement of an autonomously driven vehicle (EN), Verfahren und Vorrichtung zum Steuern einer Bewegung eines autonom fahrenden Fahrzeuges (DE)”. Patent WO 2013178423 A1, DE 10 2012 209 110 A1. 2012.

Supervised Theses

Q. T. Bui. “Online-Trajektorienplanung auf Basis einer Initialisierung mittels Homotopieklassen für einen Roboterarm mit Hindernissen”. Master’s thesis. TU Dortmund University, 2018.

D. Feng. “Entwicklung einer Parklückendetektion für automatisierte Fahrzeuge mit Ackermann-Lenkung”. Bachelor’s thesis. TU Dortmund University, 2018.

P. Hartmann. “Rapidly-Exploring-Random-Trees als globaler Planungsansatz für automatisierte Fahrzeuge mit Ackermann-Lenkung”. Bachelor’s thesis. TU Dortmund University, 2018.

M. Klöpper. “Design of the Model Predictive Trajectory Set Control for a Wide Operation Range of a Proportional Directional Control Valve”. Master’s thesis. TU Dortmund University, 2018.

S. Kurzawe. “Modellprädikatives Ballancieren eines Stabs mit einem seriellen Roboterarm”. Master’s thesis. TU Dortmund University, 2018.

M. Mokhadri. “Model Predictive Path Integral Control for Joint Space Control of a Link-Elastic Robot Arm”. Master’s thesis. TU Dortmund University, 2018.

O. Ojekunle. “Systematische Analyse und Entwurf echtzeitfähiger modellprädiktiver Regelungen am Beispiel des inversen Pendels”. Master’s thesis. TU Dortmund University, 2018.

A. Puzicha. “Modellprädiktive Regelung eines strukturelastischen Roboterarms im Gelenkraum”. Bachelor’s thesis. TU Dortmund University, 2018.

V. Rodríguez. “Vision-based Simultaneous Localization and Mapping with Multiple Mobile Robots”. Masters’s thesis. TU Dortmund University, 2018.

C. Schmickler. “Simulationsbasierte Stabilitätsuntersuchung der modellprädikativen Trajektorienscharregelung für mechatronische Systeme”. Bachelor’s thesis. TU Dortmund University, 2018.

M. Seemann. “Systematische Analyse verschiedener Mehrgrößenregler am Beispiel eines Systems zur Mischwasservorbereitung”. Bachelor’s thesis. TU Dortmund University, 2018.

L. Zeng. “Online Trajectory Planning for Mobile Robots with Dynamic Obstacles”. Master’s thesis. TU Dortmund University, 2018.

F. Albers. “Online-Trajektorienoptimierung für mobile Roboter mit dynamischen Hindernissen und alternativen Topologien”. Master’s thesis. TU Dortmund University, 2017.

J. Bahne. “Automatisches Differenzieren für die modellprädiktive Regelung”. Bachelor’s thesis. TU Dortmund University, 2017.

P. Dorpmüller. “Analyse und Vergleich von numerischen Optimierungsalgorithmen für die Trajektorienplanung automatisierter Fahrzeuge”. Bachelor’s thesis. TU Dortmund University, 2017.

M. Golonka. “Systematische Analyse des PID- und des modellprädiktiven Reglerentwurfs am Beispiel eines mechatronischen Systems”. Bachelor’s thesis. TU Dortmund University, 2017.

E. Mares. “Radarbasierter Einparkassistent”. Master’s thesis. TU Dortmund University, 2017.

S. Neumann. “Benchmarking von Trajektorienplanern im Robot Operating System”. Bachelor’s thesis. TU Dortmund University, 2017.

C. Booms. “Navigation mobiler Roboter mit Ackermann Lenkung am Beispiel von Lego Mindstorms EV3 Plattformen”. Bachelor’s thesis. TU Dortmund University, 2016.

Q. T. Bui. “Aufbau einer Experimentalumgebung zur Lokalisation und Trajektorienregelung von LEGO Mindstorms Robotern mit einem Motion Capture System”. Bachelor’s thesis. TU Dortmund University, 2016.

S. Kurzawe. “Evaluations-Framework für die vergleichende Langzeitanalyse von Navigationsverhalten mobiler Roboter”. Bachelor’s thesis. TU Dortmund University, 2016.

O. Mogylenko. “Vergleich prädiktiver Trajektorienregelungen und -optimierungen für nicht-holonome mobile Roboter”. Bachelor’s thesis. TU Dortmund University, 2016.

C. Wolf. “Auslegung und Evaluation eines modellprädiktiven Reglers für ein mechatronisches Antriebssystem”. Bachelor’s thesis. TU Dortmund University, 2016.

A. Bussmann. “Parallele Online-Trajektorienplanung für mobile Roboter in alternativen Homotopie-Klassen”. Master’s thesis. TU Dortmund University, 2015.

S. Gillet. “Semantic Spatial Hierarchy for Mobile Robot Navigation and Topological Mapping”. Master’s thesis. TU Dortmund University, 2015.

A. Navarro. “Trajectory Planning for Non-Holonomic Mobile Robots with Timed Elastic Bands”. Master’s thesis. TU Dortmund University, 2015.

O. Rinaldo. “ROS Framework for Local Trajectory Planning with Times Elastic Bands”. Master’s thesis. TU Dortmund University, 2015.

F. Tabassum and C. Hebbeker. “Recherche und Implementierung von Konzepten zur Regelung von Portalkränen am Beispiel eines Linearpendel-Experimentalsystems”. Student research project. TU Dortmund University, 2015.

L. Tan. “Exploration Strategies for Vision Based topological Map Building”. Master’s thesis. TU Dortmund University, 2015.

M. Theile. “Monte-Carlo basierte Robustheitsanalyse für die nichtlineare modellprädiktive Regelung”. Bachelor’s thesis. TU Dortmund University, 2015.

D. Thielsch. “Ein modellprädiktives Planungs- und regelungsverfahren zur Fahrzeugführung in Notsituationen”. Master’s thesis. TU Dortmund University, 2015.

Supervised Student Project Groups

M. Klöpper, R. J. V. Guillen, E. M. Hernandez, D. Sarmient, R. T. Salam, S. A. Shanaei, and J. C. Z. Jie. “Model Predictive Control for Robotic Pole Balancing”. TU Dortmund University, 2017.

W. Ahmed, J. Dizhao, S. Lai, E. V. Mfon, H. Mo, K. K. Murthy, D. A. C. Rios, F. Tabassum, and V. F. Vallejo Rodriguez. “Human-Centered Mobile Robot Navigation”. TU Dortmund University, 2016.

N. P. Anand, M. A. Bin Azhar, M. W. Gondal, A. Ismail, Y. Puttaswamy, H. ul Moqeet Riaz, F. M. Rueda, and K. Sial. “Trajectory Planning for Mobile Manipulators”. TU Dortmund University, 2015.

A. Chauhan, H. Gomri, B. Meier, F. I. Mues, A. Navarro, O. Rinaldo, A. Shahidi, L. Tan, A. Vahidi, K. S. Govindappa, and V. K. Bangarusamy. “Computer Vision for Mobile Robot Navigation, Localization and Mapping”. TU Dortmund University, 2014.