Nonlinear Model Predictive Low-Level Control

Set-Point Stabilization with Input Move-Blocking and Time-Varying Finite Control Sets

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 TU Dortmund University

by

Artemi Makarow, M.Sc.

Dortmund, Germany

Date of submission: August 22, 2022

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: December 22, 2022

Vorwort

Meine Tätigkeit als Wissenschaftler am Lehrstuhl für Regelungssystemtechnik der Fakultät Elektrotechnik und Informationstechnik der Technischen Universität Dortmund hat mir die Anfertigung der vorliegenden Dissertation ermöglicht.
Ganz besonders bedanke ich mich bei Herrn Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram für seine exzellente Betreuung, seine Unterstützung, sein Vertrauen in mich und die zahlreichen Möglichkeiten, bei denen ich von ihm lernen durfte. Seine sehr wertvollen Impulse haben mich motiviert und angeregt, über mich hinauszuwachsen. Die mir eingeräumten wissenschaftlichen Freiheiten und die wertschätzende Atmosphäre am Lehrstuhl für Regelungssystemtechnik haben maßgeblich zum Gelingen dieser Arbeit beigetragen.
Ich bedanke mich sehr bei Herrn Univ.-Prof. Dr.-Ing. Martin Mönnigmann für die Übernahme des Korreferats, seine Zeit und sein Interesse an meiner Arbeit.
Mein Dank gebührt ebenfalls Frau Univ.-Prof. Dr.-Ing. Selma Saidi als Drittprüferin und Herrn Univ.-Prof. Dr.-Ing. Christian Rehtanz für den Vorsitz der Prüfungskommission.
Die großartigen Team-Mitglieder des Lehrstuhls für Regelungssystemtechnik haben dazu beigetragen, dass die sehr intensive Promotionszeit für mich unvergesslich in Erinnerung bleiben wird.
Ich danke Herrn apl. Prof. Dr. rer. nat. Frank Hoffmann für seine sehr wertvollen Anregungen und die Weitergabe seiner Begeisterung für die Wissenschaft und die Lehre. Ein großer Dank gilt Herrn Dr.-Ing. Daniel Schauten für seine unschätzbaren Beratungen und Ratschläge zu Problemen und Herausforderungen jeglicher Art.
In besonderer Erinnerung wird mir die Zeit mit Herrn Dr.-Ing. Christoph Rösmann im Büro 201 bleiben. Aufgrund seiner außergewöhnlichen Aufnahmefähigkeit, seiner Freude an neuen Herausforderungen und seiner offenen und zuvorkommenden Art war dieses Büro der perfekte Ort zum Träumen und Arbeiten. Ich danke Christoph vielmals für die zahlreichen wissenschaftlichen Diskussionen, seine Unterstützung und die gemeinsamen Erlebnisse auf Konferenzreisen.
Ich bedanke mich bei allen ehemaligen und aktuellen Wissenschaftlerinnen und Wissenschaftlern des Lehrstuhls für Regelungssystemtechnik für die sehr aufregende Promotionszeit, in der ich mit ihnen und von ihnen lernen durfte. Insbesondere bedanke ich mich bei Jan Braun, Prof. Dr.-Ing. Martin Keller, Dr.-Ing. Malte Oeljeklaus, Dr.-Ing. Manuel Schmidt, Freia Muster, Maximilian Krämer, Christopher Diehl und Martin Krüger.
Als wissenschaftlicher Mitarbeiter konnte ich mich stets über kompetente, schnelle und freundliche Unterstützung bei administrativen und technischen Angelegenheiten freuen. Ein besonderer Dank hierfür gebührt Nicole Czerwinski, Sascha Kersting, Halit Cicek, Gabriele Rebbe, Jürgen Limhoff und Rainer Müller-Burtscheid.
Für die anregenden Diskussionen und den wissenschaftlichen Austausch innerhalb der

Forschungskooperation mit der Bosch Rexroth AG zu Beginn meiner Promotionszeit bedanke ich mich insbesondere bei Herrn Dr.-Ing. Georg Schoppel und Herrn Ingo Glowatzky.
Die Herausforderungen der Promotionszeit haben mich auch im privaten Umfeld oft begleitet und beschäftigt. Für das Verständnis und die emotionale Unterstützung gilt mein großer Dank meiner Familie und meinen Freunden. Besonders danke ich meiner Lebensgefährtin Nadine für ihre Geduld, ihre unermüdliche Unterstützung und ihren bedingungslosen Rückhalt.

Dortmund, Dezember, 2022 Artemi Makarow

Abstract

This dissertation focuses on the development, formalization, and systematic evaluation of a novel nonlinear model predictive control (MPC) concept with derivative-free optimization. Motivated by a real industrial application, namely the position control of a directional control valve, this control concept enables straightforward implementation from scratch, robust numerical optimization with a deterministic upper computation time bound, intuitive controller design, and offers extensions to ensure recursive feasibility and asymptotic stability by design. These beneficial controller properties result from combining adaptive input domain discretization, extreme input move-blocking, and the integration with common stabilizing terminal ingredients. The adaptive discretization of the input domain is translated into time-varying finite control sets and ensures smooth and stabilizing closed-loop control. By severely reducing the degrees of freedom in control to a single degree of freedom, the exhaustive search algorithm qualifies as an ideal optimizer. Because of the exponentially increasing combinatorial complexity, the novel control concept is suitable for systems with small input dimensions, especially single-input systems, small- to mid-sized state dimensions, and simple box-constraints. Mechatronic subsystems such as electromagnetic actuators represent this special group of nonlinear systems and contribute significantly to the overall performance of complex machinery.
A major part of this dissertation addresses the step-by-step implementation and realization of the new control concept for numerical benchmark and real mechatronic systems. This dissertation investigates and elaborates on the beneficial properties of the derivative-free MPC approach and then narrows the scope of application. Since combinatorial optimization enables the straightforward inclusion of a non-smooth exact penalty function, the new control approach features a numerically robust real-time operation even when state constraint violations occur. The real-time closed-loop control performance is evaluated using the example of a directional control valve and a servomotor and shows promising results after manual controller design.
Since the common theoretical closed-loop properties of MPC do not hold with input move-blocking, this dissertation provides a new approach for general input move-blocked MPC with arbitrary blocking patterns. The main idea is to integrate input move-blocking with the framework of suboptimal MPC by defining the restrictive input parameterization as a source of suboptimality. Finally, this dissertation extends the proposed derivative-free MPC approach by stabilizing warm-starts according to the suboptimal MPC formulation. The extended horizon scheme divides the receding horizon into two parts, where only the first part of variable length is subject to extreme move-blocking. A stabilizing local controller then completes the second part of the prediction. The approach involves a tailored and straightforward combinatorial optimization algorithm that searches efficiently for suboptimal horizon partitions while always reproducing the stabilizing warm-start control sequences in the nominal setup.

Kurzfassung

Diese Dissertation befasst sich mit der Entwicklung, Formalisierung und systematischen Bewertung eines neuartigen nichtlinearen modellprädiktiven Regelungskonzepts mit ableitungsfreier Optimierung. Motiviert durch eine reale industrielle Anwendung, die Positionsregelung eines Wegeventils, ermöglicht dieses Regelungskonzept eine von Grund auf einfache Implementierung, eine robuste numerische Optimierung mit einer deterministischen oberen Rechenzeitschranke und einen intuitiven Reglerentwurf. Zudem bietet dieser Ansatz Erweiterungen, um rekursive Durchführbarkeit und asymptotische Stabilität der Regelung sicherzustellen. Diese vorteilhaften Reglereigenschaften resultieren aus der Kombination adaptiver Diskretisierung des Stellraums, restriktiver Parametrisierung der prädizierten Steuerungssequenzen (*input move-blocking*) und der Integration mit allgemeinen stabilisierenden Entwurfskomponenten. Die adaptive Diskretisierung des Stellraums wird in zeitvariante endliche Stellmengen übersetzt und ermöglicht eine glatte und stabilisierende Regelung. Durch die starke Reduktion der Freiheitsgrade in der Steuerung auf einen einzigen Freiheitsgrad stellt der vollständige Suchalgorithmus einen idealen Optimierer dar. Aufgrund der exponentiell ansteigenden kombinatorischen Komplexität ist das neuartige Regelungskonzept für Systeme mit kleinen Eingangsdimensionen, insbesondere für Eingrößensysteme, kleinen bis mittleren Zustandsdimensionen und einfachen Beschränkungen geeignet. Mechatronische Subsysteme wie beispielsweise elektromagnetische Aktuatoren repräsentieren diese spezielle Gruppe nichtlinearer Systeme und tragen wesentlich zur Gesamtleistung komplexer Maschinen bei.

Ein großer Teil dieser Dissertation behandelt die schrittweise Implementierung und Umsetzung des neuen Regelungskonzepts für numerische Referenz- und reale mechatronische Systeme. Es werden die vorteilhaften Eigenschaften des ableitungsfreien modellprädiktiven Regelungsansatzes untersucht und ausgearbeitet sowie anschließend der Anwendungsbereich der Regelung eingegrenzt. Da die kombinatorische Optimierung die einfache Einbeziehung einer nicht-glatten exakten Straffunktion ermöglicht, zeichnet sich der neue Regelungsansatz durch einen numerisch robusten Echtzeitbetrieb aus, selbst wenn Zustandsbeschränkungen verletzt werden. Die Regelungsgüte wird am Beispiel eines Wegeventils und eines Servomotors evaluiert und zeigt bereits nach dem manuellen Reglerentwurf vielversprechende Ergebnisse.

Da die üblichen theoretischen Eigenschaften der modellprädiktiven Regelung bei der Beschränkung der Freiheitsgrade in der Steuerung nicht gelten, wird in dieser Dissertation ein neuer Ansatz für die klassische modellprädiktive Regelung mit beliebigen Restriktionen der Freiheitsgrade in der Steuerung vorgestellt. Die Hauptidee besteht darin, die restriktiven Parametrisierungen der Steuerungssequenzen in den Kontext der suboptimalen modellprädiktiven Regelung zu integrieren. Die Einschränkung der Freiheitsgrade wird hier als eine Ursache für die Suboptimalität definiert. Schließlich wird in dieser Dissertation der vorgeschlagene ableitungsfreie modellprädiktive Regelungsansatz um stabilisierende Warmstarts gemäß der suboptimalen Formulierung der modellprädiktiven Regelung erweitert. Das erweiterte Horizontschema unterteilt den beschränkten und gleitenden Horizont in zwei Teile, wobei nur der erste Teil mit variabler Länge der extremen Beschränkung der Freiheitsgrade unterliegt. Ein stabilisierender lokaler Regler wird genutzt, um den zweiten Teil der Vorhersage zu vervollständigen. Der vorgeschlagene Ansatz umfasst die Entwicklung eines maßgeschneiderten und einfachen kombinatorischen Optimierungsalgorithmus, der effizient nach suboptimalen Horizontteilungen sucht und dabei stets die stabilisierenden Warmstarts in der nominellen Konfiguration reproduziert.

Contents

Nomenclature

Acronyms

ARBS	**A**mplitude Modulated **R**andom **B**inary **S**ignal
BFGS	**B**royden-**F**letcher-**G**oldfarb-**S**hanno
CLF	**C**ontrol **L**yapunov **F**unction
CPU	**C**entral **P**rocessing **U**nit
DSP	**D**igital **S**ignal **P**rocessor
EH	**E**xtended **H**orizon
ES	**E**xhaustive **S**earch
FCS-MPC	**F**inite **C**ontrol **S**et Nonlinear **M**odel **P**redictive **C**ontrol
FD	**F**ull **D**iscretization
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
GPU	**G**raphical **P**rocessing **U**nit
IPOPT	**I**nterior **P**oint **OPT**imizer
IVP	**I**nitial **V**alue **P**roblem
KKT	**K**arush-**K**uhn-**T**ucker
LHS	**L**eft-**H**and **S**ide
LMPC	**L**inear **M**odel **P**redictive **C**ontrol
LQR	**L**inear **Q**uadratic **R**egulator
LTI	**L**inear **T**ime-**I**nvariant
MB	**M**ove-**B**locking
MBMPC	Input **M**ove-**B**locked Nonlinear **M**odel **P**redictive **C**ontrol
MPC	Nonlinear **M**odel **P**redictive **C**ontrol
MPTSA	**M**odel **P**redictive **T**rajectory **S**et **A**pproach
MPTSC	**M**odel **P**redictive **T**rajectory **S**et **C**ontrol
MS	**M**ultiple **S**hooting
NLP	**N**on**L**inear **P**rogram
NRMSE	**N**ormalized **R**oot **M**ean **S**quare **E**rror
NTI	**N**onlinear **T**ime-**I**nvariant
OCP	**O**ptimal **C**ontrol **P**roblem
OSQP	**O**perator **S**plitting **Q**uadratic **P**rogram
PC	**P**ersonal **C**omputer
QP	**Q**audratic **P**rogram
RE	**R**ecursive **E**limination
RHS	**R**ight-**H**and **S**ide
RTI	**R**eal-**T**ime **I**teration

SBMPC	**S**ampling **B**ased **M**odel **P**redictive **C**ontrol
SDNMPC	**S**ampling **D**riven **N**onlinear **M**odel **P**redictive **C**ontrol
SF	**S**ingle Degree of **F**reedom
SFEMPC	**S**ingle Degree of **F**reedom **E**xtended Horizon Nonlinear **M**odel **P**redictive **C**ontrol
SFMPC	**S**ingle Degree of **F**reedom Nonlinear **M**odel **P**redictive **C**ontrol
SFVMPC	**S**ingle Degree of **F**reedom **V**ariable Horizon Nonlinear **M**odel **P**redictive **C**ontrol
SQP	**S**equential **Q**uadratic **P**rogramming
VH	**V**ariable **H**orizon

Abbreviations

Alg.	Algorithm
App.	Appendix
Asm.	Assumption
cf.	compare
Ch.	Chapter
Cor.	Corollary
Def.	Definition
e.g.	for example
Eq.	Equation
et al.	and others
Fig.	Figure
i.e.	that is
minimizer	local optimum
Off.	Offset
optimizer	optimization algorithm
Proc.	Procedure
Prop.	Proposition
Rem.	Remark
Sec.	Section
Thm.	Theorem

Roman Letters for Scalars

a_0, a_1, a_2, b_0	mathematical model parameters (control valve)
$a(t)$	acceleration
$\hat{a}(t)$	estimated acceleration
b	positive scalar for adjusting the gradient of $\tanh(\cdot)$ at zero
b_1, b_2, b_3	scaling parameters of power-law bounds (exponential stability)
c	cardinality of a finite set
c_j	cardinality of a finite set for the j-th input dimension
C_{ecp}	Coulomb friction torque (industrial plant emulator)

C_{sd}	linear spring constant (mass-spring-damper system)
$d_y(t)$	difference between measured and estimated position
D_{ecp}	damping constant (industrial plant emulator)
D_{sd}	damping constant (mass-spring-damper system)
$e_{j,i}$	dynamics discretization error for the j-th state with Δt_i
$E_{\text{dsp}}(t)$	encoder output (industrial plant emulator)
f_{s}	closed-loop control sampling rate
$F_{\text{h}}(t)$	hydraulic flow force (control valve)
$F_{\text{s}}(t)$	solenoid force (control valve)
i	running index or positive integer only
$i_{\text{h},1}(t), i_{\text{h},2}(t), i_{\text{h},3}(t)$	Hall sensors outputs (industrial plant emulator)
$i_{\text{m}}(t)$	measured solenoid current (control valve)
$i_{\text{p},1}(t), i_{\text{p},2}(t)$	motor phase currents (industrial plant emulator)
$i_{\text{ref}}(t)$	reference solenoid current (control valve)
$i_{\text{ref},1}(t)$	reference current motor 1 (industrial plant emulator)
$i_{\text{ref},2}(t)$	reference current motor 2 (industrial plant emulator)
$i_{\text{s}}(t)$	solenoid current (control valve)
I_{ecp}	total load inertia (industrial plant emulator)
j	running index or positive integer only
J, J_1, J_2	current cost function values (pseudo-code)
$\bar{J}_{\text{cl}}$	performance metric for closed-loop control
J_∞	infinite horizon costs
k	time instant (discrete-time, open-loop)
k_1, k_2	running indices (pseudo-code)
K_{ecp}	motor constant (industrial plant emulator)
l	running index or positive integer only
l_{u}	number of closed half-spaces (polyhedral input constraints)
l_{x}	number of closed half-spaces (polyhedral state constraints)
ℓ_1	symbol indicates inclusion of absolute values
L_0	number of control vectors in first shooting interval
L_1	number of control vectors in second shooting interval
m	number of input dimensions
M	number of degrees of freedom in control
M_{sd}	mass (mass-spring-damper system)
$M_1(t)$	drive torque motor 1 (industrial plant emulator)
$M_2(t)$	drive torque motor 2 (industrial plant emulator)
n	time instant (discrete-time, closed-loop)
N	prediction horizon (discrete-time)
N^*	optimal horizon length (VH)
N_1, N_2	lengths of horizon parts (EH)
N_1^0	initial first horizon length (EH)

$N_1^\dagger(\underline{x}_0), N_2^\dagger(\underline{x}_0)$	suboptimal horizon partition (EH)
$NRMSE$	normalized root mean square error
p	number of state dimensions
p_1, p_2, p_3, p_4	mathematical model parameters (industrial plant emulator)
$\tilde{p}_1, \tilde{p}_2, ..., \tilde{p}_6$	coefficients of polynomial discretization mapping
$\bar{p}(t)$	hydraulic pressure (control valve)
$P_1(t), P_2(t), P_3(t)$	motor phase voltages (industrial plant emulator)
q	running index or positive integer only
q_1, q_2, q_3	weighting parameters (state penalty)
$Q(t)$	fluid flow rate (control valve)
r_1	weighting parameter (control penalty)
r_g	weighting parameter (regularization)
$\tilde{s}_j$	j-th slack variable
$S_\mathrm{g}(t)$	general signal
t	time
$\bar{t}$	normalized time (value range $[0,1]$)
t_k	k-th time point on uniform time grid (open-loop)
t_m	median of execution time measurements
$\bar{t}_\mathrm{m}$	normalized median of execution time measurements
$\bar{t}_\mathrm{m,ref}$	reference median of execution time measurements
t_max	time duration of experiment (control valve)
t_n	n-th time point on uniform time grid (closed-loop)
t_N	prediction horizon (continuous-time)
t_run	run time (task execution time)
t_{95}	0.95-quantile of execution time measurements
T	number of shooting intervals
T_s	step time (step width of sampled data system)
u	general control
u_h	control interval center in polynomial discretization
u_j	j-th element of general control vector $\underline{u}$
u_max	maximum control
$u_{\mathrm{max},j}$	j-th element of maximum control vector $\underline{u}_\mathrm{max}$
u_min	minimum control
$u_{\mathrm{min},j}$	j-th element of minimum control vector $\underline{u}_\mathrm{min}$
$u(k)$	control value
$u_\mathrm{a}(t)$	continuous solenoid input voltage (control valve)
$\tilde{u}_\mathrm{f}(n)$	control corresponding to $\tilde{\underline{x}}_\mathrm{f}(n)$ (offset compensation)
$u_j(k)$	j-th element of control vector $\underline{u}(k)$
$u_j^\circ(n)$	j-th element of control vector $\underline{u}^\circ(n)$ (adaptive discretization)
$u_\mathrm{p}(t)$	pulse width modulated solenoid input voltage (control valve)
v_lim	velocity limit

v_{max}	maximum velocity
$v(t)$	velocity
$\hat{v}(t)$	estimated velocity
$v_{\text{a}}(t)$	angular velocity (industrial plant emulator)
$v_{\text{dsp}}(t)$	measured angular velocity (DSP)
$v_{\text{filt}}(t)$	filtered velocity (offline zero-phase filtering, control valve)
$v_{\text{ls}}(t)$	estimated velocity (polynomial online filter, control valve)
W	filter window length (polynomial online filter)
x_j	j-th element of general state vector $\underline{x}$
$x_j(k)$	j-th element of state vector $\underline{x}(k)$
y_{f}	constant reference position
y_i	i-th element of vector $\underline{y}$ (NRMSE)
$y_{\text{ref},i}$	i-th element of reference vector $\underline{y}_{\text{ref}}$ (NRMSE)
$y(t)$	output (position)
$\hat{y}(t)$	estimated position
$y_{\text{dsp}}(t)$	measured angular position (DSP)
$\tilde{y}_{\text{f}}(t)$	virtual reference position (offset compensation)
$y_{\text{ls}}(t)$	estimated position (polynomial online filter, control valve)
$y_{\text{m}}(t)$	measured position/stroke (control valve)
$y_{\text{ref}}(t)$	set-point/reference position (control valve)
$y_{\text{s}}(t)$	armature position/solenoid stroke (control valve)
$y_1(t)$	horizontal position of center of mass (planar aircraft)
$y_2(t)$	vertical position of center of mass (planar aircraft)
z_i	i-th optimization parameter

Roman Letters for Vectors

$\underline{e}(n)$	measurement error
$\underline{h}_{\text{u}}$	RHS of system of inequalities (polyhedral input constraints)
$\underline{h}_{\text{x}}$	RHS of system of inequalities (polyhedral state constraints)
$\underline{\mathbf{s}}$	sequence of shooting nodes
$\underline{s}_q$	q-th shooting node
$\underline{\tilde{s}}$	vector of slack variables
$\underline{u}$	general control vector
$\underline{\mathbf{u}}$	control sequence
$\underline{u}_{\text{f}}$	control vector corresponding to steady state vector $\underline{x}_{\text{f}}$
$\underline{u}_{\text{max}}$	maximum control vector
$\underline{u}_{\text{min}}$	minimum control vector
$\underline{\mathbf{u}}_q$	part of control sequence corresponding to q-th shooting interval
$\underline{\mathbf{u}}_{\text{ref}}$	reference control sequence for performance metric $\bar{J}_{\text{cl}}$
$\underline{u}_{\text{s}}$	control vector (SF)
$\underline{u}_{\text{s}}^0$	initial state vector sample (SF)
$\underline{\check{\mathbf{u}}}$	reduced-order control sequence (MB)

$\check{\underline{\mathbf{u}}}_i^+$	$\check{\underline{\mathbf{u}}}$ at time instant $n+1$ after i optimization iterations (MB)
$\underline{u}(k)$	control vector
$\underline{u}_{\mathrm{f}}(n)$	control vector corresponding to steady state vector $\underline{x}_{\mathrm{f}}(n)$
$\breve{\underline{u}}(k)$	control vector after coordinate transformation
$\check{\underline{u}}(k)$	control vector (MB)
$\underline{u}^{\circ}(n)$	central control vector for adaptive input domain discretization
$\underline{\mathbf{u}}^*(\underline{x}_0)$	optimal control sequence
$\underline{\mathbf{u}}^\dagger(\underline{x}_0)$	suboptimal control sequence
$\underline{u}_{\mathrm{s}}^*(\underline{x}_0)$	optimal control vector (SF)
$\underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0)$	optimal control sequence (SF)
$\underline{u}_{\mathrm{s}}^\dagger(\underline{x}_0)$	suboptimal control vector (EH)
$\tilde{\underline{\mathbf{u}}}(\underline{x}_0)$	warm-start control sequence
$\check{\underline{\mathbf{u}}}^*(\underline{x}_0)$	optimal control sequence (MB)
$\check{\underline{\mathbf{u}}}^\dagger(\underline{x}_0)$	suboptimal control sequence (MB)
$\underline{u}^*(k, \underline{x}_0)$	optimal control vector
$\underline{u}^\dagger(k, \underline{x}_0)$	suboptimal control vector
$\underline{u}^*(k, \underline{x}_0, n)$	optimal control vector (with time-varying control set)
$\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)$	optimal control vector (SF with time-varying control set)
$\underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0, n)$	optimal control sequence (SF with time-varying control set)
$\tilde{\underline{u}}(k, \underline{x}_0)$	control vector of warm-start control sequence
$\check{\underline{u}}^*(k, \underline{x}_0)$	optimal control vector (MB)
$\check{\underline{u}}^\dagger(k, \underline{x}_0)$	suboptimal control vector (MB)
$\underline{w}(n)$	modeling error
$\underline{x}$	general state vector
$\underline{\mathbf{x}}$	sequence of state vectors
$\underline{x}_{\mathrm{f}}$	steady state vector
$\underline{x}_0$	initial state vector for internal model
$\underline{x}_{\mu,0}$	initial state vector for closed-loop system
$\underline{x}^+$	state vector at time instant $k+1$ (or $n+1$)
$\underline{x}_0^+$	state vector of closed-loop system at time instant $n+1$
$\underline{x}_N^*$	last state vector of optimal open-loop state trajectory
$\underline{x}(k)$	state vector
$\underline{x}_{\mathrm{f}}(n)$	steady state vector (time-varying reference)
$\breve{\underline{x}}(k)$	state vector after coordinate transformation
$\tilde{\underline{x}}_{\mathrm{f}}(n)$	virtual steady state vector (offset compensation)
$\underline{x}_\mu(n)$	state vector of closed-loop system
$\hat{\underline{x}}_\mu(n)$	state vector of disturbed closed-loop system
$\underline{y}$	arbitrary vector (NRMSE)
$\underline{y}_l$	difference in Lagrange function between iterations l and $l+1$
$\underline{y}_{\mathrm{ref}}$	reference vector (NRMSE)
$\underline{z}$	optimization parameter vector

$\underline{\mathbf{z}}$	sequence of optimization parameter vectors
$\underline{z}^*$	optimization parameter vector at a local optimum
$\underline{z}_l$	parameter vector after l optimization iterations
$\tilde{\underline{z}}_l$	difference in primal variables between iterations l and $l+1$

Roman Letters for Matrices

$\underline{A}$	system matrix (linear discrete-time system)
$\underline{A}_\mathrm{c}$	system matrix (linear continuous-time system)
$\underline{A}_\mathrm{K}$	system matrix (linear and autonomous discrete-time system)
$\underline{B}$	input matrix (linear discrete-time system)
$\underline{B}_\mathrm{c}$	input matrix (linear continuous-time system)
$\check{\underline{B}}$	input move-blocking matrix
$\underline{C}$	output matrix (linear system)
$\underline{D}(\underline{\mathbf{z}})$	Jacobian matrix of the constraint functions evaluated at at parameter vector $\underline{\mathbf{z}}$ (with RE and MS)
$\underline{D}_\mathrm{h}(\underline{z})$	Jacobian matrix of $\underline{h}(\cdot)$ evaluated at $\underline{z}$
$\underline{G}_\mathrm{u}$	LHS of system of inequalities (polyhedral input constraints)
$\underline{G}_\mathrm{x}$	LHS of system of inequalities (polyhedral state constraints)
$\underline{H}_l$	Hessian approximation at optimization iteration l
$\underline{K}$	controller gain matrix (local state space controller)
$\underline{L}$	observer gain matrix (control valve)
$\underline{P}$	positive definite weighting matrix (terminal penalty)
$\underline{Q}$	positive definite weighting matrix (state penalty)
$\underline{Q}_\mathrm{K}$	positive definite weighting matrix (state penalty, linear system)
$\underline{R}$	positive definite weighting matrix (control penalty)

Roman Letters for Scalar-Valued Maps

$F(\cdot)$	terminal cost function
$g_i(\cdot)$	i-th inequality constraint function (NLP)
$h_i(\cdot)$	i-th equality constraint function (NLP)
$h_{\mathrm{ps},i}(\cdot)$	i-th state constraint function (polyhedral constraints)
$I(\cdot)$	function for determining time instants of shooting nodes
$J_1(\cdot)$	one-step horizon cost function
$J_N(\cdot)$	finite horizon cost function
$J_{N^*-1}(\cdot)$	cost function with horizon length N^*-1 (VH)
$\tilde{J}_N(\cdot)$	exact penalty function (for softening state constraints)
$\mathcal{L}(\cdot)$	Lagrange function (NLP)
$\ell(\cdot)$	stage cost function
$u_\mathrm{I}(\cdot)$	uniform discretization of closed interval
$V(\cdot)$	general Lyapunov function
$V_1(\cdot)$	optimal one-step horizon cost function
$V_1^\mathrm{s}(\cdot)$	optimal one-step horizon cost function (SF)

$V_N(\cdot)$	optimal finite horizon cost function
$V_N^{\mathrm{s}}(\cdot)$	optimal finite horizon cost function (SF)
$V_\infty(\cdot)$	optimal infinite horizon cost function
$\tilde{V}_N(\cdot)$	optimal exact penalty function
$\tilde{V}_N^{\mathrm{s}}(\cdot)$	optimal exact penalty function (SF)
$V_{N^*}(\cdot)$	optimal variable horizon cost function
$V_{N^*}^{\mathrm{s}}(\cdot)$	optimal finite horizon cost function (SF, VH)

Roman Letters for Vector- and Set-Valued Maps

$\underline{f}(\cdot)$	transition map (discrete-time system)
$\underline{\mathcal{F}}(\cdot)$	general terminal constraint function
$\underline{f}_{\mathrm{c}}(\cdot)$	vector field (continuous-time system)
$\underline{\tilde{g}}(\cdot)$	transition map (autonomous discrete-time system)
$\underline{\mathcal{G}}(\cdot)$	general state constraint function
$\underline{h}(\cdot)$	equality constraint function
$\underline{H}(\cdot)$	set-valued transition map (suboptimal MPC)
$\underline{h}_{\mathrm{KKT}}(\cdot)$	KKT conditions function
$\underline{h}_{\mathrm{pd}}(\cdot)$	primal-dual system function
$\underline{h}_{\mathrm{ps}}(\cdot)$	state constraint function (polyhedral constraints)
$\underline{\mathbf{u}}_{\mathrm{e}}(\cdot)$	control parameterization based on EH formulation

Greek Letters for Scalars

$\bar{\alpha}_l$	search step width after l Newton steps
β	penalty scaling parameter (exact penalty function)
$\bar{\beta}$	scaling parameter of barrier penalty function
$\bar{\beta}_l$	$\bar{\beta}$ after l optimization iterations
$\chi_j(t)$	j-th element of state vector $\underline{\chi}(t)$ (continuous-time system)
$\hat{\chi}_j(t)$	j-th element of state vector $\underline{\hat{\chi}}(t)$ (sampled data system)
$\hat{\chi}_{\mu,j}(t)$	j-th element of state vector $\underline{\hat{\chi}}_{\mu}(t)$ (sampled data system)
$\chi_{\mathrm{ref},j}(t)$	j-th state generated with reference integration scheme
δ	radius of terminal ball
Δt_i	sampling time with $\Delta t_i = 2^{-12+i}$ s
Δt_{s}	sampling time
Δu_j	discretization step width for the j-th input dimension
ϵ	radius to describe some neighborhood of the origin
$\tilde{\epsilon}$	coupling parameter (planar aircraft)
γ	number of oversampling steps (sampled data system)
ι	exponent in polynomial input domain discretization
λ	auxiliary optimization variable (MB)
$\lambda^{\dagger}$	suboptimal choice of auxiliary variable (MB)
λ_i	λ after i optimization iterations (MB)
λ_i^{+}	λ at time instant $n+1$ and after i optimization iterations (MB)

$\tilde{\lambda}_i$	i-th element of Lagrange multiplier vector $\underline{\tilde{\lambda}}$
$\tilde{\mu}_i$	i-th element of Lagrange multiplier vector $\underline{\tilde{\mu}}$
ν	exponent of power-law bound (exponential stability)
ω	number of optimization parameters
$\tilde{\omega}$	number of optimization parameters containing slack variables
$\varphi_{\mathrm{v}}(t)$	output (position, Duffing oscillator)
π	positive constant (bound for terminal costs or circle constant)
π^*	optimal cost bound for terminal level set
ρ	scaling parameter for infinite costs of the linearized system
$\sigma_j(t)$	j-th element of control vector $\underline{\sigma}(t)$ (continuous-time system)
$\sigma_{\mu,j}(t)$	j-th element of control vector $\underline{\sigma}_\mu(t)$ (sampled data system)
τ	integration variable
$\theta(t)$	roll angle (planar aircraft)
$\zeta_1, \zeta_2, \zeta_3$	coefficients of polynomial filter (control valve)
$\zeta_1^*, \zeta_2^*, \zeta_3^*$	optimal coefficients of polynomial filter (control valve)

Greek Letters for Vectors

$\underline{\chi}(t)$	state vector (continuous-time system)
$\hat{\underline{\chi}}(t)$	open-loop state vector (sampled data system)
$\hat{\underline{\chi}}_\mu(t)$	closed-loop state vector (sampled data system)
$\Delta\underline{z}$	search direction for optimization parameter vector
$\Delta\underline{z}^*$	optimal search direction for parameter vector (SQP)
$\Delta\underline{\tilde{\lambda}}, \Delta\underline{\tilde{\mu}}$	search directions for Lagrange multipliers
$\Delta\underline{\tilde{\lambda}}^*, \Delta\underline{\tilde{\mu}}^*$	optimal search directions for Lagrange multipliers (SQP)
$\underline{\tilde{\lambda}}$	vector of Lagrange multipliers (equality constraints)
$\underline{\tilde{\lambda}}_l$	$\underline{\tilde{\lambda}}$ after l optimization iterations (equality constraints)
$\underline{\tilde{\lambda}}^*$	Lagrange multipliers at a local optimum (equality constraints)
$\underline{\tilde{\mu}}$	vector of Lagrange multipliers (inequality constraints)
$\underline{\tilde{\mu}}_l$	$\underline{\tilde{\mu}}$ after l optimization iterations (inequality constraints)
$\underline{\tilde{\mu}}^*$	Lagrange multipliers at a local optimum (inequality constraints)
$\underline{\sigma}(t)$	control vector (continuous-time system)
$\underline{\sigma}_\mu(t)$	closed-loop control vector (sampled data system)
$\underline{\vartheta}$	extended state containing state vector and warm-start
$\underline{\vartheta}^+$	extended state at the next time instant $n+1$
$\underline{\Xi}_1, \underline{\Xi}_2, \underline{\Xi}_3, \underline{\Xi}_4$	intermediate steps of Runge-Kutta integration rule

Greek Letters for Scalar-Valued Maps

$\alpha(\cdot)$	comparison function with $\alpha(\cdot) \in \mathcal{K}$
$\alpha_1(\cdot)$	comparison function with $\alpha_1(\cdot) \in \mathcal{K}_\infty$
$\alpha_2(\cdot)$	comparison function with $\alpha_2(\cdot) \in \mathcal{K}_\infty$
$\alpha_3(\cdot)$	comparison function with $\alpha_3(\cdot) \in \mathcal{K}$
$\alpha_{\mathrm{f}}(\cdot)$	comparison function with $\alpha_{\mathrm{f}}(\cdot) \in \mathcal{K}_\infty$

$\alpha_\ell(\cdot)$	comparison function with $\alpha_\ell(\cdot) \in \mathcal{K}_\infty$
$\eta(\cdot)$	discretization function
$\mu_j(\cdot)$	j-th component of implicit control law
$\psi(\cdot)$	nonlinear cost function (NLP)

Greek Letters for Vector-Valued Maps

$\underline{\Gamma}(\cdot)$	saturation function for control vectors
$\underline{\kappa}(\cdot)$	local control law
$\tilde{\underline{\kappa}}(\cdot)$	saturated local control law
$\underline{\mu}(\cdot)$	implicit control law
$\underline{\Omega}(\cdot)$	operator to generate an admissible warm-start control sequence
$\underline{\Omega}_\kappa(\cdot)$	generate control sequence based on the local control law $\underline{\kappa}(\cdot)$
$\underline{\Omega}_{\tilde{\kappa}}(\cdot)$	generate control sequence based on the local control law $\tilde{\underline{\kappa}}(\cdot)$
$\underline{\Omega}_{\mathrm{st}}(\cdot)$	shift-truncate operator for control sequences (VH)
$\underline{\Omega}_{\mathrm{sta}}(\cdot)$	shift-truncate-append operator for control sequences
$\underline{\varphi}(\cdot)$	open-loop state trajectory (discrete-time system)
$\underline{\varphi}_{\mathrm{g}}(\cdot)$	closed-loop state trajectory of the system $\underline{x}^+ = \tilde{\underline{g}}(\underline{x})$
$\underline{\varphi}_\mu(\cdot)$	closed-loop state trajectory (discrete-time system)
$\underline{\phi}(\cdot)$	open-loop state trajectory (continuous-time system)
$\underline{\phi}_\Sigma(\cdot)$	open-loop state trajectory (sampled data system)
$\underline{\Theta}_N(\cdot)$	control parameterization based on move-blocking
$\underline{\Theta}_N^{\mathrm{s}}(\cdot)$	control parameterization based on extreme move-blocking
$\underline{\Xi}(\cdot)$	integration kernel of iterative IVP solution method

Sets

$\mathbb{A}(n)$	finite control set (adaptive input domain discretization)
$\mathbb{A}_j(n)$	finite set of scalar controls for the j-th input dimension (adaptive input domain discretization)
$\tilde{\mathbb{A}}(\underline{x}_0, n)$	extended finite control set (including local control sample)
$\mathcal{A}(\underline{z})$	set of indices of all active inequality constraints at point $\underline{z}$
$\mathcal{B}_\delta$	closed terminal ball with radius $\delta > 0$
$\mathbb{D}$	finite control set (uniform input domain discretization)
$\mathbb{D}_j$	finite set of scalar controls for the j-th input dimension (uniform input domain discretization)
$\mathcal{E}, \tilde{\mathcal{E}}$	finite sets of indices (equality constraints)
$\emptyset$	empty set
$\mathcal{I}$	finite set of indices (inequality constraints)
$\mathcal{K}$	set of comparison functions (function space)
$\mathcal{K}_\infty$	set of unbounded comparison functions (function space)
$\mathbb{N}$	set of all positive integers excluding zero
$\mathcal{N}$	compact interval of integer horizon lengths
$\mathbb{N}_0$	set of all positive integers including zero

$\mathbb{N}_{\geq 2}$	set of all positive integers starting from 2
$\mathbb{N}^{[a,d]}$	set of all positive integers on the compcat interval $[a, d]$
$\mathbb{O}_{\geq 3}$	set of all odd numbers starting from 3
$\bar{\mathbb{P}}$	placeholder set
$\mathbb{P}(\underset{\sim}{x}_0, n)$	placeholder set (for finite control sets)
$\mathbb{R}$	set of real numbers
$\mathbb{R}_0^+$	set of positive real numbers including zero
$\mathbb{R}^+$	set of positive real numbers excluding zero
$\mathcal{S}$	set of admissible tuples (pseudo-code)
$\mathcal{S}_l$	stabilizable set
$\mathcal{S}_l^{\mathrm{s}}$	stabilizable set (SF)
$\mathcal{T}$	terminal set (experimental stability analysis)
$\mathrm{lev}_\pi F$	terminal level set
U	input set
$\mathbb{U}$	input constraint set
$\mathcal{U}_N(\underset{\sim}{x}_0)$	set of all admissible control sequences
$\mathcal{U}_N^{\mathrm{s}}(\underset{\sim}{x}_0)$	set of all admissible control sequences (SF)
$\tilde{\mathcal{U}}_N(\underset{\sim}{x}_0)$	set of all admissible warm-start control sequences
$\bar{\mathcal{U}}_N(\underset{\sim}{x}_0)$	set of all admissible warm-start control sequences (EH)
$\mathcal{U}_N^{\dagger}(\underset{\sim}{x}_0, \underline{\tilde{\mathbf{u}}}(\underset{\sim}{x}_0))$	set of all admissible control sequences, which result in lower or equal costs than the warm-start $\underline{\tilde{\mathbf{u}}}(\underset{\sim}{x}_0)$
$\mathcal{U}_N^{\mathrm{e}}(\underset{\sim}{x}_0, N_1)$	set of all admissible control sequences with extreme move-blocking up to horizon length N_1 (EH)
X	state set
$\mathbb{X}$	state constraint set
$\mathbb{X}_{\mathrm{f}}$	terminal set
$\mathbb{X}_{\tilde{\kappa}}(N)$	subset of the region of attraction of the saturated local controller
$\mathcal{X}$	positive invariant set for system $\underset{\sim}{x}^+ = \tilde{\underset{\sim}{g}}(\underset{\sim}{x})$
$\mathcal{X}_1$	feasible set for one-step horizon
$\mathcal{X}_1^{\mathrm{s}}$	feasible set for one-step horizon (SF)
$\mathcal{X}_N$	feasible set
$\mathcal{X}_N^{\mathrm{s}}$	feasible set (SF)
$\bar{\mathcal{X}}_M$	feasible set (MB)
$\bar{\mathcal{X}}_N^{\mathrm{e}}$	finite union of feasible sets for different horizon lengths (EH)
$\mathcal{X}_N^{\mathrm{e}}(N_1)$	feasible set for a given first horizon length N_1 (EH)
$\bar{\mathcal{X}}_{\mathrm{s}}$	finite union of feasible sets for different horizon lengths (SF, VH)
$\mathbb{Z}$	set of all integers including zero
$\mathbb{Z}^{[a,d]}$	set of all integers on the compact interval $[a, d]$
$\mathcal{Z}_N$	feasible set of extended states
$\bar{\mathcal{Z}}_N$	feasible set of extended states (EH)

Mathematical Notation

$\lvert a \rvert$	absolute value of a
$\lvert \mathbb{D} \rvert$	cardinality of finite set $\mathbb{D}$
$\approx$	approximately equal
$a := b$	define a to be equal to b
$\mathrm{diag}(a, b, c)$	square matrix with diagonal (a, b, c), other elements are zero
$f : A \mapsto B$	function $f(\cdot)$ maps set A into set B
$x \mapsto f(x)$	function $f(\cdot)$ maps x to $f(x)$
$\dot{f}(t_0)$	time derivative of function $f(\cdot)$ evaluated at time t_0
$\nabla f(\underline{z}_0)$	gradient of scalar function $f(\cdot)$ evaluated at $\underline{z}_0$
$\nabla_{\underline{z}} f(\underline{z}_0, \underline{x}_0)$	gradient of scalar function $f(\cdot)$ with respect to vector $\underline{z}$ and evaluated at $(\underline{z}_0, \underline{x}_0)$
$\nabla^2_{\underline{z}\underline{z}} f(\underline{z}_0, \underline{x}_0)$	Hessian matrix of scalar function $f(\cdot)$ with respect to vector $\underline{z}$ and evaluated at $(\underline{z}_0, \underline{x}_0)$
$\mathrm{D}\underline{f}(\underline{z}_0)$	Jacobian matrix of vector-valued function $\underline{f}(\cdot)$ evaluated at $\underline{z}_0$
$f(x) = \mathcal{O}(h(x))$	$f(x) \leq a\, h(x)$ for all $x \geq \bar{x}$ with $\bar{x}, a \in \mathbb{R}^+$ and $f, h : \mathbb{R} \mapsto \mathbb{R}_0^+$ (Big O notation)
$\partial f(z, x)/\partial z\rvert_{(z_0, x_0)}$	partial derivative of function $f(\cdot)$ with respect to z and evaluated at (z_0, x_0)
$f_{\{1,2\}}(t)$	short notation for $f_1(t)$ and $f_2(t)$
$\forall$	for all
$x \in [a, b]$	bounded and closed (compact) interval with $a \leq x \leq b$
$x \in (a, b)$	bounded and open interval with $a < x < b$
$x \in [a, b)$	bounded, left-closed, right-open interval with $a \leq x < b$
$\in$	is an element of
$\otimes$	Kronecker product
$\max \mathbb{U}$	element-wise maximum of compact set $\mathbb{U}$
$\min \mathbb{U}$	element-wise minimum of compact set $\mathbb{U}$
$\lVert \cdot \rVert$	Euclidean norm
$\lVert \underline{x} \rVert^2_{\underline{Q}}$	weighted norm with $\lVert \underline{x} \rVert^2_{\underline{Q}} := \underline{x}^\mathsf{T} \underline{Q}\, \underline{x}$
$\neq$	not equal
$a = \sum_{i=1}^{3} i$	$a = 1 + 2 + 3$ (summation notation)
$a = \prod_{i=1}^{3} i$	$a = 1 \cdot 2 \cdot 3$ (product notation)
$A = \bigcup_{i=1}^{3} B_1$	$A = B_1 \cup B_2 \cup B_3$ (union of sets)
$\ll$	significantly smaller
$\underline{a} \preceq \underline{b}$	elementwise inequality between vector $\underline{a}$ and vector $\underline{b}$
$A \subseteq B$	set A is a subset of set B
$A \subset B$	set A is a strict subset of set B
$A \backslash B$	elements of set A that are not elements of set B (set difference)
sup	supremum
min	minimum

$\arg\min$	argument of the minimum
T	transpose operator
$\underline{A}^{\#}$	Moore-Penrose inverse of matrix $\underline{A}$
$\log(\cdot)$	natural logarithm function
$\sin(\cdot), \cos(\cdot)$	trigonometric functions
$\tanh(\cdot)$	hyperbolic tangent function
$\underline{I}_p$	identity matrix with dimensions $p \times p$
$\underline{1}_p$	unit vector with dimensions $p \times 1$
$\underline{0}_p$	zero vector with dimensions $p \times 1$
$\underline{0}$	zero vector (dimensions result from the context)

1 Introduction

1.1. Motivation

Most technical systems, such as automated vehicles, robots, construction and manufacturing machines, result from system integration and the interaction of various subsystems from different domains. Here, a considerable part is dedicated to mechatronic subsystems. For example, electric motors contribute to the complex operation or even motion of technical apparatus, such as industrial robots, by exerting force or torque on internal mechanical components. To reduce the overall complexity of central processing and control units responsible for specific functionalities such as vehicle maneuvering, these units usually do not directly compute the control input for all individual actuators. Instead, the central control units access the low-level controllers, which are then individually charged with minimizing the distance of the controlled variable (internal system states) to some commanded reference (set-point). Hence, the operation or motion of the overall system strongly depends on the closed-loop control performance of the individual subsystems. Figure 1.1 shows a generalized block diagram of a mechatronic subsystem. Note that the system in Figure 1.1 satisfies characteristic aspects of the definition of a mechatronic system according to [Har+96]. It integrates multiple information processing units such as sensors and controllers, as well as an electromagnetic actuator driving a mechanical system to perform a complex task. The inner control unit in Figure 1.1 represents a single current control loop or, for example, again a control cascade comprising an inner current and an outer velocity control loop.

A real industrial application motivates the development of a novel model predictive low-level control concept in this dissertation. Control valves are applied to route and control oil flow in hydraulic systems, for example, in lifting devices, which need to exert high force densities to complete their tasks (e.g., [Ewa+03]). However, the inner structure of a control valve coincides with the structure introduced in Figure 1.1. A cascaded control concept, containing an outer position controller and inner current controller, together with the power electronics, drive a solenoid that changes the position of a piston inside the valve body. The oil flow through the valve depends, in particular, on the position of the piston and the resulting cross-sectional openings. For example, with the 4WRPEH 6 directional control valve of the Bosch Rexroth com-

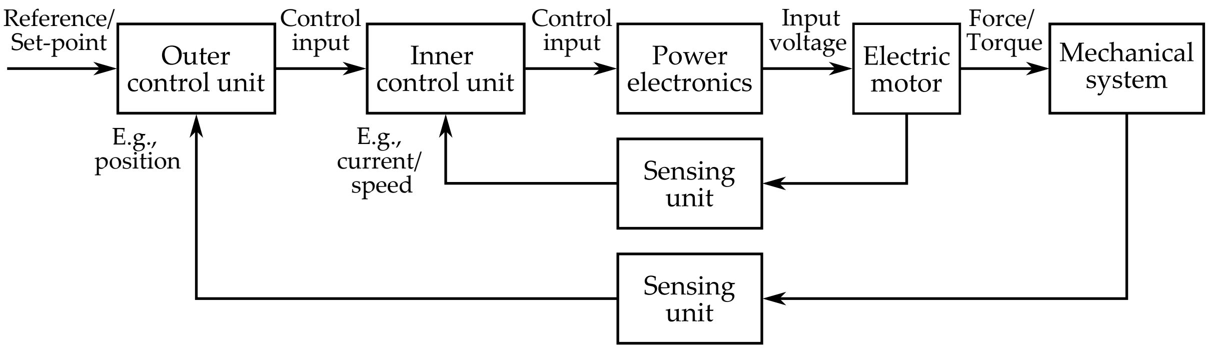


Figure 1.1.: Generalized structure of mechatronic systems used in this dissertation. The objective is to replace the native low-level controllers by a derivative-free MPC approach that allows straightforward implementation from scratch and intuitive controller design.

pany [Bos10], a tailored nonlinear proportional-integral position controller with a high number of coupled parameters ensures a high closed-loop control performance for different operating ranges (e.g., [Ott04; Kre+07]). However, the controller design requires a high level of human expertise and experience. Automated controller design based on hardware-in-the-loop simulations and global optimization techniques minimizes the time required to find optimal controller parameters (e.g., [Nic+01; Kre+07]). A subsequent fine tuning of closed-loop control performance requires again profound human expertise. Sliding mode control as an alternative (model-free) control concept reveals promising results and reduces the parameter complexity significantly [Kri+15]. However, the controller design again rests upon global optimization or is subject to self-tuning [Kri+16b], see also [Kri18] for more details. The question arises whether a suitable control scheme can be developed to replace the complex low-level position controller while allowing straightforward implementation, manual intuitive controller design, and explicit compliance with input and state constraints.

Nonlinear model predictive control (MPC) is a powerful control concept suitable for cross-domain applications. Conventional MPC solves an optimal control problem (OCP) in each sampling interval while explicitly considering user-defined objective functions, future system behavior, and nonlinear input and state constraints (e.g., [ML99; May+00]). Usually, the implicit control law applies the first optimal control vector to the plant in every closed-loop sampling interval. Important closed-loop properties such as recursive feasibility and asymptotic stability are well understood and can be established intuitively for nonlinear systems based on Lyapunov's direct method (e.g., [May+00; Raw+20]). Because of significant advances in numerical optimization, MPC is emerging as an alternative control approach for mechatronic systems with fast dynamics and continuously differentiable functions (e.g., [Ver+18; And+18; Eng+19]). MPC is based on a dynamics model of the real system and therefore integrates explicit and systematically acquired system knowledge. Including as much system knowledge as possible reduces parameter complexity on the controller side and thus facilitates controller design. At first glance, only the choice of the weights of the individual cost terms remains for adjusting the control performance. However, specifically for nonlinear systems, the control engineer needs to gain deeper insight into direct transcription methods, constrained nonlinear optimization, and especially their efficient implementation under real-time constraints. Numerical optimization, in particular, yields additional free parameters such as numerical step sizes and tolerances, which again complicate controller design. In addition, for a special class of nonlinear systems, MPC offers more degrees of freedom in control than required to achieve high closed-loop control performance. In particular, systems with small to mid-sized input and state dimensions and simple box-constraints are suitable for minimizing the degrees of freedom in control and thus the computational effort (e.g., [Mak+18d]). Input move-blocking, for example, merges consecutive control vectors on the prediction horizon such that they have the same values along each input dimension (e.g., [TJ02; Cag+07]). When replacing the position controller without directly manipulating the switch positions of the individual transistors of the power electronics, as with finite control set MPC (FCS-MPC, e.g., [Rod+13]), the underlying control loops usually have a single or at least only a few inputs.

The model predictive trajectory set approach (MPTSA) was proposed in the context of developing an emergency steering assist [Kel+15; Kel17]. This driving assistance system plans repeatedly a finite number of trajectories with a constant steering wheel angle based on a vehicle dynamics model and guides the human driver in critical situations by superimposing steering torques to avoid collisions. By severely reducing the degrees of freedom in control to a single degree, the optimal control and state trajectories can be determined by simple exhaustive search (ES). An adaptive discretization of control variables (quantization in the technical sense), such as the steering wheel angle, further enables smooth vehicle guidance. However, this MPC approach can also be used for a general class of systems beyond the context of vehicle guidance. Numerical analyses and evaluations of the generalized model predictive trajectory set control (MPTSC) in [Kel17; Mak+17] and [Mak+18d] show promising results for linear and nonlinear dynamical single-input systems, respectively. However, the initial formulation of MPTSC does not include stabilizing ingredients such as a terminal cost function or terminal set constraints. Thus, the closed-loop control stability with MPTSC rests upon extensive numerical simulations and experiments. For some linear dynamical benchmark system, Figure 1.2 visualizes the open-loop prediction of the first state with $\hat{\chi}_{\mu,1} : \mathbb{R}_0^+ \mapsto \mathbb{R}$ on top of the time evolution of the closed-loop system with MPTSC. Here, the single degree of freedom in control, in conjunction with the input domain discretization, creates the characteristic trajectory sets at each closed-loop time instant. Note that Figure 1.2 only shows every 50th open-loop prediction. Though MPTSC greatly reduces the implementation effort compared to conventional MPC by eliminating smooth optimization, the control performance might suffer from the restrictive input parameterization. Since common theoretical closed-loop stability properties of MPC do not hold in case of input move-blocking (e.g., [Cag+07]), the horizon length turns out to be a free controller parameter.

However, (extreme) input move-blocking, adaptive input domain discretization, and combinatorial optimization are promising ingredients for developing an alternative and derivative-free model predictive control concept for fast mechatronic systems. In particular, the single degree of freedom in control is of central interest, since it leads to a combinatorial complexity that can be solved by a simple search algorithm with a small and deterministic upper runtime bound.

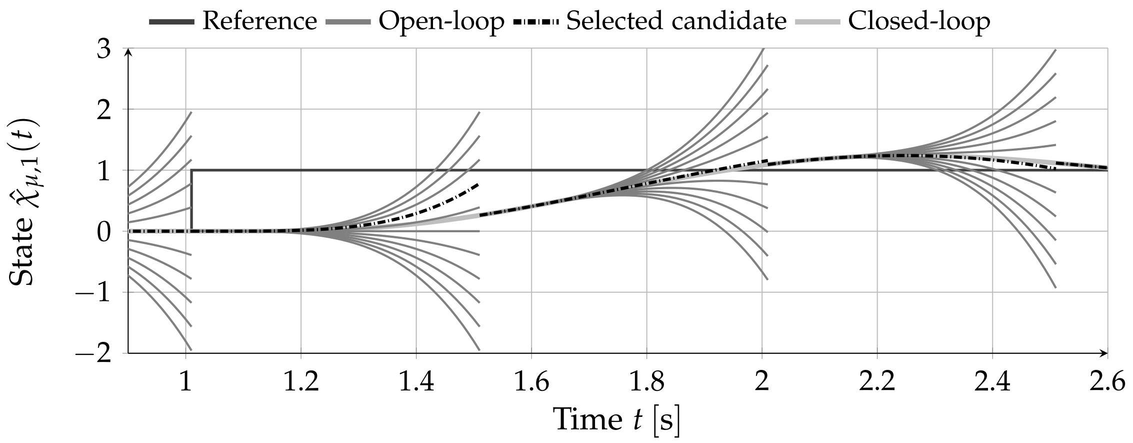


Figure 1.2.: Open- and closed-loop control of a linear dynamical benchmark system with MPTSC. With a single degree of freedom in control and a suitable input quantization, an exhaustive search qualifies for determining the optimal control and state trajectories. The open-loop prediction is visualized at every 50th closed-loop time step. Compare also Figure 4 in [Kel+15].

1.2. Contribution and Outline

This dissertation addresses the development, formalization, and evaluation of a model predictive low-level control concept for nonlinear systems with small input dimensions, preferably a single input, and small to mid-sized state dimensions. This low-level control concept satisfies the following partially conflicting requirements:

- Straightforward implementation without dependencies on external optimization libraries,
- Low computational effort with a deterministic upper bound,
- Intuitive controller design and adjustment of closed-loop control performance during real-time operation,
- If requested, ensuring theoretical closed-loop stability properties: Recursive feasibility and asymptotic stability.

Motivated by the promising results of MPTSC, the model predictive low-level controllers in this dissertation adopt and integrate the basic ingredients of MPTSC, namely a single degree of freedom in control, a suitable adaptive input domain discretization, and the exhaustive search strategy. Though MPTSC already satisfies the first two requirements, it is still an open question whether these two requirements are also fulfilled when stabilizing closed-loop properties have to be enforced by an extended OCP formulation. The last two demands first require a precise formalization and classification of the proposed ingredients into the existing literature of MPC. Novel extensions to basic MPTSC are proposed to relax the restrictive input parameterization and guarantee stabilizing closed-loop properties. Then, this dissertation contributes to the systematic development of a controller design guide considering closed-loop control performance, recursive feasibility, and asymptotic stability. Extensive numerical analyses and evaluations narrow the scope of the novel low-level control concept. In addition, experimental investigations demonstrate the applicability and highlight the advantages of the proposed low-level control scheme for industrial valve and servo motor control. Because of its close connection to conventional MPC, the novel basic control approach is referred to as single degree of freedom MPC (SFMPC) hereafter. The following outline provides an overview of the structure of this dissertation and highlights the detailed contribution:

Chapter 2: The second chapter first summarizes the state-of-the-art in MPC with a focus on set-point stabilization. The basic ingredients of SFMPC are each considered separately in the context of MPC, and the related literature is reviewed and summarized.

Chapter 3: This chapter defines the basic nomenclature and summarizes the key fundamentals and findings in optimal control, conventional MPC, and numerical discretization that are highly relevant for this dissertation. The sampled-data formulation with the zero-order hold strategy connects the continuous-time nature of mechatronic systems with the established discrete-time MPC framework.

Chapter 4: By integrating common stabilizing terminal ingredients into the formulation of basic SFMPC, stabilizing closed-loop properties can already be shown for three special system configurations. From the systematic analysis of the properties of SFMPC with uncountable input constraint sets (for smooth optimization), the definition and temporal evolution of finite control sets (for combinatorial optimization) are derived to obtain the same stabilizing closed-loop properties and a similar closed-loop control performance. Since SFMPC with an infinite horizon cost function and an uncountable control set can reproduce, under mild assumptions, the linear-quadratic regulator (LQR), this chapter examines and evaluates the discretization (quantization in the technical sense) error with finite control sets independently of other effects such as input move-blocking.

Chapter 5: This chapter investigates the applicability of basic SFMPC to constrained nonlinear systems. The first part of this chapter compares closed-loop control performances, computational and implementation efforts of SFMPC, conventional MPC, and the LQR. With SFMPC, a short to mid horizon length combined with a final cost function, representing the cost of an infinite horizon for some linearized system, proves to be a suitable configuration to ensure recursive feasibility and high closed-loop control performances for common nonlinear benchmark systems. The second part of this chapter introduces state-constraint softening with SFMPC. Since SFMPC with finite control sets rests upon exhaustive search, processing non-smooth exact penalty functions does not lead to any additional effort. However, for practical applications, the proposed softening approach ensures reliable closed-loop control operation even if short-term state constraint violations occur because of perturbations and model mismatch.

Chapter 6: A step-by-step realization of real-time capable MPC and SFMPC is presented in this chapter using two real mechatronic subsystems with different characteristics. The implementation from scratch includes model identification, controller design, and compensation techniques. Since model mismatch in modelbased state space controllers often results in closed-loop control offset, this chapter addresses the development of a tailored offset compensation technique that is suitable for real-time closed-loop control of directional control valves. In this chapter, the softened state constraint handling emerges as the key feature of SFMPC that allows a straightforward implementation, intuitive controller design, and smooth closed-loop operation in the presence of state constraints. According to the literature review, this chapter presents, for the first time, a model predictive low-level controller for a directional control valve that covers a large operating range and is real-time capable at a closed-loop control frequency of 10 kHz.

Chapter 7: The detailed literature review reveals that asymptotic stability with input move-blocked nonlinear model predictive control (MBMPC) is still an open problem for nonlinear systems. There is no design recommendation for MBMPC in the literature that aims at guaranteeing both recursive feasibility and asymptotic stability. This chapter integrates input move-blocking in the theoretical framework of suboptimal MPC by explicitly defining input move-blocking as a source of suboptimality. This novel formulation closes the gap and ensures asymptotic stability for MBMPC with arbitrary blocking patterns. Most importantly, this chapter introduces the main ingredients for stabilizing SFMPC in the next chapter.

Chapter 8: This chapter relaxes the extreme input move-blocking of SFMPC and introduces a variable horizon formulation that ensures stabilizing closed-loop properties even by design. To ensure asymptotic set-point stabilization and to enlarge the feasible set and thus the region of attraction of SFMPC, an extended SFMPC formulation is introduced to seamlessly integrate with the suboptimal MPC framework. Single degree of freedom extended horizon MPC (SFEMPC) inherits the idea of a receding horizon, however, the control trajectory is only subject to the extreme input move-blocking on a first part of the horizon. A (stabilizing) local control law completes the second part of the horizon, while the horizon partitioning is subject to mixed-integer optimization. Applying time-varying finite control sets enables the design of a practical and derivative-free optimization algorithm that searches efficiently for suboptimal horizon partitions.

Chapter 9: This dissertation concludes with a summary of the main results and findings and provides an outlook on possible further research directions.

Appendix A: The first part of the appendix provides supplemental information on numerical discretization and optimization. In addition, this appendix chapter includes details on the implementation of a generic and proprietary MPC software framework.

Appendix B: This appendix chapter includes supplemental results on stability analysis, experimental system identification, and further real-time closed-loop control experiments.

2 Related Work in Model Predictive Control

This chapter gives an overview about related work in the field of MPC. Section 2.4 has been published in a similar form in [Mak+22].

2.1. Set-Point Stabilization for Nonlinear Systems

Though the basic idea of modern MPC was already recorded, for example, in [LM67, Ch. 7], the conventional MPC formulation for constrained dynamical and nonlinear systems, as it is understood today, was established in the late 1990s and the early 2000s. Since the computing power was very limited at the beginning of the research activities in this field, initially a great deal of research has been done in MPC for linear systems. Therefore, the first industrial realizations include MPC based on impulse response models (e.g., [Ric+77; Ric+78]) and step response models in the context of dynamic matrix control (e.g., [CR80]). A detailed summary of the development up to and also beyond the 1990s can be found in [Gar+89; QB03].

Nowadays, conventional MPC for set-point stabilization and tracking usually implements a receding horizon and integrates a general nonlinear state space model, a continuous stage cost function, nonlinear input and state constraints, a special terminal cost function, and special terminal constraints (e.g., [GP17; Raw+20]). Common MPC approaches differ particularly in the last two ingredients, which are used to prove stabilizing closed-loop control properties.

In [KG88] and [MM90], the authors establish stabilizing closed-loop properties by introducing a terminal equality constraint while omitting the terminal cost function. Since a terminal equality constraint might be too restrictive, resulting in a severe restriction of the region of attraction of MPC, the authors of [CA98] propose the combination of a terminal penalty function and terminal inequality constraints that enforce the last predicted state vector to be a member of a control invariant terminal set. Chen and Allgöwer [CA98] show that, under mild assumptions, a quadratic terminal cost function bounds the infinite horizon costs that result from applying a stabilizing local control law to the nonlinear system from above if the last predicted state is inside some terminal cost level set. The terminal ingredients are derived from linear system theory after linearizing the nonlinear system at the origin. Though this approach relaxes the computational effort of MPC significantly, the proposed stabilizing terminal ingredients require an offline determination of the terminal level set prior to control. A more generalized framework for nonlinear systems with input constraints is presented in [Fon01]. The key feature of this generalized framework is the possibility to derive terminal ingredients even if the linearized system is not stabilizable. Mayne et al. [May+00] give an extensive survey on stabilizing MPC and standardize the formulation of stabilizing terminal ingredients. Refer also to [May13] for a detailed discussion on the necessity of terminal ingredients. The authors of the work in [Köh+20] propose an extensive framework that enables the offline determination of parameterized terminal ingredients independently of the specific set-point. In principle, the work in [DeN+98] can be placed in the same category, namely to MPC with stabilizing terminal conditions. De Nicolao et al. [DeN+98] implement a non-quadratic cost function that represents the infinite horizon costs when applying some local stabilizing control law to the nonlinear system. Here, the value of the terminal cost function is estimated based on the recursive application of the local controller for a sufficiently large number of steps. However, instead of introducing constraints on the terminal set, the authors enforce compliance with some terminal stability region of the local controller by weighting predicted trajectories not ending inside the local stability region at infinity. Magni et al. [Mag+01] follow up on this theoretical work, dividing the prediction horizon into two intervals and proposing a finite step terminal cost function. The first interval is subject to optimization, while the second interval is completed by recursively applying some local control law to the nonlinear system. Regardless, terminal set constraints enforce the last predicted state vector to be a member of the control invariant terminal set. The authors of [Mag+01] further show that there exists a positive number of control steps with respect to the local control law, which must be applied to the nonlinear system inside the terminal set, such that the finite step terminal cost function represents a valid control Lyapunov function. The last step is a key ingredient of stabilizing terminal conditions.

There are many contributions and research directions on MPC that either dispense with terminal constraints or even stabilizing terminal components entirely. Since the theoretical derivations in this dissertation follow, in particular, the derivations of suboptimal MPC according to [All+17; Raw+20], which rely on common stabilizing terminal ingredients, only a brief overview of MPC without terminal ingredients is given below. Parisini and Zoppoli [PZ95], for example, do not impose hard terminal set constraints. However, by choosing a sufficiently large horizon length and weighting of the terminal cost function, compliance with a certain terminal cost level set is ensured implicitly. However, the authors do not propose a design guide for free parameters. The authors of [AB95] and also [JH05] show that there is at least a theoretical minimal horizon length that ensures closed-loop stability properties, even in the absence of a terminal cost function and terminal constraints. Jadbabaie and Hauser [JH05] extend these theoretical results for input constrained nonlinear systems for the case when the terminal cost function represents an upper bound on the infinite horizon costs. Limon et al. [Lim+06] also drop the terminal set constraints and investigate the impact on some weighting of the terminal cost function on the region of attraction. Usually, those contributions that omit terminal constraints entirely assume exponential controllability with respect to the stage cost function as, for example, in [Gri+05; Tun+06; GR08;

Grü09]. The survey in [Grü12] summarizes and further extends the existing literature on MPC without stabilizing terminal conditions. Since MPC schemes without terminal ingredients cannot inherit the invariance property from some terminal set, recursive feasibility has to be examined separately (e.g., [Boc+14]). Köhler and Allgöwer [KA21] show stabilizing closed-loop properties of MPC with a finite step terminal cost function, similar to [Mag+01], assuming exponential controllability. However, since the authors do not impose terminal set constraints and the terminal cost function results from applying the local control law for a finite number of steps, this new approach closes the gap between MPC approaches with and without terminal ingredients. Similar to [Mag+01], this strategy allows to reduce the horizon on which the control variables are subject to optimization.

2.2. Suboptimal Framework

Suboptimal MPC is highly relevant for practical applications since it explicitly considers the situation that the optimizer cannot find the global optimum. This situation may arise with a non-convex OCP formulation or with limited computation times. Stability then mainly relies on manually generated stabilizing warm-starts [Sco+99; Pan+11; All+17; Raw+20]. While the initial work in [Sco+99] either chooses from a terminal equality constraint or a dual-mode scheme similar to [MM93], the contributions in [Pan+11; All+17; Raw+20] rely on common stabilizing terminal ingredients. The stability analysis in conventional MPC, as described for example in [May+00], is based on the fact that the optimal cost function represents a suitable Lyapunov function. However, the requirement of global optimality might be too restrictive for practical applications. Fortunately, the authors of [Pan+11; All+17; Raw+20] show that, under mild assumptions, the general (non-optimal) cost function is a suitable Lyapunov function candidate when introducing the evolution of a difference inclusion, including an extended state that consists of the initial state vector and a warm-start control sequence. As soon as an admissible control sequence exists, suboptimal MPC can completely dispense with optimization while nevertheless offering inherent robustness margins for systems with continuous- and discrete-valued inputs [RR17a]. Nominal stabilization and robustness properties result from designing admissible warm-start control sequences for each next closed-loop control step. The main ingredient for generating these warm-starts is the existence of some stabilizing local control law [Pan+11; All+17; Raw+20]. In contrast to the work in [GK10], which investigates stabilizing properties of suboptimal MPC for continuous-time systems with input and without terminal constraints, the suboptimal MPC framework with stabilizing terminal ingredients according to [Pan+11; All+17; Raw+20] does not require a lower bound on the number of optimization iterations for closed-loop stabilization.

2.3. Efficient Realizations with Smooth Optimization

As mentioned above, MPC solves an OCP in each closed-loop control interval. For this purpose, most MPC realizations first convert the OCP into a nonlinear program (NLP), an ordinary parameter optimization problem that is subject to nonlinear equality and inequality constraints (e.g., [GP17; Raw+20]). For solving an NLP, the literature offers a large collection of established numerical optimization algorithms (e.g., [NW06]). In general, a distinction must be made between continuous- and discrete-time OCPs during conversion. Though this dissertation relies on the discrete-time MPC formulation, sequential and simultaneous conversion methods originate from the continuous-time domain (e.g., [Bin+01]). In the continuous-time domain, the control and state trajectories have infinite degrees of freedom. Therefore, direct transcription methods aim at transforming an infinite dimensional OCP into a finite dimensional parameter optimization problem (e.g., [Kra85; Bin+01; Bet98; Raw+20]). Direct single shooting discretizes the control trajectory on a uniform time grid and assumes a typically piecewise constant control parameterization resulting in a finite dimensional parameter optimization vector. The state trajectory results from solving the corresponding initial value problem (IVP) over the entire horizon (shooting interval). Since a variation of an element of the parameter vector requests resolving the IVP, starting from the time point the individual parameters corresponds to, optimization algorithms might exhibit poor convergence properties when applying this sequential approach (e.g., [Bin+01]). In direct multiple shooting, the state trajectory is composed of multiple shooting intervals, with additional deflection constraints ensuring continuous state trajectories after convergence is reached. On each shooting interval, the optimizer requests the solution of an IVP starting at an individual shooting node [BP84]. Though this horizon partitioning results in higher dimensional problem matrices of the NLP, the optimization algorithms usually show faster convergence with this simultaneous approach [BP84] (see also [Bet98]). In addition, matrices such as the Jacobian of constraints have sparse patterns, which can be exploited efficiently as shown in [Lei+03]. The term simultaneous indicates that both the control and the state trajectory are subject to discretization and optimization. Multiple shooting turns into a full discretization when each shooting interval has a length of one step. In this full discretization case, the one-step integration kernel for the nonlinear system dynamics can be optionally included into the NLP formulation. Direct collocation is also a member of the direct transcription method and allows to parameterize the control and state trajectories with basis functions such as polynomials [Raw+20, Sec. 8.5]. This approach, however, is usually related to the continuous-time domain. For example, the time optimal MPC framework with variable discretization presented in [Rös19] investigates direct collocation. In the discrete-time domain, the control and state trajectories are of finite dimension by design, at least if the input and state dimensions of the nonlinear systems under consideration are finite. The conversion of a discrete-time OCP into an NLP is mainly a question of how to embed the solution to the nonlinear system dynamics. Here, the conversion procedure follows the basic ideas of the sequential and simultaneous approaches, only excluding the input and state parameterization (e.g., [GP17, Sec. 12.1]).

Proper warm-starts contribute to the stabilization of steady states and to the reduction of optimization times. However, warm starting is not only restricted to the primal optimization variables, such as in suboptimal MPC. Diehl et al. [Die+02] introduce the initial value embedding technique in the context of sequential-quadratic-programming (SQP). Since with multiple shooting the shooting nodes are subject to optimization, the linearization of the NLP at the next closed-loop time instant can be initialized with the linearization at the current time instant. Linearization of the NLP is a major processing step of constrained optimization and is required to formulate the first-order necessary Karush-Kuhn-Tucker (KKT) conditions (e.g., [NW06]). With SQP, the Newton-KKT system, representing a Newton iteration on the KKT conditions, can be recast into a quadratic program (QP) comprising the first- and second-order NLP derivative information (Hessian of the Lagrangian, Jacobian of the constraints, gradient of the cost function (e.g., [NW06, Ch. 18]). If the measured initial states of two consecutive closed-loop time steps only differ slightly, a single SQP iteration suffices to satisfy the constraints again and to improve control performance [Die+05b]. The real-time-iteration (RTI) scheme finally builds on this idea and divides the single SQP iteration into a preparation and short feedback phase, distributing computation over two consecutive closed-loop steps [Die+05b]. The value embedding technique is especially suitable for constant references, otherwise the gradient of the cost function cannot be warm-started from the previous closed-loop time instant, refer to the discussion in [GP17, Sec. 12.5]. Independent of the value embedding technique, Diehl et al. [Die+05a] combine the RTI scheme with a shifting strategy of the previous solution and prove that MPC with the RTI scheme and a terminal equality constraint renders the origin asymptotically stable. For completeness, it should be noted that interior point methods can also be warm started properly. By the NLP sensitivity according to [Fia83], there exists a local minimizer in some neighborhood of the current solution if the initial state vector of two consecutive closed-loop steps do not differ strongly. If, in addition, the change between the initial state vectors does not impose a change in the active set of the inequality constraints, the KKT system factorization can be reused to minimize computation time [ZB09]. This dissertation does not include warm-starting techniques related to optimizer components, such as the cost gradient, because of changing steady states and deterministic upper computing time bounds of exhaustive search. The computation time after switching the steady state mainly determines the real-time capability of the low-level control concepts presented in this dissertation. However, some discussions in this dissertation include the RTI scheme. Verschueren et al. [Ver+16] improve the Hessian approximation of the SQP algorithm used in [Die+02] by exploiting convexity in the NLP formulation. Zanelli et al. [Zan+17] eliminate constraints on the rear part of the horizon by introducing logarithmic barrier cost functions. With this tightening approach, the authors can remove the corresponding optimization variables in the solution phase of the underlying QP of the RTI scheme by applying Riccati sweeps. In [WB10], the authors exploit the structure of quadratic programs emerging in MPC. In addition, by combining warm-starting techniques with an early terminated interior-point method, the authors enable fast computation.

First- and second-order derivative information can either be derived from sparse finite differences or automatic differentiation (e.g., [NW06, Ch. 8]). Automatic differentiation decomposes functions into individual components and then determines analytic directional derivatives. When NLP dimensions vary during runtime, a hypergraph representation is well-suited to enable efficient calculation of derivatives based on sparse finite differences without the need to implement graph coloring techniques [Rös+18a] (see also [Küm+11]). Varying NLP dimensions occur, for example, when implementing variable horizon formulations as with time-optimal MPC [Rös+21b]. The established open source MPC frameworks ACADO [Hou+10] and the successor ACADOS [Ver+18] rely on the method of SQP. These MPC implementations integrate the automatic differentiation framework CasADi [And+18] and thus build on the exact first- and second-order derivatives of the underlying NLPs. Condensing reduces QP dimensions by using the recursion of linear system dynamics in matrix form, thus eliminating state vectors from the optimization vector [Jer+11]. With multiple shooting and SQP, condensing is used to exploit the sparsity of the underlying QPs by eliminating the state vectors from the linearized continuity constraints. Thus, condensing strategies transform the sparse but large dimensional QP into a dense but small QP. After solving the dense QP, the result has to be transformed back into the sparse formulation (e.g., [BP84; Kou+15b; Fri+16]). Another possibility for reducing computation time is the integration of sparse solvers such as IPOPT [WB06] and OSQP [Ste+20], which directly process sparsity/structure patterns.

The open source MPC framework GRAMPC [KG14; Eng+19] provides short computation times by implementing an augmented Lagrangian method in combination with a projected gradient approach. In general, (primal-dual) first-order methods are especially suitable for MPC for embedded systems since gradient-based approaches offer low implementation complexity. To deal with state constraints, first-order methods usually include Lagrangian dual problems (e.g., [Kuf+14; Gis14; Kou+15a; NK15]). Refer also to the surveys on embedded optimization in MPC for more details [Fer+17; Fin+18]. An extensive comparison on embedded optimization with common methods is given in [Kuf+15].

In linear MPC (LMPC), the determination of the implicit control law is confined to the solution of a single convex QP. With linear system dynamics, the application of the batch approach transforms the general full degree of freedom OCP into a single QP by successive substitution of linear system dynamics (e.g., [Bor+17]). To further reduce the optimization effort of LMPC systematically, tracking the cost descent property of the closed-loop Lyapunov function allows to remove constraints that are inactive at the next closed-loop time instant and thus to reduce the dimension of the QP (e.g., [Jos+15; Jos+17]). For a recent extension of the strategy for removing constraints from [Jos+17] to nonlinear MPC, refer to [DM21]. A major advantage of LMPC is that the implicit control law can be pre-determined offline. The state space can be partitioned by multi-parametric programming into convex polyhedrons describing a combination of active constraints. Within a single polyhedral region, the control law is an explicit affine function of the initial state vector. Therefore, at runtime, only the region to which the current initial state vector belongs to needs to be determined. Then, the optimal control vector follows by evaluating a single algebraic equation [Bem+00; AB09]. Depending on the number of polyhedrons and the chosen representation, such as binary (e.g., [Tøn+03]) or multi-way trees [MK11], explicit LMPC can reduce the computational effort significantly. Another considerable increase in efficiency in polyhedron exploration can be achieved by pre-processing a strictly convex support function over all polyhedrons [Hol+20]. However, the partitioning process of the state space in explicit LMPC depends on the individual cost weighting parameters and thus complicates online controller design.

2.4. Set-Point Stabilization with Input Move-Blocking

Naive input move-blocking reduces the degrees of freedom in control to a requested degree by applying a user-defined but fixed blocking pattern (e.g., [TJ02; Mac02]). For example, MPTSC in [Mak+18d] implements extreme and naive input move-blocking and reduces the degrees of freedom in control to a single degree. With input move-blocking as a special fixed input trajectory parameterization, the optimizer cannot recover the previous but shifted solution to ensure at least recursive feasibility. Some of the first papers considering input move-blocking include [Ric85; Lee+95; TJ02]. For a visualization of naive input move-blocking, refer to Figure 2.1. This example shows uniform input move-blocking for a discrete-time single-input system with simple input box-constraints and the control trajectory $u : \mathbb{N}_0 \mapsto \mathbb{R}$ with $u_{\min}, u_{\max} \in \mathbb{R}$. As the closed-loop system evolves by repeatedly applying the first element of the (sub)optimal control trajectory, the optimizer cannot recover previous switching points during prediction because of a lack of degrees of freedom in control. For the same reason, the optimizer in this example cannot append a (stabilizing) terminal control at the end of the prediction horizon. Notice that with the conventional implicit MPC control law, i.e., applying the first control vector to the plant, only the open-loop control performance is subject to input move-blocking.

This section covers contributions focusing on theoretical closed-loop properties in the presence of input move-blocking. Cagienard et al. [Cag+07] encounter the problem of missing stabilizing closed-loop guarantees with MBMPC by introducing a time-varying blocking scheme in combination with offset input move-blocking. For linear time-invariant (LTI) systems, the LQR can generate a base control sequence satisfying stabilizing terminal conditions. The authors of [Cag+07] then implement offset move-blocking to improve the base sequence, while applying a time-varying shifting strategy for the blocking pattern to preserve previous control interventions. Because of the LQR base control sequence, closed-loop properties mainly hold for LTI systems. In the context of MBMPC with variable horizon, Shekhar and Maciejowski [SM12] also apply time-dependent blocking matrices to ensure recursive feasibility and finite completion times to pre-defined terminal sets. The contributions in [GI07; Gon+09] focus on strong feasibility (see [Ker00] for the definition) issues with MBMPC. Gondhalekar and Imura [GI07] introduce the definition of controlled invariant feasibility to address the problem that, when input move-blocking is involved, strong feasibility cannot be derived from stabilizing terminal conditions. To ensure that the next OCP is feasible after applying the conventional implicit control law, the definition of controlled invariant feasibility enforces the first predicted state vector to be again a member of the controlled invariant feasible set. As the authors in [GI07] note, the determination of the controlled invariant feasibility set for nonlinear systems is a challenging task. The authors of [Lon+11] ensure stabilizing closed-loop properties with MBMPC by considering two different uniform time grids. Here, the step size on the second time grid is an integer multiple of the step size of the first time grid. However, input move-blocking only applies on the first time grid. The combination of a sampled-data system formulation with piecewise constant inputs, a uniform time grid, and a uniform blocking pattern allows to convert an originally move-blocked control

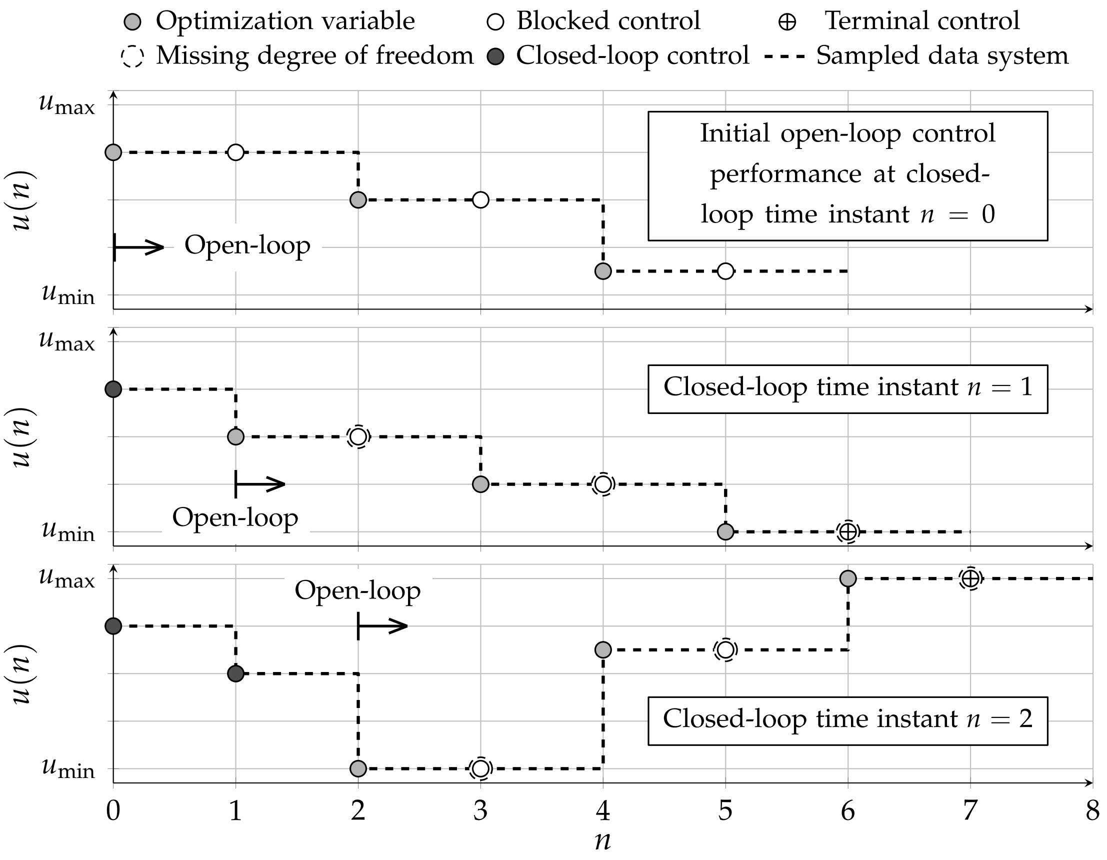


Figure 2.1.: Basic principle of model predictive control with naive input move-blocking using a discrete-time single-input system and a uniform blocking pattern with three degrees of freedom in control, a horizon length of six, and input box-constraints. Top to bottom: Evolution of the closed-loop control trajectory and the corresponding open-loop control performances. The first element of the (sub)optimal control trajectory is applied for closed-loop control. The dashed graphs indicate how the discrete-time control trajectories can be translated into a continuous-time representation by applying a piecewise constant input parameterization.

sequence into an unblocked version by increasing the step widths to the blocking interval lengths. The sampled-data formulation then offers a new discrete-time system for which stabilizing terminal conditions ensure theoretical closed-loop properties. Finally, Longo et al. [Lon+11] propose a parallelizable evaluation of differently shifted move-blocked control sequences with respect to the first time grid.

Chen et al. [Che+20] focus on merging efficiently input move-blocking with multiple shooting in the framework of the RTI scheme. Instead of adding blocking constraints after the condensing step, the authors of [Che+20] introduce a tailored condensing algorithm that exploits the reduced degrees of freedom in control. Despite of the efficient integration into the SQP method, the naive move-blocking strategy does not ensure stabilizing closed-loop properties. Gonzales and Rossiter [GR20] propose an admissible shifting strategy of move-blocking patterns and also integrate input move-blocking into the RTI scheme. In contrast to [Cag+07], the shifting strategy in [GR20] preserves the dimensions of the blocking matrix by altering the blocking pattern as the closed-loop system evolves. This altering strategy of blocking patterns represents the required prerequisite to ensure recursive feasibility with MBMPC. As the authors of [GR20] outline, stabilizing closed-loop properties might follow from introducing a terminal equality constraint. However, as mentioned above, a terminal equality constraint might be too restrictive for practical applications.

The authors of [OW14] and [SM15] show that the shifted and truncated control sequence of the previous closed-loop time instant can be embedded into the current OCP formulation by introducing auxiliary optimization parameters. Similar to [Cag+07], the previous solution serves as the base control sequence at the current closed-loop time instant. Shekhar and Manzie [SM15] show that, without shifting the blocking pattern, the optimizer can improve the base sequence by seeking a suitable move-blocked offset control sequence. Otherwise, the optimizer resorts to the previous solution and appends the local control law as a terminal control step. With this embedding strategy, the authors of [SM15] prove recursive feasibility with MBMPC and raise the topic of closed-loop asymptotic stability, however, do not prove the existence of a suitable Lyapunov function candidate. The question arises whether the blocking of the input within the terminal set is important for determining a suitable Lyapunov function. Son et al. [Son+21] extend this embedding strategy by an interpolation step between the previous solution and an LQR base sequence. This interpolation step aims at improving optimality issues and thus the closed-loop control performance, especially for LTI systems. Here, the optimizer can switch to the pure LQR base sequence and therefore disable move-blocking inside the terminal set (similar to [Cag+07]). However, the applicability to nonlinear systems is limited because of the LQR base sequence.

The literature review in this section indicates that closed-loop asymptotic stability for nonlinear discrete-time systems is still an open problem (see Table A.1 for a tabular summary). Since this section focuses on the set-point stabilization with online MBMPC with a receding horizon for discrete-time systems, it does not include contributions that split the optimization into an offline and an online part as, for example, in [TJ02; GA14], investigate non-uniform time discretization as, for example, in [YB16], investigate alternative parameterization as, for example, in [RW08], and implement a shrinking horizon formulation [Far+20].

2.5. Single Degree of Freedom in Control

The following part of the literature review excludes trajectory planning approaches based on motion primitives. These motion primitives are deemed feasible for mobile robot kinematics when internal quantities such as the robot's velocity and steering angle are kept constant or at the limit values (e.g., [LaV06]). However, these motion planning approaches do not directly operate on the input domain. In contrast, one of the established local motion planners in mobile robotics, namely the dynamic window approach, discretizes the space of translational and rotational velocities (input of kinematic model) and keeps them constant while predicting and evaluating the robot's trajectories. The dynamic window reduces the combinatorial complexity by estimating velocities that can be reached with the robot's maximum acceleration within a pre-defined time interval [Fox+97]. This approach explicitly considers dynamic

constraints and is similar to MPTSC. However, MPTSC covers a more general class of nonlinear systems and provides an adaptive sampling of the input domain. To handle more complex collision avoidance scenarios, the authors of [Hom+18] extend the basic MPTSA and introduce an additional degree of freedom in control. Since the switching point on the horizon is also subject to adaptive sampling, the combinatorial complexity increases exponentially. In [Hom+19], MPTSC is implemented as a trajectory following controller, thus having the ability to react locally to new obstacles.
The survey on first industrial MPC realizations until the 2000s in [QB03] records that single degree of freedom in control (therein referred to as a single move) was already used in multi-variable predictive control with impulse response models and nonlinear input-output models. However, as mentioned earlier, this dissertation focuses on conventional MPC with general nonlinear state space models and classifies the single degree of freedom in control as an extreme but conventional input move-blocking.
The PhD thesis [Ham95] is closely related to basic MPTSC and also involves a hydraulic application. Here, the control task is the positioning of a linear hydraulic actuator controlled by a serial servo valve, with the position sensor mounted on the hydraulic linear actuator. Instead of integrating optimal control based on indirect methods into the MPC framework, a single stage search procedure is introduced for reducing optimization time significantly. Recall that indirect methods require the solution of a two-point boundary problem. From today's perspective, this single stage search represents unconstrained MPC for nonlinear systems relying on single shooting, extreme input move-blocking, and exact penalty functions for implicitly adhering to state constraints. Because of the given hardware specification, the control input signal to the valve is subject to quantization with 480 equidistant distributed levels. Author S. P. Hampson [Ham95] motivates the implementation of the golden search algorithm, which seeks local optima by reducing the computational effort of exhaustive search over all quantization levels. Though the numerical simulations and real-time experiments show promising results, closed-loop stability analysis is missing. The major differences of MPTSC in [Kel17; Mak+18d] compared to the single stage search procedure in [Ham95] are the adaptive discretization of the input domain and the explicit consideration of input and state constraints. Therefore, MPTSC also applies to systems with non-quantized control inputs. The development of SFMPC is further dedicated to closed-loop stability analysis. The authors in [BC12] realize MPC with a single degree of freedom in control for a hybrid pneumatic-electric actuator controlled by on/off valves. Here, the combination of fixed discrete-valued inputs with a single degree of freedom enables the efficient application of exhaustive search.

2.6. Input Sampling and Finite Control Sets

Input domain discretization in MPC can either be considered as a property of the optimization algorithm or directly included in the formulation of the OCP. The first case is referred to sampling based MPC, while the latter case is known as finite control set MPC, MPC with quantized inputs, or in general as MPC with discrete-valued inputs.

Sampling based model predictive control (SBMPC) has its origin in the navigation and motion planning of mobile robots [Dun+08; Dun+10]. However, as the authors show, it can solve generic mixed-integer OCPs with variable horizon. In contrast to common sample-based motion planners, SBMPC explores the state space by sampling the input domain. Using an implicit state grid, only those input samples are used for exploration that provide some diversity in the state space. SBMPC thus dynamically builds a directed graph. Finally, an admissible heuristic is required to realize a goal-directed optimization, which is not trivial to find for arbitrary nonlinear systems. As the authors in [Dun+10] mention, the implementation is based on extensive experience of the developer with graph search approaches and requires a special graph search library. Sampling-driven suboptimal nonlinear model predictive control (SDNMPC) [BL17] combines sampling-based optimization with suboptimal MPC according to [Sco+99; Pan+11; All+17]. The main idea behind SDNMPC is to improve the already stabilizing warm-start by systematically sampling the input domain. In each time interval of the open-loop prediction, multiple samples are generated either uniformly deterministically or randomly and only for the current input element. The optimization strategy behind SDNMPC starts at the end or beginning of the warm-started control sequence and proceeds sequentially to the first or last element, updating the warm-start as soon as a better solution is available. Though the Greedy algorithm does not guarantee to find the optimal solution within a single closed-loop time interval, it can improve the warm-start. Otherwise, the manually generated warm-start represents a stabilizing fallback solution according to [Sco+99; Pan+11; All+17]. In the dissertation [Bob17], author R. V. Bobiti shows that SDNMPC can even converge to the optimal solution with increasing closed-loop iterations. The more samples SDNMPC generates at each closed-loop time instant, the faster the algorithm converges to the optimal solution. However, the very first warm-start results from a derivative based MPC implementation or originates from an “oracle” [BL17]. The major advantage of SDNMPC is its derivative-free and strongly parallelizable implementation. In the context of automatic landing of a fixed-wing aircraft, the authors of [Joo+11] discretize and quantize the control trajectories with respect to time and amplitude, respectively. The resulting high combinatorial complexity is addressed using parallel computing on field-programmable gate arrays (FPGAs). Then, in the sense of exhaustive search, the first control vector of the best control trajectory is used for closed-loop control. Stabilizing properties of the closed-loop system are outlined, resulting from simply buffering previous shifted solutions, augmented by a local control law. However, due to fixed quantization, an improvement of the buffered control sequences cannot be guaranteed, such that these cached solutions serve as a pure fallback level.

MPC algorithms that directly operate on graphical processing units (GPUs) mostly solve complex control tasks. The model predictive path integral control in [Wil+16] refines stochastic trajectory optimization for scaled autonomous rally vehicles. In the context of the control of semi-active suspension systems, the authors in [Rat+18] propose a parameterized MPC scheme that discretizes the control input with respect to time and amplitude. This MPC scheme is then integrated into an algorithm that efficiently operates on GPUs [Rat+19]. In [HK17], the authors parameterize the control trajectory and implement a real-time capable evolutionary optimization algorithm that

is highly parallelizable on GPUs. In contrast, SFMPC aims at encountering the combinatorial complexity by a sparse and time-varying input domain sampling. However, the resulting finite control sets are directly integrated into the OCP formulation.
There are many contributions in the literature dealing with finite control set MPC (FCS-MPC), as summarized by the review/survey papers [Goo+10; Rod+13; KG20]. FCS-MPC explicitly exploits the discrete nature of the switching states of power electronics. Multi-level converters, such as those used in induction motors, have a finite number of discrete switching states. As the authors discuss in [KG20], most work dealing with FCS-MPC considers a horizon length of one, since the combinatorial complexity grows exponentially with the horizon. However, in [GQ14], the authors realize multi-step FCS-MPC with the help of a customized branch-and-bound algorithm. In [Ste+17], the authors combine a short horizon with a terminal cost function that approximates the infinite horizon costs based on approximate dynamic programming according to [Wan+14]. FCS-MPC improves control performance by completely replacing conventional control structures with a predictive control approach. For example, in the context of linear electromagnetic actuators, the authors in [Has+17] introduce a cascaded control structure with an outer proportional-integral position controller and an inner velocity controller based on FCS-MPC (in simulation).
The publications [Pic+03; AQ11; AQ13] present a more general investigation on MPC with finite control sets. These papers deal with an in-depth analysis of closed-loop stability properties with systems with quantized inputs, resulting in a discrete input alphabet. The contribution [Pic+03] analyzes stabilization of discrete-time LTI systems with uniformly quantized inputs and focuses on the construction of invariant sets. In [AQ11], the authors treat the quantization error of the input as a disturbance and apply robust control analysis. The work in [AQ13] focuses, inter alia, on determining a quantized local control law for LTI systems that ensures the existence of an ultimately bounded invariant set containing the origin.
Rawlings and Risbeck [RR17a] show that common optimal and suboptimal MPC with stabilizing terminal ingredients cover both systems with only continuous- as well as partially discrete-valued inputs since the derivations on closed-loop stability do not require the control set to have an interior.
As already mentioned, SFMPC does not focus on systems with quantized or discrete-valued inputs in general. Rather, its development aims at replacing existing low-level controllers with continuous-valued control signals by the novel model predictive low-level control concept.

3 Fundamentals

The fundamentals and the notation mainly rely on the textbooks [GP17] and [Raw+20]. Parts of the following description have been used in a similar form in [Mak+22].

3.1. Discrete-Time Optimal Control Problem

The discrete-time formulation enables a concise notation and a straightforward derivation of important closed-loop properties with MPC. However, to connect the discrete-time domain with the continuous-time behavior of mechatronic systems, a sampled data formulation is used in Section 3.4. The following difference equation describes a discrete-time, nonlinear, and time-invariant system with the control trajectory $\underline{u} : \mathbb{N}_0 \mapsto U$ and the state trajectory $\underline{x} : \mathbb{N}_0 \mapsto X$:

$$\underline{x}^+ := \underline{x}(k+1) = \underline{f}\big(\underline{x}(k), \underline{u}(k)\big), \underline{x}(0) := \underline{x}_0. \tag{3.1.1}$$

Here, $X := \mathbb{R}^p$ and $U := \mathbb{R}^m$ are Euclidean spaces with the dimensions $p \in \mathbb{N}$ and $m \in \mathbb{N}$, respectively. Equation (3.1.1) determines the successor state vector $\underline{x}(k+1)$ at the next time instant $k+1$ and uses the state vector $\underline{x}_0 \in X$ for initialization.

Assumption 3.1.1: *Continuity of transition map* $\big($[Raw+20, Asm. 2.2]$\big)$. The transition map $\underline{f} : X \times U \mapsto X$ is continuous. For some steady state $(\underline{x}_\mathrm{f}, \underline{u}_\mathrm{f})$, the transition map satisfies $\underline{f}(\underline{x}_\mathrm{f}, \underline{u}_\mathrm{f}) = \underline{x}_\mathrm{f}$.

The state sequence $\underline{\mathbf{x}} := \big(\underline{x}(0), \underline{x}(1), ..., \underline{x}(N)\big) \in X^{N+1}$ results from controlling the nonlinear system (3.1.1) by the input sequence $\underline{\mathbf{u}} := \big(\underline{u}(0), \underline{u}(1), ..., \underline{u}(N-1)\big) \in U^N$. Here, $N \in \mathbb{N}$ denotes the prediction horizon. Note that the sequence formulation reshapes the individual vectors separated by commas into a single row vector. Hence, separation by a comma has a different interpretation in the multi-dimensional space than in the scalar case with $\underline{x}(k) = \big(x_1(k), x_2(k), ..., x_p(k)\big)^\mathsf{T}$ and $\underline{u}(k) = \big(u_1(k), u_2(k), ..., u_m(k)\big)^\mathsf{T}$. In the case of a single-input system, the index is obsolete such that $u(k) := u_1(k)$ applies. The function $\underline{\varphi} : \mathbb{N}_0 \times X \times U^N \mapsto X$ describes the open-loop state trajectory at different time points, providing information about the initial state vector and the applied control sequence:

$$\underline{x}(k) := \underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}}) = \begin{cases} \underline{x}_0 & \text{if } k = 0, \\ \underline{f}\big(\underline{\varphi}(k-1, \underline{x}_0, \underline{\mathbf{u}}), \underline{u}(k-1)\big) & \text{otherwise.} \end{cases} \tag{3.1.2}$$

Since the recursive function $\underline{\varphi}(\cdot)$ is a finite composition of the continuous transition map (3.1.1), continuity of the map $(\underline{x}, \underline{\mathbf{u}}) \mapsto \underline{\varphi}(k, \underline{x}, \underline{\mathbf{u}})$ directly follows from Assumption 3.1.1 [Raw+20, Prop. 2.1]. This dissertation considers the following composite finite horizon cost function:

$$J_N(\underline{x}_0, \underline{\mathbf{u}}) := \sum_{k=0}^{N-1} \ell\big(\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}}), \underline{u}(k)\big) + F\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{u}})\big). \tag{3.1.3}$$

Assumption 3.1.2: *Continuity of cost functions* ([Raw+20, Asm. 2.2]). The stage cost function $\ell : X \times U \mapsto \mathbb{R}_0^+$ and the terminal cost function $F : X \mapsto \mathbb{R}_0^+$ are continuous. The individual functions satisfy $\ell(\underline{0}_p, \underline{0}_m) = 0$ and $F(\underline{0}_p) = 0$.

The second part in Assumption 3.1.2 is important for controlling (open-loop) the nonlinear system (3.1.1) to the origin (or steady state in general). Again, since the cost function (3.1.2) is a finite composition of continuous functions, the continuity of $J_N(\cdot)$ on the set $X \times U^N$ follows from Assumptions 3.1.1 and 3.1.2 [Raw+20, Prop. 2.4 (a)]. The Euclidean norm $\|\cdot\|$ and the weighted norm $\|\underline{x}\|_{\underline{Q}}^2 := \underline{x}^\mathsf{T} \underline{Q} \underline{x}$, with a positive definite weighting matrix $\underline{Q} \in \mathbb{R}^{p \times p}$, represent the distance metrics in this dissertation. Assumption 3.1.2 is satisfied by implementing quadratic cost functions defined by:

$$\ell\big(\underline{\breve{x}}(k), \underline{\breve{u}}(k)\big) := \|\underline{\breve{x}}(k)\|_{\underline{Q}}^2 + \|\underline{\breve{u}}(k)\|_{\underline{R}}^2,\ \forall k \in \mathbb{N}^{[0,N-1]},\ F\big(\underline{\breve{x}}(N)\big) := \|\underline{\breve{x}}(N)\|_{\underline{P}}^2. \tag{3.1.4}$$

Here, $\underline{Q} \in \mathbb{R}^{p \times p}$, $\underline{R} \in \mathbb{R}^{m \times m}$, and $\underline{P} \in \mathbb{R}^{p \times p}$ are positive definite weighting matrices. Applying coordinate transformation yields $\underline{\breve{x}}(k) := \underline{x}(k) - \underline{x}_\mathrm{f}$ and $\underline{\breve{u}}(k) := \underline{u}(k) - \underline{u}_\mathrm{f}$. If not otherwise stated, the reference vectors are defined as $\underline{x}_\mathrm{f} := \underline{0}_p$ and $\underline{u}_\mathrm{f} := \underline{0}_m$.
In the following, the sets $\mathbb{U} \subset U$, $\mathbb{X} \subseteq X$, and $\mathbb{X}_\mathrm{f} \subseteq \mathbb{X}$ represent the input, state, and terminal constraint sets, respectively.

Assumption 3.1.3: *State and input constraint sets* ([RR17a, Asm. 2]). The state constraint set $\mathbb{X}$ is closed. The terminal set $\mathbb{X}_\mathrm{f} \subseteq \mathbb{X}$ is closed and contains the reference state vector $\underline{x}_\mathrm{f}$ in its interior. The input constraint set $\mathbb{U}$ is compact and contains the reference input vector $\underline{u}_\mathrm{f}$.

Recall that a compact subset of the Euclidean space is closed and bounded (e.g., [Blo97, Sec. 1.6]). Hence, the input constrained set $\mathbb{U}$ includes the maximum element $\max \mathbb{U} := \underline{u}_\mathrm{max} = (u_{\mathrm{max},1}, u_{\mathrm{max},2}, ..., u_{\mathrm{max},m})^\mathsf{T}$ and the minimum element $\min \mathbb{U} := \underline{u}_\mathrm{min} = (u_{\mathrm{min},1}, u_{\mathrm{min},2}, ..., u_{\mathrm{min},m})^\mathsf{T}$. Since the state constraint set $\mathbb{X}$ is assumed to be a closed subset of the Euclidean space, it contains all of its boundary points (see, e.g., [Blo97, Sec. 1.2]). The open-loop system must explicitly adhere to input and state constraints. Therefore, the set of all admissible control sequences is defined by:

$$\mathcal{U}_N(\underline{x}_0) := \{\underline{\mathbf{u}} \in \mathbb{U}^N \mid \underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}}) \in \mathbb{X}, \forall k \in \mathbb{N}^{[0,N-1]}, \underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{u}}) \in \mathbb{X}_\mathrm{f}\}. \tag{3.1.5}$$

The constraint sets $\mathbb{U}, \mathbb{X}$, and $\mathbb{X}_\mathrm{f}$ can be represented by a finite number of non-strict inequalities (polyhedral constraints). As already noted, the map $\underline{\mathbf{u}} \mapsto \underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}})$ is continuous. Therefore, the admissible set $\mathcal{U}_N(\underline{x}_0)$ is closed and further compact since $\mathcal{U}_N(\underline{x}_0) \subset \mathbb{U}^N$ [Raw+20, Prop. 2.4 (b)]. Recall that a closed subset of a compact set is compact (see, e.g., [Blo97, Sec. 1.6]). Based on the previous definitions, the OCP with a full degree of freedom in control is given by:

$$V_N(\underline{x}_0) := \min_{\underline{\mathbf{u}} \in \mathcal{U}_N(\underline{x}_0)} J_N(\underline{x}_0, \underline{\mathbf{u}}). \tag{3.1.6}$$

Since the cost function $J_N(\cdot)$ is continuous $\big($and thus is $J_N(\underline{x}_0, \cdot)\big)$ and the set $\mathcal{U}_N(\underline{x}_0)$ is compact, the Weierstrass extreme value theorem can be used to prove that OCP (3.1.6) has a solution for all $\underline{x}_0 \in \mathcal{X}_N$ [Raw+20, Prop. 2.4 (c)]. In other words, if $\mathcal{U}_N(\underline{x}_0)$ is not empty, OCP (3.1.6) is called feasible for $\underline{x}_0$ [GP17, p. 49]. Consequently, the feasible set contains all initial state vectors for which an admissible control sequence exists:

$$\mathcal{X}_N := \{\underline{x}_0 \in \mathbb{X} \mid \mathcal{U}_N(\underline{x}_0) \neq \emptyset\}. \tag{3.1.7}$$

The feasible set $\mathcal{X}_N$ is closed [Raw+20, Prop. 2.10]. Here, the proof of closedness is based on the Bolzano-Weierstrass convergence theorem, assuming that the transition map is continuous (see Asm. 3.1.1) and the terminal set $\mathbb{X}_\mathrm{f}$ is closed (see Asm. 3.1.3). The optimal solution for which OCP (3.1.6) attains its globally optimal solution is denoted by $\underline{\mathbf{u}}^*(\underline{x}_0) := \big(\underline{u}^*(0, \underline{x}_0), \underline{u}^*(1, \underline{x}_0), ..., \underline{u}^*(N-1, \underline{x}_0)\big) \in \mathcal{U}_N(\underline{x}_0)$. If the solution $\underline{\mathbf{u}}^*(\underline{x}_0)$ is not unique, then there are at least two control sequences that return the same cost function value. In such cases, assume that the optimization framework simply selects a control sequence. In the remainder of this dissertation, the term open-loop control refers to the conventional optimal control without applying a feedback law.

3.2. Conversion to a Nonlinear Program

This section summarizes the main steps to transform the discrete-time OCP (3.1.6) into a mathematical representation that a numerical optimization algorithm can process. The following description only presents the main steps up to the interface of an ideal optimizer. Appendix A.1 provides visual details on structure and sparsity exploitation, while Appendix A.2 summarizes the fundamentals of constrained optimization.
Conventional optimization algorithms, such as the general purpose solver IPOPT [WB06], solve common parameter optimization problems that are subject to equality and inequality constraints. A nonlinear cost function and/or nonlinear constraints turn the optimization problem into a nonlinear program (NLP). A common NLP adopts the following structure (e.g., [NW06, Eq. 12.1]):

$$\min_{\underline{z} \in \mathbb{R}^\omega} \psi(\underline{z}) \quad \text{subject to} \quad \begin{cases} h_i(\underline{z}) = 0, \ \forall\, i \in \mathcal{E} \subset \mathbb{N}_0, \\ g_i(\underline{z}) \leq 0, \ \forall\, i \in \mathcal{I} \subset \mathbb{N}_0. \end{cases} \tag{3.2.1}$$

Here, $\psi : \mathbb{R}^\omega \mapsto \mathbb{R}_0^+$, $h_i, g_i : \mathbb{R}^\omega \mapsto \mathbb{R}$ represent the continuous cost and constraint functions, respectively, with $\omega \in \mathbb{N}$. The optimal solution $\underline{z}^*$ has to adhere to the constraints $h_i(\underline{z}^*) = 0$ and $g_i(\underline{z}^*) \leq 0$ for all indices $i \in \mathcal{E}$ and $i \in \mathcal{I}$. A solution to NLP (3.2.1) has to satisfy the Karush-Kuhn-Tucker (KKT) conditions, which represent the necessary conditions in constrained optimization (e.g., [NW06, Th. 12.1]). To solve

OCP (3.1.6) numerically, first, all constrained sets are represented by a system of algebraic inequalities.
The intersection of a finite number of closed half-spaces $l_\mathrm{x} \in \mathbb{N}$ can represent the closed state constraint set $\mathbb{X}$, also referred to as the polyhedral constraint set (e.g., [Bor+17, Sec. 4.2]):

$$\mathbb{X} := \{\underline{x} \in X \mid \underline{G}_\mathrm{x}\underline{x} \preceq \underline{h}_\mathrm{x}\},\ \underline{G}_\mathrm{x} \in \mathbb{R}^{l_\mathrm{x}\times p},\ \underline{h}_\mathrm{x} \in \mathbb{R}^{l_\mathrm{x}}. \tag{3.2.2}$$

Since this dissertation only focuses on box-constraints, a closed convex polytope represents the compact input constraint set (e.g., [Bor+17, Sec. 4.2]):

$$\mathbb{U} := \{\underline{u} \in U \mid \underline{G}_\mathrm{u}\underline{u} \preceq \underline{h}_\mathrm{u}\},\ \underline{G}_\mathrm{u} \in \mathbb{R}^{l_\mathrm{u}\times m},\ \underline{h}_\mathrm{u} \in \mathbb{R}^{l_\mathrm{u}}. \tag{3.2.3}$$

Here, $l_\mathrm{u} \in \mathbb{N}$ is the number of closed half-spaces. A polytope is said to be a bounded polyhedron (e.g., [Bor+17, Sec. 4.2]). For example, if a second-order system with a single input is subject to input and state box-constraints, $u_\mathrm{min} \leq u \leq u_\mathrm{max}$ and $(x_{\mathrm{min},1}, x_{\mathrm{min},2})^\intercal \preceq \underline{x} \preceq (x_{\mathrm{max},1}, x_{\mathrm{max},2})^\intercal$, the polyhedral constraints are defined as:

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} u \preceq \begin{pmatrix} u_\mathrm{max} \\ -u_\mathrm{min} \end{pmatrix},\ \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}^\intercal \underline{x} \preceq (x_{\mathrm{max},1}, -x_{\mathrm{min},1}, x_{\mathrm{max},2}, -x_{\mathrm{min},2})^\intercal. \tag{3.2.4}$$

In conventional MPC, the terminal set $\mathbb{X}_\mathrm{f}$ is often approximated by a compact terminal level set (e.g., [MM93; CA98] and [Raw+20, Sec. 2.5.5]):

$$\mathrm{lev}_\pi F := \{\underline{x} \in \mathbb{X} \mid F(\underline{x}) \leq \pi\} \subseteq \mathbb{X}_\mathrm{f},\ \text{for some small } \pi > 0. \tag{3.2.5}$$

With $F(\underline{x}) := \|\underline{x}\|^2_{\underline{P}}$ and $\underline{P}$ positive definite, the terminal level set $\mathrm{lev}_\pi F$ is an ellipsoid. Therefore, a single inequality determines compliance with the terminal constraint.
As mentioned in the previous chapter, a discrete-time OCP renders the steps of discretizing and paramterizing the control trajectory superfluous. In the discrete-time domain, every control vector $\underline{u}(k)$ with $k \in \mathbb{N}^{[0,N-1]}$ is subject to optimization. However, there exist different approaches for integrating the system dynamics (3.1.1) into the NLP (3.2.1). Approaches for embedding system dynamics include recursive elimination, multiple shooting, and full discretization [GP17, Sec. 12.1].

Recursive Elimination

Let the parameter vector be defined as a control sequence $\underline{\mathbf{z}} := \big(\underline{u}(0), \underline{u}(1), ..., \underline{u}(N-1)\big) \in \mathbb{R}^{mN}$ with $\underline{u}(k) \in \mathbb{R}^m$ for all $k \in \mathbb{N}^{[0,N-1]}$. Recursive elimination uses this parameter vector $\underline{\mathbf{z}}$ to generate $N+1$ state vectors by simply executing the operator $\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{z}})$ for all $k \in \mathbb{N}^{[0,N]}$ with $\underline{\varphi}(0, \underline{x}_0, \underline{\mathbf{z}}) = \underline{x}_0$. The state constraint function is defined by:

$$\underline{\mathcal{G}}\big(\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{z}})\big) := \begin{cases} \underline{G}_\mathrm{x}\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{z}}) - \underline{h}_\mathrm{x} & \text{if } \mathbb{X} \subset X, \\ \underline{0} & \text{if } \mathbb{X} = X. \end{cases} \tag{3.2.6}$$

The terminal constraint function is defined by:

$$\underline{\mathcal{F}}\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}})\big) := \begin{cases} F\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}})\big) - \pi & \text{if } \underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}}) \in \mathbb{X}_\mathrm{f} := \mathrm{lev}_\pi F, \\ \underline{G}_\mathrm{x}\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}}) - \underline{h}_\mathrm{x} & \text{if } \underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}}) \in \mathbb{X}_\mathrm{f} = \mathbb{X} \subset X, \\ \underline{0} & \text{if } \underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}}) \in \mathbb{X}_\mathrm{f} = \mathbb{X} = X. \end{cases} \tag{3.2.7}$$

The following NLP includes the solution to the nonlinear dynamics (similar to [GP17, Sec. 12.1]):

$$\min_{\underline{\mathbf{z}} \in \mathbb{R}^{mN}} \psi(\underline{\mathbf{z}}) := \min_{\underline{\mathbf{z}} \in \mathbb{R}^{mN}} \sum_{k=0}^{N-1} \ell\big(\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{z}}), \underline{u}(k)\big) + F\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}})\big) \tag{3.2.8}$$

subject to

$$\underline{G}_\mathrm{u}\, \underline{z}(k) \preceq \underline{h}_\mathrm{u},\ \forall\, k \in \mathbb{N}^{[0,N-1]}, \qquad (\underline{u}(k) \in \mathbb{U})$$

$$\underline{\mathcal{G}}\big(\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{z}})\big) \preceq \underline{0},\ \forall\, k \in \mathbb{N}^{[1,N-1]}, \qquad (\underline{x}(k) \in \mathbb{X})$$

$$\underline{\mathcal{F}}\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{z}})\big) \preceq \underline{0}. \qquad (\underline{x}(N) \in \mathbb{X}_\mathrm{f})$$

The optimization routine triggers an external algorithm to solve the system dynamics each time the NLP optimizer modifies the elements of the parameter vector $\underline{\mathbf{z}}$ [GP17, Sec. 12.1]. In direct transcription, this strategy is referred to as the sequential approach, refer, for example, to [Bin+01]. Assume that the optimizer provides a variation of an element in $\underline{u}(l)$ with $l \in \mathbb{N}_0$. Then, this variation affects the state trajectory (only) on the interval $\mathbb{N}^{[l+1,N]}$. A parameter variation at $l = 0$, however, requires a simulation over the entire horizon. This observation results in a triangular structure of the Jacobian matrix of the constraint functions (see App. A.1 for visualization). When calculating derivatives based on finite differences, exploiting matrix structures systematically reduces computation times (e.g., [GP17, Sec. 12.4]). Consequently, numerical derivatives need to be calculated only for the non-zero elements. This sequential approach has two major drawbacks as discussed in [GP17, Sec. 12.1] and [Bet98]. A nontrivial relationship between the control and state trajectories complicates the initialization of the parameter vector. The second drawback is the numerical sensitivity to small parameter variations. The longer the horizon, the greater the effect of small changes at the beginning of the parameter vector $\underline{\mathbf{z}}$ on the states at the end of the horizon. Both properties may lead to poor convergence behavior of the NLP optimization algorithm.

Multiple Shooting

Multiple shooting is originally used to solve boundary value problems where the solution of a differential equation is subject to boundary conditions (e.g., [SB93, Sec. 7.3]). Direct multiple shooting, however, solves continuous-time OCPs [BP84]. Both approaches share the same concept, which can also be transferred to the discrete-time domain as shown in [GP17, Sec. 12.4]. The main idea is to divide the prediction horizon into $T \in \mathbb{N}$ shooting intervals and introduce $T + 1$ shooting nodes $\underline{s}_q \in \mathbb{R}^p$ with $q \in \mathbb{N}^{[0,T]}$. Let $L_q \in \mathbb{N}_0$ denote the number of controls within the q-th shooting interval. For example, L_0 is the number of controls between $\underline{s}_0$ and $\underline{s}_1$. The function $I(q) := \sum_{i=0}^{q-1} L_i$ with $I(0) := 0$ determines the time point that is associated with the shooting node $\underline{s}_q$ on the prediction horizon. On each individual shooting interval, the optimization routine applies recursive elimination to generate the required state vectors. Appendix A.1 visualizes the principle of multiple shooting for the discrete-time domain. Let the parameter vector be defined as $\underline{\mathbf{z}} := (\underline{\mathbf{s}}, \underline{\mathbf{u}}) \in \mathbb{R}^{p(T+1)+mN}$ with $\underline{\mathbf{s}} := \big(\underline{s}_0, \underline{s}_1, ..., \underline{s}_T\big) \in \mathbb{R}^{p(T+1)}$ and let

$\underline{\mathbf{u}}_q := \big(\underline{u}(I(q)), \underline{u}(I(q)+1), ..., \underline{u}(I(q)+L_q-1)\big) \in \mathbb{R}^{mL_q}$ denote the part of the control trajectory which corresponds to the q-th shooting interval. Then, the optimizer solves the following NLP (similar to [GP17, Sec. 12.1]):

$$\min_{\underline{\mathbf{z}} \,\in\, \mathbb{R}^{p(T+1)+mN}} \psi(\underline{\mathbf{z}}) := \min_{\underline{\mathbf{z}} \,\in\, \mathbb{R}^{p(T+1)+mN}} \sum_{q=0}^{T-1} \sum_{k=0}^{L_q-1} \ell\big(\underline{\varphi}(k, \underline{s}_q, \underline{\mathbf{u}}_q), \underline{u}(I(q)+k)\big) + F(\underline{s}_T) \tag{3.2.9}$$

subject to

$$\begin{aligned}
\underline{s}_0 &= \underline{x}_0, && \text{(initialization)}\\
\underline{G}_\mathrm{u}\underline{u}(k) &\preceq \underline{h}_\mathrm{u}, \ \forall\, k \in \mathbb{N}^{[0,N-1]}, && (\underline{u}(k) \in \mathbb{U})\\
\underline{\mathcal{G}}\big(\underline{\varphi}(k, \underline{s}_q, \underline{\mathbf{u}}_q)\big) &\preceq \underline{0}, \ \forall\, q \in \mathbb{N}^{[0,T-1]}, \forall\, k \in \mathbb{N}^{[0,L_q-1]}, && (\underline{x}(k) \in \mathbb{X})\\
\underline{\varphi}(L_q, \underline{s}_q, \underline{\mathbf{u}}_q) - \underline{s}_{q+1} &= \underline{0}, \ \forall\, q \in \mathbb{N}^{[0,T-1]}, && \text{(continuity)}\\
\underline{\mathcal{F}}(\underline{s}_T) &\preceq \underline{0}. && (\underline{x}(N) \in \mathbb{X}_\mathrm{f})
\end{aligned}$$

Here, the shooting nodes are directly subject to optimization since $\underline{\varphi}(0, \underline{s}_q, \underline{\mathbf{u}}_q) = \underline{s}_q$ applies. Note that NLP (3.2.9) also includes intermediate state vectors between two shooting nodes $\underline{s}_q$ and $\underline{s}_{q+1}$ for all $q \in \mathbb{N}^{[0,T-1]}$. Although NLP (3.2.9) involves more optimization parameters than NLP (3.2.8) and is subject to additional equality constraints enforcing continuity, multiple shooting enables faster convergence of optimization [BP84]. If the optimizer varies the elements of a shooting node $\underline{s}_q$, the optimization routine only needs to execute recursive elimination up to the next shooting node $\underline{s}_{q+1}$, thereby generating state vectors on the interval $\mathbb{N}^{[I(q)+1, I(q)+L_q]}$. Variations of the input vector $\underline{u}(I(q)+l)$ only affect the state trajectory on the interval $\mathbb{N}^{[I(q)+l+1, I(q)+L_q]}$. The last shooting node $\underline{s}_T$ is directly subject to the terminal constraint with $L_T = 0$. Since the continuity of the state trajectory is only required after the optimization converges, first- and second-order derivative information matrices exhibit sparse patterns depending on the number of shooting intervals T [BP84] (see also [GP17, Sec. 12.4] and App. A.1). Sparse solvers such as IPOPT build on sparse linear system solvers and sparse algebra and thus can process structure information, provided externally, to accelerate optimization.

Full Discretization

By setting $T = N$, the multiple shooting approach turns into the full discretization approach [GP17, Sec. 12.1], implying $L_q = 1$ for all $q \in \mathbb{N}^{[0,N]}$. In direct transcription, full discretization is referred to as the simultaneous approach since the NLP optimizer directly accesses and optimizes the state trajectory (e.g., [Bin+01]). With full discretization, each interval has exactly one control such that $\underline{u}(k) = \underline{\mathbf{u}}_k$ applies. Let the definitions $\underline{s}_k := \underline{x}(k)$ and $\underline{s}_{k+1} := \underline{x}(k+1)$ hold. Then, the continuity constraint in (3.2.9) is defined by $\underline{\varphi}\big(1, \underline{x}(k), \underline{u}(k)\big) - \underline{x}(k+1) = \underline{0}$ for all $k \in \mathbb{N}^{[0,N-1]}$. With full discretization, there are no intermediate state vectors that are not directly subject to optimization such as with $T < N$. The full discretization approach greatly simplifies NLP (3.2.9), which is shown in (A.1.3) in Appendix A.1.

3.3. Conventional Nonlinear Model Predictive Control

The main idea in MPC is to solve OCP (3.1.6) at every closed-loop time instant $n \in \mathbb{N}_0$ and to apply the first control vector of the optimal sequence $\underline{\mathbf{u}}^*(\underline{x}_0)$ to the plant (e.g., [ML99; May+00]). To this end, conventional MPC rests upon the following implicit control law:

$$\underline{\mu}(\underline{x}_0) = \big(\mu_1(\underline{x}_0), \mu_2(\underline{x}_0), ..., \mu_m(\underline{x}_0)\big)^\intercal := \underline{u}^*(0, \underline{x}_0). \tag{3.3.1}$$

The closed-loop system evolves as:

$$\underline{x}_0^+ := \underline{x}_\mu(n+1) = \underline{f}\Big(\underline{x}_\mu(n), \underline{\mu}\big(\underline{x}_\mu(n)\big)\Big), \underline{x}_\mu(0) := \underline{x}_{\mu,0}. \tag{3.3.2}$$

Throughout this dissertation, the index μ and the variable n indicate the closed-loop system and its evolution with the state trajectory $\underline{x}_\mu : \mathbb{N}_0 \to X$. The variable $\underline{x}_{\mu,0}$ defines the initial state at time $n = 0$. At each time instant n, OCP (3.1.6) is initialized with $\underline{x}_0 := \underline{x}_\mu(n)$. In the nominal case, the successor state $\underline{x}_0^+ := \underline{x}_\mu(n+1)$ initializes OCP (3.1.6) at time instant $n+1$. Let $\underline{\varphi}_\mu(n, \underline{x}_{\mu,0})$ denote the closed-loop state trajectory at time instant n. In the rest of this dissertation, the term closed-loop control is used as a synonym for MPC.

Definition 3.3.1: *Nominal setting.* The internal model and the plant dynamics use the same state space representation (3.1.1) (no model mismatch). In addition, there are no disturbances which could affect the evolution of the open- and/or closed-loop system. Solving a feasible OCP requires an execution time of $t_{\text{run}} = 0\,\text{s}$.

Unless otherwise stated, this dissertation always investigates the nominal case.
The infinite horizon formulation with $N \to \infty$, $\mathbb{X}_\text{f} = \mathbb{X}$, and $F(\cdot) := 0$ represents a special case and is particularly suitable to introduce theoretical analyses in MPC [GP17, Ch. 4]. If there exists a finite solution at time instant n, it follows from Bellman's principle of optimality that $V_\infty(\underline{x}_0) = \ell\big(\underline{x}_0, \underline{\mu}(\underline{x}_0)\big) + V_\infty(\underline{x}_0^+)$ (e.g., [GP17, Thm. 4.6]). Therefore, the closed- and open-loop trajectories coincide exactly in the case of an infinite horizon assuming nominal setting (e.g., [GP17, Cor. 4.7]). Solving OCP (3.1.6) at time instances $n > 0$ is actually not required. Closed-loop stability mainly follows from assuming asymptotic controllability [GP17, Def. 4.2, Thm. 4.3+4.8]. However, the realization of an infinite horizon is not possible in practice for arbitrary systems, since an infinite horizon obviously implies an infinite number of optimization parameters.
Conventional MPC with a finite receding horizon, however, optimizes the internal system behavior only for some steps in the future, allowing practical realization. Thus, in order to transfer the nonlinear system (3.1.1) to the origin and to stabilize the origin with the system of interest, it is necessary to solve OCP (3.1.6) repeatedly as the system evolves. Since the implicit control law (3.3.1) relies on the solution to an optimization, it is important to ensure that $\underline{x}_\mu(n) \in \mathcal{X}_N$ for all $n \in \mathbb{N}_0$. Recall that OCP (3.1.6) is only feasible if $\underline{x}_\mu(n) \in \mathcal{X}_N$.

Basic Stability Definitions

This subsection summarizes the most important definitions for MPC concerning asymptotic stability of the origin.

Definition 3.3.2: *Positive invariant set* ([Raw+20, Def. 2.9]). The set $\mathcal{X} \subseteq \mathbb{R}^p$ is positive invariant for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$ if $\underline{x}^+ \in \mathcal{X}$ holds for all $\underline{x} \in \mathcal{X}$.

Definition 3.3.3: *Control invariant set* ([Raw+20, Def. 2.9]). Let $\mathcal{X} \subseteq \mathbb{R}^p$ be a control invariant set for the system $\underline{x}^+ = \underline{f}(\underline{x}, \underline{u})$. Then, there exists a control vector $\underline{u} \in \mathbb{U}$ for all $\underline{x} \in \mathcal{X}$ such that $\underline{x}^+ \in \mathcal{X}$.

Assume that the feasible set $\mathcal{X}_N$ is positive invariant for the closed-loop system (3.3.2). Then, $\mathcal{X}_N$ is also called to be recursively feasible [GP17, p. 51]. This property is a necessary prerequisite for asymptotic stability of the origin in $\mathcal{X}_N$. Let $\underline{\varphi}_{\mathrm{g}}(n, \underline{x})$ denote the state trajectory of the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$ at time instant n with the initial state $\underline{x}$. In the following, let $\mathcal{X} \subseteq \mathbb{R}^p$ be a positive invariant set for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$. The closed unit ball is denoted by $\mathcal{B}_\delta := \{\underline{x} \in X \,|\, \|\underline{x}\| \leq \delta\}$.

Definition 3.3.4: *Local stability in the sense of Lyapunov* [1]. The origin is locally stable in $\mathcal{X}$ for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$ if for any $\epsilon > 0$ there exists a $\delta > 0$ such that $\|\underline{\varphi}_{\mathrm{g}}(n, \underline{x})\| \leq \epsilon$ holds for all $n \in \mathbb{N}_0$ and all $\underline{x} \in \mathcal{X} \cap \mathcal{B}_\delta$.

Definition 3.3.5: *Attraction* [1]. The origin is attractive in $\mathcal{X}$ for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$ if $\|\underline{\varphi}_{\mathrm{g}}(n, \underline{x})\| \to 0$ as $n \to \infty$ for all $\underline{x} \in \mathcal{X}$.

Definition 3.3.6: *Asymptotic stability* [1]. The origin is asymptotically stable in $\mathcal{X}$ for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$ if it is locally stable and attractive in $\mathcal{X}$. The origin is globally asymptotically stable if it is asymptotically stable on $\mathcal{X} = X$.

A function $\alpha : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is said to be of class $\mathcal{K}$ if it is continuous, strictly increasing and zero at zero. If it is unbounded in addition, then it is a member of the class $\mathcal{K}_\infty$. Note that the relationship $\mathcal{K}_\infty \subset \mathcal{K}$ holds.

Definition 3.3.7: *Lyapunov function* [2]. The function $V : X \mapsto \mathbb{R}_0^+$ is a Lyapunov function candidate on the positive invariant set $\mathcal{X}$ for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$ if it satisfies the following inequalities with $\alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}_\infty$ and $\alpha_3(\cdot) \in \mathcal{K}$ for all $\underline{x} \in \mathcal{X}$:

$$\alpha_1(\|\underline{x}\|) \leq V(\underline{x}) \leq \alpha_2(\|\underline{x}\|), \quad V\big(\underline{\tilde{g}}(\underline{x})\big) \leq V(\underline{x}) - \alpha_3(\|\underline{x}\|). \tag{3.3.3}$$

Theorem 3.3.1: *Lyapunov's direct method* [3]. Let $\mathcal{X}$ be the positive invariant set for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$. Suppose that there exists a Lyapunov function $V(\cdot)$ on the set $\mathcal{X}$ for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$. Then, the origin is asymptotically stable in $\mathcal{X}$ for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$. The origin is globally asymptotically stable for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x})$ if $\mathcal{X} = X$. The origin is exponentially stable if there exist power-law bounds $\alpha_i(\|\underline{x}\|) = b_i\|\underline{x}\|^\nu$ with $i \in \mathbb{N}^{[1,3]}$ and $\nu, b_i \in \mathbb{R}^+$.

Stability Analysis with Stabilizing Terminal Conditions

For conventional MPC with stabilizing terminal conditions, it can be shown that the optimal cost function $V_N(\cdot)$ satisfies the Lyapunov inequalities in (3.3.3) [May+00].

[1] See [GP17, Def. 2.14], [Raw+20, Def. B.10]. [2] See [GP17, Def. 2.18], [Raw+20, Def. 2.12]. [3] See [GP17, Thm. 2.19], [Raw+20, Thm. 2.13].

Assumption 3.3.1: *Bounds for cost functions* $\big($[Raw+20, Asm. 2.14]$\big)$. There exists a function $\alpha_\ell(\cdot) \in \mathcal{K}_\infty$ such that $\ell(\underline{x},\underline{u}) \geq \alpha_\ell(\|(\underline{x},\underline{u})\|) \geq \alpha_\ell(\|\underline{x}\|)$ holds for all $\underline{x} \in \mathbb{X}$ and $\underline{u} \in \mathbb{U}$. There exists a function $\alpha_\mathrm{f}(\cdot) \in \mathcal{K}_\infty$ such that $\alpha_\mathrm{f}(\underline{x}) \geq F(\underline{x})$ holds for all $\underline{x} \in \mathbb{X}_\mathrm{f}$.

For costs in quadratic form with positive definite weighting matrices $\underline{Q}$ and $\underline{R}$, the comparison functions follow directly from the scaled versions of the individual cost functions. Stabilizing terminal conditions further build on a local control Lyapunov function (CLF) and bind the final predicted state to a control invariant terminal set $\mathbb{X}_\mathrm{f}$ (e.g., [May+00, Asm. A1-A4] and [Raw+20, Asm. 2.14]).

Assumption 3.3.2: *Control invariant terminal set* $\big($[Raw+20, Asm. 2.14]$\big)$. The terminal set $\mathbb{X}_\mathrm{f}$ is control invariant for system (3.1.1). The local control law $\underline{\kappa} : X \mapsto U$ ensures that:

$$\underline{f}\big(\underline{x},\underline{\kappa}(\underline{x})\big) \in \mathbb{X}_\mathrm{f} \;\text{ and }\; \underline{\kappa}(\underline{x}) \in \mathbb{U} \;\text{ if }\; \underline{x} \in \mathbb{X}_\mathrm{f}. \tag{3.3.4}$$

In addition, the local controller renders the origin asymptotically stable in $\mathbb{X}_\mathrm{f}$ for the system $\underline{x}^+ = \underline{f}\big(\underline{x},\underline{\kappa}(\underline{x})\big)$ such that $F(\cdot)$ represents a local CLF for all $\underline{x} \in \mathbb{X}_\mathrm{f}$:

$$F\Big(\underline{f}\big(\underline{x},\underline{\kappa}(\underline{x})\big)\Big) - F(\underline{x}) \leq -\ell\big(\underline{x},\underline{\kappa}(\underline{x})\big). \tag{3.3.5}$$

The following theorem summarizes stability properties of conventional MPC with stabilizing terminal conditions.

Theorem 3.3.2: *Asymptotic stability with conventional MPC* $\big($[Raw+20, Thm. 2.19]$\big)$. Suppose Assumptions 3.1.1-3.1.2 and 3.3.1-3.3.2 hold. Then, the optimal cost function $V_N(\cdot)$ represents a Lyapunov function in the set $\mathcal{X}_N$ for the closed-loop system (3.3.2) with $\alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}_\infty$ for all $\underline{x}_0 \in \mathcal{X}_N$:

$$\alpha_1(\|\underline{x}_0\|) \leq V_N(\underline{x}_0) \leq \alpha_2(\|\underline{x}_0\|), \quad V_N\Big(\underline{f}\big(\underline{x}_0,\underline{\mu}(\underline{x}_0)\big)\Big) \leq V_N(\underline{x}_0) - \alpha_1(\|\underline{x}_0\|). \tag{3.3.6}$$

Consequently, the origin is asymptotically stable in the positive invariant set $\mathcal{X}_N$ for the closed-loop system (3.3.2).

The lower bound in Theorem 3.3.2 follows from Assumption 3.3.1 with $\alpha_1(\cdot) := \alpha_\ell(\cdot)$. Optimality of OCP (3.1.6) and Assumption 3.3.2 imply that

$$\ell\big(\underline{x},\underline{u}^*(0,\underline{x})\big) + F\Big(\underline{f}\big(\underline{x},\underline{u}^*(0,\underline{x})\big)\Big) \leq \ell\big(\underline{x},\underline{\kappa}(\underline{x})\big) + F\Big(\underline{f}\big(\underline{x},\underline{\kappa}(\underline{x})\big)\Big) \leq F(\underline{x}), \tag{3.3.7}$$

holds for all $\underline{x} \in \mathbb{X}_\mathrm{f}$. Hence, $V_1(\underline{x}) \leq F(\underline{x})$ applies for all $\underline{x} \in \mathbb{X}_\mathrm{f}$ [Raw+20, Prop. 2.18]. Based on the principle of optimality, this monotonicity property can be extended such that $0 \leq V_{N+1}(\underline{x}) \leq V_N(\underline{x}) \leq F(\underline{x}) \leq \alpha_\mathrm{f}(\|\underline{x}\|)$ holds for all $\underline{x} \in \mathbb{X}_\mathrm{f}$ and $N \in \mathbb{N}$ [Raw+20, Prop. 2.16]. Since by Assumption 3.1.3 the terminal set $\mathbb{X}_\mathrm{f}$ contains the origin in its interior, the origin does not lie on the boundary of the terminal set. This allows to enclose the origin with an open ball which is fully contained in $\mathbb{X}_\mathrm{f}$. Since the monotonicity property applies in some neighborhood of the origin, $V_N(\cdot)$ is continuous at the origin [Raw+20, Prop. 2.16]. Note that a Lyapunov function candidate only needs to be continuous at the origin and not on the whole set of interest. Since $V_N(\cdot)$ is bounded on every compact subset of the feasible set $\mathcal{X}_N$, it is locally bounded on $\mathcal{X}_N$

[Raw+20, Prop. 2.15]. The proof in [Raw+20, Prop. 2.15] relies on the fact that the optimal cost function $V_N(\cdot)$ inherits the upper cost bound from the continuous cost function $J_N(\cdot)$ on the same compact subset of $\mathcal{X}_N \times \mathcal{U}_N(\underline{x})$ with respect to the first argument. The existence of the upper bound $\alpha_2(\cdot)$ in Theorem 3.3.2 finally follows from [RR17b, Prop. 14]. In fact, Proposition 14 in [RR17b] states that there exists a comparison function $\alpha(\cdot) \in \mathcal{K}$ in some closed set $\mathcal{X} \subseteq X$ for the function $V : \mathcal{X} \mapsto \mathbb{R}_0^+$ if $\mathcal{X}$ contains the origin in its interior, $V(\cdot)$ is continuous at the origin and locally bounded on $\mathcal{X}$. Then, the inequality $V(\underline{x}) \leq \alpha(\|\underline{x}\|)$ holds for all $\underline{x} \in \mathcal{X}$. The optimal cost function $V_N(\cdot)$ satisfies all these requirements on the set $\mathcal{X}_N$.
Let $\underline{\Omega}_{\text{sta}} : \mathbb{X} \times \mathbb{U}^N \mapsto \mathbb{U}^N$ be the operator which shifts and truncates a control sequence by one step and then appends the local control law. Applying this operator to the optimal solution generates the following control sequence with $\underline{x}_N^* := \underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{u}}^*(\underline{x}_0))$:

$$\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) := \underline{\Omega}_{\text{sta}}\big(\underline{x}_0, \underline{\mathbf{u}}^*(\underline{x}_0)\big) = \big(\underline{u}^*(1, \underline{x}_0), \underline{u}^*(2, \underline{x}_0), ..., \underline{u}^*(N-1, \underline{x}_0), \underline{\kappa}(\underline{x}_N^*)\big). \quad (3.3.8)$$

The following inequality follows by substituting the shifted sequence $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)$ into the cost function $J_N(\cdot)$ (e.g., [Raw+20, pp. 116-118]):

$$\begin{aligned} V_N(\underline{x}_0^+) \leq J_N\big(\underline{x}_0^+, \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)\big) =& V_N(\underline{x}_0) - \ell\big(\underline{x}_0, \underline{\mu}(\underline{x}_0)\big) + \\ & - F\big(\underline{x}_N^*\big) + \ell\big(\underline{x}_N^*, \underline{\kappa}(\underline{x}_N^*)\big) + F\Big(\underline{f}\big(\underline{x}_N^*, \underline{\kappa}(\underline{x}_N^*)\big)\Big). \end{aligned} \quad (3.3.9)$$

By Assumption 3.3.2, the last three terms in (3.3.9) are smaller or equal to zero. Therefore, combining (3.3.9) with Assumption 3.3.2 yields the second inequality in Theorem 3.3.2 for all $\underline{x}_0 \in \mathcal{X}_N$ (e.g., [Raw+20, pp. 116-118]):

$$V_N(\underline{x}_0^+) \leq V_N(\underline{x}_0) - \ell\big(\underline{x}_0, \underline{\mu}(\underline{x}_0)\big) \leq V_N(\underline{x}_0) - \alpha_1(\|\underline{x}_0\|). \quad (3.3.10)$$

By Theorem 3.3.1, the origin is asymptotically stable for the closed-loop system (3.3.2). If $\underline{\mathbf{u}}^*(\underline{x}_0^+) = \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)$ is the optimal solution to OCP (3.1.6) at time $n+1$, the presented derivation assumes that the optimizer finds this solution. Hence, explicit switching to the local controller is not required. If $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)$ is not the optimal solution, then there exists a better solution that the optimizer can also find. MPC with stabilizing terminal conditions inherits the invariance property of the terminal set $\mathbb{X}_{\text{f}}$ such that the feasible set $\mathcal{X}_N$ becomes positive invariant for the closed-loop system (3.3.2).

Remark 3.3.1: *Terminal equality constraint.* A terminal equality constraint, as presented in [KG88], allows to bypass the determination of a control invariant set $\mathbb{X}_{\text{f}}$ and a local control law $\underline{\kappa}(\cdot)$ by reducing the terminal set to $\mathbb{X}_{\text{f}} := \{\underline{0}_p\}$ with $F(\underline{0}_p) = 0$. Here, the local control law $\underline{\kappa}(\cdot) := \underline{0}_m$ satisfies Assumption 3.3.2. Only the derivation of the upper bound $\alpha_2(\cdot)$ in Theorem (3.3.1) requires some modifications (e.g., [GP17, Prop. 5.7]). However, at time $n = 0$, the nonlinear system (3.1.1) must be controlled to the origin in a finite number of steps. Hence, the feasible set $\mathcal{X}_N$ might be smaller compared to the case where $\mathbb{X}_{\text{f}}$ has an interior (see discussion in [GP17, Sec. 7.4]).

Remark 3.3.2: *Inherent robustness.* Inherent robustness is highly relevant for practical applications since it indicates the extent to which the nominal MPC configuration can maintain recursive feasibility and/or asymptotic stability in the presence of small disturbances (e.g., [Raw+20, Sec. 3.2]). Grimm et al. [Gri+04] demonstrate that nominal MPC with state constraints might exhibit absolutely no robustness margins. Numerous papers focus on proving the existence of suitable bounds for external disturbances such that robust recursive feasibility and robust asymptotic stability can be ensured (e.g., [Fin+03; Gri+07; Pan+11; All+17]). The experimental part of this dissertation in Chapter 6 simply assumes robustness margins, while the theoretical parts in Chapter 7 and Chapter 8 inherit the robustness properties from [All+17] under mild assumptions.

Design of Stabilizing Terminal Conditions

The following description follows the most common design guidelines for terminal ingredients, as summarized and prepared in [Raw+20, Sec. 2.5.5]. The nonlinear system (3.1.1) is first linearized at the origin (or steady state in general):

$$\underline{A} := \frac{\partial \underline{f}(\underline{x},\underline{u})}{\partial \underline{x}}\Big|_{(\underline{0}_p,\underline{0}_m)},\ \underline{B} := \frac{\partial \underline{f}(\underline{x},\underline{u})}{\partial \underline{u}}\Big|_{(\underline{0}_p,\underline{0}_m)},\ \underline{x}^+ = \underline{A}\,\underline{x} + \underline{B}\underline{u},\ \forall\, \underline{x} \in X,\ \forall\, \underline{u} \in U. \tag{3.3.11}$$

In this dissertation, linearization relies on central finite differences.

Definition 3.3.8: *Stabilizability*. The linear system $\underline{x}^+ = \underline{A}\,\underline{x} + \underline{B}\,\underline{u}$ with $\underline{x} \in X$ and $\underline{u} \in U$ is stabilizable if there exists a local control law $\underline{\kappa}(\underline{x}) \in U$ such that the origin is globally asymptotically stable for the system $\underline{x}^+ = \underline{\tilde{g}}(\underline{x}) = \underline{A}\,\underline{x} + \underline{B}\underline{\kappa}(\underline{x})$ (see, e.g., [AM89, Sec. B.3]).

The definition of stabilizability is weaker than the well-known definition of controllability since it does not ensure that the linear system can be steered to any point in the state space. Thus, controllability implies stabilizability. However, a stabilizable system might be not fully controllable. Here, the uncontrollable modes of $\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{B}\,\underline{u}(k)$ are assumed to be open-loop stable [AM89, Sec. B.3].

Definition 3.3.9: *Stabilizable linearization*. There exists a linear control law $\underline{\kappa}(\underline{x}) := \underline{K}\,\underline{x}$ with $\underline{x} \in X$ and $\underline{K} \in \mathbb{R}^{m\times p}$ such that the system $\underline{x}^+ = (\underline{A} + \underline{B}\,\underline{K})\,\underline{x}$ is stabilizable.

The previous assumption implies that the eigenvalues of the matrix $\underline{A}_\mathrm{K} := (\underline{A} + \underline{B}\,\underline{K})$ are inside the unit circle. The infinite horizon costs for the linearized system with $\underline{u}(k) = \underline{K}\,\underline{x}(k)$ and some $\rho \geq 1$ are denoted by:

$$J_\infty := \rho \sum_{k=0}^{\infty} \underline{x}^\mathsf{T}(k)\,\underline{Q}\,\underline{x}(k) + \underline{u}^\mathsf{T}(k)\,\underline{R}\,\underline{u}(k) = \rho \sum_{k=0}^{\infty} \underline{x}^\mathsf{T}(k)\big(\underline{Q} + \underline{K}^\mathsf{T}\,\underline{R}\,\underline{K}\big)\,\underline{x}(k). \tag{3.3.12}$$

For some initial vector $\underline{x}_0 \in X$, the relationship $\underline{x}(k) = \underline{A}_\mathrm{K}^k\underline{x}_0$ holds for all $k \in \mathbb{N}_0$. With $\underline{Q}_\mathrm{K} := \underline{Q} + \underline{K}^\mathsf{T}\,\underline{R}\,\underline{K}$, the infinite horizon cost function can be simplified as follows:

$$J_\infty := \rho \sum_{k=0}^{\infty} \big(\underline{A}_\mathrm{K}^k\underline{x}_0\big)^\mathsf{T}\,\underline{Q}_\mathrm{K}\big(\underline{A}_\mathrm{K}^k\underline{x}_0\big) = \underline{x}_0^\mathsf{T}\Big(\rho \sum_{k=0}^{\infty} \big(\underline{A}_\mathrm{K}^k\big)^\mathsf{T}\,\underline{Q}_\mathrm{K}\,\underline{A}_\mathrm{K}^k\Big)\,\underline{x}_0. \tag{3.3.13}$$

Therefore, evaluating $F\big(\underline{x}(l)\big) = \underline{x}^\mathsf{T}(l)\,\underline{P}\,\underline{x}(l)$ with $\underline{P} := \rho \sum_{k=l}^{\infty}\big(\underline{A}_\mathrm{K}^k\big)^\mathsf{T}\,\underline{Q}_\mathrm{K}\,\underline{A}_\mathrm{K}^k$ determines the infinite horizon costs for the linear system $\underline{x}(k+1) = \underline{A}_\mathrm{K}\,\underline{x}(k)$, starting from the current state vector $\underline{x}(l)$ with $l \in \mathbb{N}_0$. Because of the infinite horizon formulation, the terminal cost function $F(\cdot)$ represents a global CLF for the linearized system (see [Raw+20, Eq. (2.22)]):

$$(\underline{A}_{\mathrm{K}}\underline{x})^{\mathsf{T}}\,\underline{P}\,(\underline{A}_{\mathrm{K}}\underline{x}) - \underline{x}^{\mathsf{T}}\underline{P}\,\underline{x} = -\underline{x}^{\mathsf{T}}\rho\,\underline{Q}_{\mathrm{K}}\underline{x} \Leftrightarrow F(\underline{A}_{\mathrm{K}}\underline{x}) - F(\underline{x}) = -\rho\ell(\underline{x}, \underline{K}\,\underline{x}),\ \forall\ \underline{x} \in X,$$
$$\Leftrightarrow \underline{A}_{\mathrm{K}}^{\mathsf{T}}\underline{P}\,\underline{A}_{\mathrm{K}} - \underline{P} = -\rho\,\underline{Q}_{\mathrm{K}}.\ \text{(Discrete Lyapunov equation)} \tag{3.3.14}$$

The last equation in (3.3.14) is the well-known discrete Lyapunov equation. Solving this equation yields the matrix $\underline{P}$, which is required for the terminal cost function in (3.1.4). Now, the main idea is to apply the linear control law $\underline{K}\,\underline{x}$ to the nonlinear system (3.1.1) in some small neighborhood of the origin [Raw+20, Eq. (2.23)]:

$$F\big(\underline{f}(\underline{x}, \underline{K}\,\underline{x})\big) - F(\underline{x}) \leq -\ell(\underline{x}, \underline{K}\,\underline{x}) = \underline{x}^{\mathsf{T}}\,\underline{Q}_{\mathrm{K}}\underline{x},\ \forall\ \underline{x} \in \mathrm{lev}_{\pi}F. \tag{3.3.15}$$

The authors of [Raw+20] substitute the global CLF (3.3.14) into the local CLF (3.3.15), which results in:

$$F\big(\underline{f}(\underline{x}, \underline{K}\,\underline{x})\big) - F(\underline{A}_{\mathrm{K}}\underline{x}) \leq (\rho - 1)\,\underline{x}^{\mathsf{T}}\,\underline{Q}_{\mathrm{K}}\underline{x},\ \forall\ \underline{x} \in \mathrm{lev}_{\pi}F. \tag{3.3.16}$$

Finally, the authors of [Raw+20, p. 141] prove that for some $\rho > 1$ there exists a $\pi > 0$ satisfying inequality (3.3.16) and therefore also inequality (3.3.15). Note that satisfying inequality (3.3.15) is necessary to ensure asymptotic stability (see Asm. 3.3.2).

Remark 3.3.3: *Numerical determination of the terminal level set.* To determine the terminal set numerically, the following semi-infinite optimization problem (finite number of optimization parameters, infinite number of constraints) needs to be solved for a selected $\rho > 1$ [May13] (see also [MM93; CA98]):

$$\pi^* := \max\ \pi,\ \text{subject to inequality (3.3.16)},\ \forall\ \underline{x} \in \mathrm{lev}_{\pi}F. \tag{3.3.17}$$

The smaller the parameter ρ, the more similar are the evolutions of the linear and nonlinear system inside the terminal level set $\mathrm{lev}_{\pi}F$. A straightforward approach to solve the optimization problem (3.3.17) approximately is a heuristic search, which increases the value of π incrementally by some small step size while checking inequality (3.3.16) for some finite number of state samples on the boundary of the terminal level set.

Remark 3.3.4: *Constant reference tracking.* In some numerical simulations and experiments in this dissertation, the constant reference pair $(\underline{x}_{\mathrm{f}}, \underline{u}_{\mathrm{f}})$ changes its values. Instead of introducing a time-varying reference pair $\big(\underline{x}_{\mathrm{f}}(n), \underline{u}_{\mathrm{f}}(n)\big)$, the time variable is reset to $n = 0$ each time the reference pair changes. Therefore, the stability analysis is restricted to the time sections where the reference is constant. Since the terminal level set $\mathrm{lev}_{\pi}F$ depends on the system linearization at the current reference pair, this dissertation assumes some constant $\pi > 0$, representing (approximately) a subset of all terminal sets occurring in the individual experiments. A more rigorous approach would have to rely on the parameterized representation of terminal ingredients proposed in [Köh+20]. To further guarantee closed-loop stability even at the time of switching, refer, for example, to the work in [Lim+18]. Here, the authors extend the OCP by a virtual reference, which is considered as an additional optimization variable.

Remark 3.3.5: *Riccati equation*. The gain matrix $\underline{K}$ can be chosen optimally by minimizing analytically the cost function in (3.3.13) (e.g., [Ack85, Sec. 9.4.1]):

$$\underline{P} = \underline{A}^\mathsf{T}\underline{P}\,\underline{A} - (\underline{A}^\mathsf{T}\underline{P}\,\underline{B})(\rho\underline{R} + \underline{B}^\mathsf{T}\underline{P}\,\underline{B})^{-1}(\underline{B}^\mathsf{T}\underline{P}\,\underline{A}) + \rho\underline{Q}, \tag{3.3.18}$$

$$\underline{K} := -(\rho\underline{R} + \underline{B}^\mathsf{T}\underline{P}\,\underline{B})^{-1}\underline{B}^\mathsf{T}\underline{P}\,\underline{A}. \tag{3.3.19}$$

The solution to the Riccati equation (3.3.18) satisfies the Lyapunov equation (3.3.14). Unless otherwise stated, the parameter ρ is equal to one.

3.4. Sampled Data System with Zero-Order Hold

This subsection mainly follows the presentation in [GP17, Sec. 2.2]. The continuous-time, nonlinear, and time-invariant systems in this dissertation are described, respectively, by an ordinary differential equation of the following form with the control trajectory $\underline{\sigma} : \mathbb{R}_0^+ \mapsto \mathbb{R}^m$ and the state trajectory $\underline{\chi} : \mathbb{R}_0^+ \mapsto \mathbb{R}^p$:

$$\dot{\underline{\chi}}(t) := \frac{\mathrm{d}\underline{\chi}(t)}{\mathrm{d}t} = \underline{f}_\mathrm{c}\big(\underline{\chi}(t), \underline{\sigma}(t)\big),\ \underline{\chi}(t) = \big(\chi_1(t), \chi_2(t), ..., \chi_p(t)\big)^\mathsf{T}. \tag{3.4.1}$$

Here, the vector field $\underline{f}_\mathrm{c} : \mathbb{R}^p \times \mathbb{R}^m \mapsto \mathbb{R}^p$ is assumed to be continuous and Lipschitz in its first argument. The control trajectory is assumed to be piecewise constant according to the following formulation with the step time $T_\mathrm{s} \in \mathbb{R}^+$:

$$\underline{\sigma}(t) = \big(\sigma_1(t), \sigma_2(t), ..., \sigma_m(t)\big)^\mathsf{T} := \underline{u}(k),\ \forall\, t \in \big[kT_\mathrm{s}, (k+1)T_\mathrm{s}\big),\ \forall\, k \in \mathbb{N}^{[0,N-1]}. \tag{3.4.2}$$

Caratheodory's existence theorem (e.g., [Hal09]), ensures that there exists a unique solution to the following initial value problem (IVP):

$$\underline{\chi}(t) := \underline{\phi}\big(t, \underline{x}_0, \underline{\sigma}(t)\big) := \underline{x}_0 + \int_{t_0=0}^{t} \underline{f}_\mathrm{c}\big(\underline{\chi}(\tau), \underline{\sigma}(\tau)\big)\,\mathrm{d}\tau. \tag{3.4.3}$$

Here, the state vector $\underline{x}_0$ is also used to initialize the continuous-time state trajectory with $\underline{\chi}(t_0) := \underline{x}_0$. With the presented setting, the discrete- and continuous-time state trajectories coincide at discrete time steps (e.g., [GP17, Thm. 2.7]):

$$\underline{\varphi}\big(k, \underline{x}_0, \mathbf{\underline{u}}\big) = \underline{\phi}\big(kT_\mathrm{s}, \underline{x}_0, \underline{\sigma}(t)\big),\ \forall\, k \in \mathbb{N}^{[0,N]}. \tag{3.4.4}$$

In practice, however, the IVP (3.4.3) cannot be solved exactly for arbitrary nonlinear systems. An approximate solution to the IVP (3.4.3) is obtained by applying iterative solution methods such as the Euler or Runge-Kutta method, as shown in, for example, [SB93, Sec. 7.2]. Within this dissertation, the integration approaches are always defined on a uniform and fixed time grid $t_{l+1} = t_l + \Delta t_\mathrm{s}$ with $t_0 = 0\,\mathrm{s}$, $l \in \mathbb{N}_0$, and the sampling time $\Delta t_\mathrm{s} \in \mathbb{R}^+$. Assume that each interval $T_\mathrm{s} = \gamma\Delta t_\mathrm{s}$ allows oversampling with the number of steps per interval $\gamma \in \mathbb{N}$. With piecewise constant controls, the Runge-Kutta integration kernel, for example, is given by (e.g., [SB93, Sec. 7.2]):

$$\begin{aligned}
&\underline{\Xi}\big(\underline{x}, \underline{\sigma}(t_l)\big) := \frac{1}{6}(\underline{\Xi}_1 + 2\,\underline{\Xi}_2 + 2\,\underline{\Xi}_3 + \underline{\Xi}_4),\ \underline{x} \in \mathbb{R}^p,\\
&\underline{\Xi}_1 := \underline{f}_\mathrm{c}\big(\underline{x}, \underline{\sigma}(t_l)\big),\ \underline{\Xi}_2 := \underline{f}_\mathrm{c}\big(\underline{x} + \frac{1}{2}\Delta t_\mathrm{s}\,\underline{\Xi}_1, \underline{\sigma}(t_l)\big),\\
&\underline{\Xi}_3 := \underline{f}_\mathrm{c}\big(\underline{x} + \frac{1}{2}\Delta t_\mathrm{s}\,\underline{\Xi}_2, \underline{\sigma}(t_l)\big),\ \underline{\Xi}_4 := \underline{f}_\mathrm{c}\big(\underline{x} + \Delta t_\mathrm{s}\,\underline{\Xi}_3, \underline{\sigma}(t_l)\big).
\end{aligned} \tag{3.4.5}$$

The application of iterative integration methods can also be considered as discretizing the continuous-time system. Consequently, the evolution of the discretized system results from the following recursion:

$$\underline{\phi}_{\Sigma}\big(t_l, \underline{x}_0, \underline{\sigma}(t)\big) := \begin{cases} \underline{x}_0 & \text{if } l = 0, \\ \underline{\phi}_{\Sigma}\big(t_{l-1}, \underline{x}_0, \underline{\sigma}(t)\big) + \Delta t_{\mathrm{s}}\ \Xi\Big(\underline{\phi}_{\Sigma}\big(t_{l-1}, \underline{x}_0, \underline{\sigma}(t)\big), \underline{\sigma}(t_{l-1})\Big) & \text{otherwise.} \end{cases} \tag{3.4.6}$$

For a fixed sampling time $\Delta t_{\mathrm{s}} > 0$, the discretization error is thus given by:

$$\|\underline{\phi}\big(t_{l+1}, \underline{x}_0, \underline{\sigma}(t)\big) - \underline{\phi}_{\Sigma}\big(t_{l+1}, \underline{x}_0, \underline{\sigma}(t)\big)\| \geq 0,\ \forall\, l > 0. \tag{3.4.7}$$

All numerical simulations in this dissertation involve the same state space representation and iterative integration method for the internal model and the plant, such that the definition $\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}}) := \underline{\phi}_{\Sigma}\big(kT_{\mathrm{s}}, \underline{x}_0, \sigma(t)\big)$ is used for all $k \in \mathbb{N}^{[0,N]}$. However, to capture the characteristic properties of the originally continuous-time benchmark systems, the sampling time Δt_{s} follows from a systematic identification procedure. The main idea is to determine the lowest exponent $i \in \mathbb{N}^{[0,12]}$ in $\Delta t_{\mathrm{s}} = 2^{-i}\,\mathrm{s}$ such that the resulting time performance does not exceed a specified error bound compared to a reference integration approach with $i = 12$ (see [Wor12, Sec. 1.3]). The sampling time $\Delta t_{\mathrm{s}} = 2^{-12}\,\mathrm{s}$ is sufficiently small to capture all dynamical effects of the selected benchmark systems. Appendix A.4 describes this procedure in more detail. Though numerical integration in this dissertation provides the values of the state trajectories on a fixed time grid for nonlinear systems, all figures show continuously connected graphs. Let $\underline{\hat{\chi}}(t) = \big(\hat{\chi}_1(t), \hat{\chi}_2(t), ..., \hat{\chi}_p(t)\big)^{\mathsf{T}}$ be the state trajectory that coincides with the state trajectory resulting from iterative integration with $\underline{\hat{\chi}}(t_l) := \underline{\phi}_{\Sigma}\big(t_l, \underline{x}_0, \underline{\sigma}(t)\big)$, $t_{l+1} = t_l + \Delta t_{\mathrm{s}}$, $t_l = 0\,\mathrm{s}$, and $l \in \mathbb{N}_0$. To distinguish between the open- and closed-loop control performances, the index μ is introduced to indicate the closed-loop system evolution as follows:

$$\begin{aligned} \underline{\sigma}_{\mu}(t) &= \big(\sigma_{\mu,1}(t), \sigma_{\mu,2}(t), ..., \sigma_{\mu,m}(t)\big)^{\mathsf{T}} := \underline{\mu}\big(\underline{x}_{\mu}(n)\big),\ \forall\, t \in \big[nT_{\mathrm{s}}, (n+1)T_{\mathrm{s}}\big),\ n \in \mathbb{N}_0, \\ \underline{\hat{\chi}}_{\mu}(t_l) &= \big(\hat{\chi}_{\mu,1}(t_l), \hat{\chi}_{\mu,2}(t_l), ..., \hat{\chi}_{\mu,p}(t_l)\big)^{\mathsf{T}} := \underline{\phi}_{\Sigma}\big(t_l, \underline{x}_{\mu,0}, \underline{\sigma}_{\mu}(t)\big),\ \forall\, l \in \mathbb{N}_0. \end{aligned} \tag{3.4.8}$$

In the case of a single-input system, the input is denoted by $\sigma_{\mu}(t) := \sigma_{\mu,1}(t)$. For visualization between discrete time steps t_l and t_{l+1}, all state variables with a hat are determined using interpolation.

Remark 3.4.1: *Linear systems.* Let the continuous- and discrete-time linear state space representations be given by $\underline{f}_{\mathrm{c}}\big(\underline{\chi}(t), \underline{\sigma}(t)\big) := \underline{A}_{\mathrm{c}}\underline{\chi}(t) + \underline{B}_{\mathrm{c}}\underline{\sigma}(t)$ and $\underline{f}\big(\underline{x}(k), \underline{u}(k)\big) := \underline{A}\,\underline{x}(k) + \underline{B}\,\underline{u}(k)$, respectively. With the chosen control parameterization (sample and zero-order hold), there exists an exact relationship between the individual domains (e.g., [Ack85; Lun16]):

$$\underline{A} = \mathrm{e}^{\underline{A}_{\mathrm{c}} T_{\mathrm{s}}}, \quad \underline{B} = \underline{A}_{\mathrm{c}}^{-1}(\underline{A} - \underline{I}_p)\underline{B}_{\mathrm{c}}. \tag{3.4.9}$$

Here, the matrix $\underline{A}_{\mathrm{c}}$ is assumed to be a non-singular square matrix. Therefore, $\underline{\hat{\chi}}(kT_{\mathrm{s}}) = \underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}})$ and $\underline{\hat{\chi}}_{\mu}(nT_{\mathrm{s}}) = \underline{\varphi}_{\mu}\big(n, \underline{x}_{\mu,0}\big)$ are defined for all $k, n \in \mathbb{N}_0$.

4 Single Degree of Freedom Model Predictive Control

This chapter introduces the basic formulation of SFMPC and examines the straightforward implementation and the resulting closed-loop properties. The initial analysis of SFMPC with uncountable control sets inspires the specific formulation of finite control sets and their evolution over time. Parts of this chapter have been published in [Mak+17; Mak+18d; Mak+18e; Mak+20].

4.1. Basic Formulation

The first step in SFMPC is to limit the degrees of freedom in control to a single degree. In this case, all vectors of the control sequence share the same value $\underline{u}_\mathrm{s} \in U$. Let $\underline{\Theta}^\mathrm{s}_N : U \mapsto U^N$ be the operator that reshapes a single vector into a sequence of equal vectors (e.g., [Cag+07]):

$$\big(\underline{u}(0) = \underline{u}_\mathrm{s}, \underline{u}(1) = \underline{u}_\mathrm{s}, ..., \underline{u}(N-1) = \underline{u}_\mathrm{s}\big) = \underline{\Theta}^\mathrm{s}_N(\underline{u}_\mathrm{s}) := \big(\underline{1}_N \otimes \underline{I}_m\big)\underline{u}_\mathrm{s}. \tag{4.1.1}$$

Here, $\otimes$ denotes the Kronecker product. The set of all admissible control sequences with a single degree of freedom is denoted by:

$$\mathcal{U}^\mathrm{s}_N(\underline{x}_0) := \{\underline{\mathbf{u}} \in \mathbb{U}^N \,|\, \underline{u}(k) = \underline{u}(0), \underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}}) \in \mathbb{X}, \forall\, k \in \mathbb{N}^{[0,N-1]}, \underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{u}}) \in \mathbb{X}_\mathrm{f}\}. \tag{4.1.2}$$

Since the admissible set $\mathcal{U}^\mathrm{s}_N(\underline{x}_0)$ only differs from $\mathcal{U}_N(\underline{x}_0)$ by imposing additional equality constraints on the control trajectory, $\mathcal{U}^\mathrm{s}_N(\underline{x}_0)$ is also closed and further compact because $\mathcal{U}^\mathrm{s}_N(\underline{x}_0) \subset \mathbb{U}^N$. The equality constraints select those control vectors which lie on a hyperline passing through the origin. Note that $J_N(\underline{x}_0, \cdot)$ and $\underline{\varphi}(k, \underline{x}_0, \cdot)$ are still continuous on the set $\mathcal{U}^\mathrm{s}_N(\underline{x}_0)$. The closed feasible state space contains all initial states from which the nonlinear system (3.1.1) can be steered to the terminal set $\mathbb{X}_\mathrm{f}$ at constant control:

$$\mathcal{X}^\mathrm{s}_N := \{\underline{x}_0 \in \mathbb{X} \,|\, \mathcal{U}^\mathrm{s}_N(\underline{x}_0) \neq \emptyset\}. \tag{4.1.3}$$

Input move-blocking yields the relations $\mathcal{U}^\mathrm{s}_N(\underline{x}_0) \subseteq \mathcal{U}_N(\underline{x}_0)$ and $\mathcal{X}^\mathrm{s}_N \subseteq \mathcal{X}_N$. Basic SFMPC builds upon the solution to the following OCP:

$$V^\mathrm{s}_N(\underline{x}_0) := \min_{\underline{u}_\mathrm{s} \in \mathbb{P}} J_N\big(\underline{x}_0, \underline{\Theta}^\mathrm{s}_N(\underline{u}_\mathrm{s})\big) \quad \text{subject to} \quad \underline{\Theta}^\mathrm{s}_N(\underline{u}_\mathrm{s}) \in \mathcal{U}^\mathrm{s}_N(\underline{x}_0). \tag{4.1.4}$$

Let the place holder set be defined by $\bar{\mathbb{P}} := \mathbb{U}$. Regardless of the definition of the place holder set, $\underline{u}_{\mathrm{s}}^*(\underline{x}_0)$ denotes the optimal solution to OCP (4.1.4) and generates the control sequence $\underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0) := \big(\underline{u}^*(0, \underline{x}_0) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0), \underline{u}^*(1, \underline{x}_0) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0), ..., \underline{u}^*(N-1, \underline{x}_0) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0)\big) = \underline{\Theta}_N^{\mathrm{s}}\big(\underline{u}_{\mathrm{s}}^*(\underline{x}_0)\big) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0)$. Note that for solving OCP (4.1.4) only a single control vector $\underline{u}_{\mathrm{s}}^*(\underline{x}_0) \in \mathbb{U}$ has to be determined. In contrast, in OCP (3.1.6) an entire control sequence $\underline{\mathbf{u}} \in \mathbb{U}^N$ is subject to optimization. The SFMPC configuration obviously reduces the computational effort and allows straightforward combinatorial optimization in Section 4.4. However, limiting the degrees of freedom in control has a detrimental effect on theoretical closed-loop properties.

4.2. Stability Analysis

SFMPC suffers from the same problem as input move-blocking MPC with a fixed blocking pattern [Cag+07]. For recursive feasibility and the cost decent property to hold, the following control sequence has to be an element of $\mathcal{U}_N^{\mathrm{s}}(\underline{x}_0^+)$ (see Asm. 3.3.2):

$$\underline{\Omega}_{\mathrm{sta}}\big(\underline{x}_0, \underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0)\big) = \Big(\underline{u}^*(1, \underline{x}_0) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0), \underline{u}^*(2, \underline{x}_0) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0), ..., \underline{u}^*(N-1, \underline{x}_0) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0), \underline{\kappa}\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0))\big)\Big). \tag{4.2.1}$$

Obviously, if $\underline{\kappa}\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0))\big) \neq \underline{u}_{\mathrm{s}}^*(\underline{x}_0)$, another degree of freedom in control is required such that $\underline{\Omega}_{\mathrm{sta}}\big(\underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0)\big) \notin \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0^+)$. In addition, with input move-blocking, the monotonicity property of the cost function does not apply in general. Here, the inequality $0 \leq V_{N+1}^{\mathrm{s}}(\underline{x}_0) \leq V_N^{\mathrm{s}}(\underline{x}_0) \leq F(\underline{x}_0) \leq \alpha_{\mathrm{f}}(\|\underline{x}_0\|)$ does not hold for all $\underline{x}_0 \in \mathbb{X}_{\mathrm{f}}$. However, this relationship is essential for proving continuity of the cost function $V_N^{\mathrm{s}}(\cdot)$ at the origin (see Sec. 3.3). Nevertheless, there exist three special system configurations for which theoretical closed-loop properties even hold with basic SFMPC. The following first two cases are rather theoretical and are required for the systematic analysis of SFMPC in the remainder of this dissertation. The third configuration, in turn, is directly relevant for practical applications.

Special Case I: One-Step Horizon for Systems with Input and State Constraints $(N = 1)$

If the horizon is set to $N = 1$, SFMPC reproduces conventional MPC.

Proposition 4.2.1: *SFMPC reproduces conventional MPC.* Suppose Assumptions 3.1.1-3.1.3 and 3.3.1-3.3.2 hold. Define $\bar{\mathbb{P}} := \mathbb{U}$ and the implicit control law $\underline{\mu}(\underline{x}_0) := \underline{u}_{\mathrm{s}}^*$ for all $\underline{x}_0 \in \mathcal{X}_1^{\mathrm{s}}$. Then, the optimal cost function $V_1^{\mathrm{s}}(\cdot)$ in (4.1.4) represents a Lyapunov function in the set $\mathcal{X}_1^{\mathrm{s}}$ for the closed-loop system (3.3.2). Therefore, the origin is asymptotically stable in the positive invariant set $\mathcal{X}_1^{\mathrm{s}}$ for the closed-loop system (3.3.2).

Proposition 4.2.1 is stated without proof since input move-blocking only applies for $N > 1$. This implies that $V_1^{\mathrm{s}}(\underline{x}_0) = V_1(\underline{x}_0)$ holds for all $\underline{x}_0 \in \mathcal{X}_1^{\mathrm{s}} = \mathcal{X}_1$. The corresponding proof is summarized below Theorem 3.3.2. Note that the monotonicity property applies for a horizon of $N = 1$ such that $0 \leq V_1^s(\underline{x}_0) \leq F(\underline{x}_0) \leq \alpha_f(\|\underline{x}_0\|)$ holds for all $\underline{x}_0 \in \mathbb{X}_f$. The feasible set includes, in addition to the terminal set $\mathbb{X}_f$, all initial states from which the nonlinear system (3.1.1) can be steered to the terminal set by a single control action. Thus, $\underline{f}(\underline{x}_0, \underline{\mu}(\underline{x}_0)) \in \mathbb{X}_f$ applies for all $\underline{x}_0 \in \mathcal{X}_1^s = \mathcal{X}_1$.

Special Case II: Linear Systems without Input and State Constraints $(N = 1)$

The second configuration follows up the first case with $N = 1$, however, does not comprise any constraints such that the terminal set $\text{lev}_\pi F$ can be chosen arbitrarily large with $\pi > 0$ [Raw+20, Sec. 2.5.1].

Proposition 4.2.2: *SFMPC reproduces LQR.* Assume that the horizon is set to $N = 1$, there are no constraints such that the relations $\mathbb{U} = U$ and $\mathbb{X} = \mathbb{X}_f = X$ apply, and the linear system $\underline{x}^+ = \underline{f}(\underline{x}, \underline{u}) = \underline{A}\,\underline{x} + \underline{B}\,\underline{u}$ with $\underline{x} \in X$ and $\underline{u} \in U$ is stabilizable. Consider quadratic form cost functions according to (3.1.4) with positive definite weighting matrices $\underline{Q}$ and $\underline{R}$. The terminal cost function $F(\underline{x}) = \underline{x}^\mathsf{T}\underline{P}\,\underline{x}$ relies on the solution $\underline{P}$ to the Riccati equation (3.3.18). Then, the implicit control law $\underline{\mu}(\underline{x}_0) := \underline{u}_s^*(\underline{x}_0)$, which is driven by the solutions to OCP (4.1.4) with $\bar{\mathbb{P}} := \mathbb{U}$, renders the origin globally exponentially stable for the closed-loop system (3.3.2).

This proposition is also stated without proof since with $N = 1$, SFMPC reproduces conventional MPC (see Prop. 4.2.1). Though, note that the unconstrained case does not satisfy Assumption 3.1.3 since U is not compact. However, there exists an alternative derivation that is adopted from [Raw+20, Prop. 2.4]. The linear transition map $\underline{f}(\underline{x}, \underline{u}) = \underline{A}\,\underline{x} + \underline{B}\,\underline{u}$ and the quadratic form functions $\ell(\cdot)$ and $F(\cdot)$ are continuous. It follows that the composite function $J_1(\cdot)$ is also continuous. Therefore, the set $\mathcal{U}(\underline{x}_0) := \{\underline{u}_s \in U \mid J_1(\underline{x}_0, \underline{u}_s) \leq \pi\}$ is closed for arbitrary $\pi \in \mathbb{R}^+$ and all $\underline{x}_0 \in X$. Since $J_1(\underline{x}_0, \underline{u}_s) \to \infty$ if $\|\underline{u}_s\| \to \infty$, $J_1(\underline{x}_0, \cdot)$ is coercive such that the set $\mathcal{U}(\underline{x}_0)$ is further bounded. For properties of coercive functions refer, for example, to [Per+88, Sec. 1.4]. The extreme value theorem applies on the compact set $\mathcal{U}(\underline{x}_0)$. Consequently, there exists a solution to OCP (4.1.4) with $N = 1$ and $\bar{\mathbb{P}} = \mathbb{U} = U$ although U is unbounded [Raw+20, Prop. 2.4]. By (3.3.13), the terminal cost function $F(\underline{x}_0) = \underline{x}_0^\mathsf{T}\underline{P}\,\underline{x}_0$ determines the quadratic form costs over an infinite horizon for the linear system $\underline{x}^+ = (\underline{A} + \underline{B}\,\underline{K})\underline{x} = \underline{A}_K\underline{x}$ starting at $\underline{x}_0$. Based on the principle of optimality, it follows that $F(\underline{x}_0) = V_1^s(\underline{x}_0) = \ell(\underline{x}_0, \underline{K}\,\underline{x}_0) + F(\underline{A}_K\underline{x}_0)$ holds for all $\underline{x}_0 \in X$. The solution to OCP (4.1.4) with $N = 1$ is thus given by $\underline{u}_s^*(\underline{x}_0) = \mathbf{u}_s^*(\underline{x}_0) = \underline{K}\,\underline{x}_0$ for all $\underline{x}_0 \in X$. Since the terminal cost function $F(\cdot)$ is a Lyapunov function in X, global exponential stability follows from Theorem 3.3.1 with $\alpha_1(\|\underline{x}_0\|) = \alpha_3(\|\underline{x}_0\|) = \alpha_\ell(\|\underline{x}_0\|) = b_1\|\underline{x}_0\|^2_{\underline{Q}_K}, b_1 \in (0, 1]$, and $\alpha_2(\|\underline{x}_0\|) = \alpha_f(\|\underline{x}_0\|) = b_2\|\underline{x}_0\|^2_{\underline{P}}, b_2 \geq 1$, for all $\underline{x}_0 \in X$.

Special Case III: Open-Loop Stable Linear System with Input Constraints $(N \geq 1)$

If the system of interest is linear, open-loop stable, subject to quadratic form cost functions with positive definite weighting matrices $\underline{Q}$ and $\underline{R}$, and only subject to input constraints, stabilizing closed-loop properties can be derived from defining $\underline{\kappa}(\cdot) := \underline{0}_m$ and $F(\underline{x}) := \underline{x}^\mathsf{T}\underline{P}\,\underline{x}$, where $\underline{P}$ is the solution to the Lyapunov equation

$\underline{P} = \underline{A}^\intercal \underline{P}\,\underline{A} + \underline{Q}$ [Raw+20, Sec. 2.5.3] (see also [RM93]). The major modification to the previous Special Case II is that the terminal cost function now determines the quadratic form costs over an infinite horizon for the autonomous system $\underline{x}^+ = \underline{A}\,\underline{x}$ with $\underline{K}\,\underline{x} = \underline{0}_m$ for all $\underline{x} \in X$. If the input constraint set $\mathbb{U}$ is polyhedral, the admissible control set $\mathcal{U}_N(\underline{x})$ is again compact for all $\underline{x} \in X$ as in the Special Case I. The following proposition adapts the findings in [Raw+20, Sec. 2.5.3] to SFMPC with $N \geq 1$.

Proposition 4.2.3: *SFMPC for input constrained open-loop stable linear systems.* Consider an open-loop stable system $\underline{x}^+ = \underline{f}(\underline{x}, \underline{u}) = \underline{A}\,\underline{x} + \underline{B}\,\underline{u}$ with $\underline{x} \in X$ and $\underline{u} \in U$ such that all eigenvalues of the matrix $\underline{A}$ lie strictly inside the unit circle. Assume that there are no state constraints such that $\mathbb{X} = \mathbb{X}_\mathrm{f} = X$ holds. Let the input constraint set $\mathbb{U}$ be defined by a convex polytope that contains the origin according to (3.2.3). Consider quadratic form cost functions with positive definite weighting matrices $\underline{Q}$ and $\underline{R}$ as defined in (3.1.4). The terminal cost function relies on the solution to the Lyapunov equation $\underline{A}^\intercal \underline{P}\,\underline{A} + \underline{Q} = \underline{P}$. Assume that the set of admissible control sequences is given by:

$$\mathcal{U}_N^\mathrm{s}(\underline{x}_0) := \{\underline{\mathbf{u}} \in \mathbb{U}^N \mid \underline{u}(k) = \underline{u}(0), \forall\, k \in \mathbb{N}^{[1,N-1]}, F\big(\underline{A}\,\underline{x}_0 + \underline{B}\,\underline{u}(0)\big) \leq F(\underline{A}\,\underline{x}_0)\}. \tag{4.2.2}$$

Then, the implicit control law $\underline{\mu}(\underline{x}_0) := \underline{u}_\mathrm{s}^*(\underline{x}_0)$, which is driven by the solutions to OCP (4.1.4) with $\bar{\mathbb{P}} := \mathbb{U}$, renders the origin globally exponentially stable for the closed-loop system (3.3.2).

Proof. The convex polytope is compact and contains $\underline{u}_\mathrm{s} = \underline{0}_m \in \mathbb{U}$. The linear transition map $\underline{f}(\underline{x}, \underline{u}) := \underline{A}\,\underline{x} + \underline{B}\,\underline{u}$ and the quadratic form functions $\ell(\cdot)$ and $F(\cdot)$ are continuous. The constraint $F\big(\underline{A}\,\underline{x}_0 + \underline{B}\,\underline{u}(0)\big) \leq F(\underline{A}\,\underline{x}_0)$ is defined by a non-strict ($\leq$) inequality. Therefore, the admissible input set $\mathcal{U}_N^\mathrm{s}(\underline{x}_0) \subset \mathbb{U}^N$ is closed and bounded and, thus, compact. The composite function $J_N(\underline{x}_0, \cdot)$ is continuous on the compact set $\mathcal{U}_N^\mathrm{s}(\underline{x}_0)$ (see [Raw+20, Prop. 2.4 (a)-(b)]). By the Weierstrass extreme value theorem, there exists a solution to OCP (4.1.4) for all $\underline{x}_0 \in \mathcal{X}_N^\mathrm{s}$ (see [Raw+20, Prop. 2.4 (c)]). The optimizer can selectively disable input move-blocking by falling back to $\underline{\mathbf{u}}_\mathrm{s}^*(\underline{x}_0) = \underline{0}_{mN} \in \mathcal{U}_N^\mathrm{s}(\underline{x}_0)$ for arbitrary $N \in \mathbb{N}$. This implies that $\mathcal{X}_N^\mathrm{s} = X$ holds, ensuring recursive feasibility for all initial states. Quadratic form cost functions satisfy the Assumption 3.3.1 on the existence of suitable comparison functions. As discussed in the introduction of Section 4.2, $V_N^\mathrm{s}(\cdot)$ does not satisfy the descent property between two closed-loop time steps. Therefore, consider the terminal cost function $F(\cdot)$, which is bounded as follows for all $\underline{x}_0 \in X$:

$$\alpha_\ell(\|\underline{x}_0\|) \leq \ell(\underline{x}_0, \underline{0}_m) \leq J_N(\underline{x}_0, \underline{0}_{mN}) = F(\underline{x}_0) \leq \alpha_\mathrm{f}(\|\underline{x}_0\|). \tag{4.2.3}$$

To serve as a Lyapunov function, the terminal cost function $F(\cdot)$ needs to satisfy the following descent property for all $\underline{x}_0 \in X$:

$$F(\underline{x}_0^+) = F(\underline{A}\,\underline{x}_0 + \underline{B}\,\underline{u}_\mathrm{s}^*(\underline{x}_0)) \overset{!}{\leq} F(\underline{A}\,\underline{x}_0) = F(\underline{x}_0) - \ell(\underline{x}_0, \underline{0}_m) \leq F(\underline{x}_0) - \alpha_\ell(\|\underline{x}_0\|). \tag{4.2.4}$$

Case I: $N = 1$. Quadratic cost functions imply that $\ell\big(\underline{x}_0, \underline{u}_\mathrm{s}^*(\underline{x}_0)\big) \geq \ell(\underline{x}_0, \underline{0}_m)$ holds for all $\underline{x}_0 \in X$. Since $J_1(\underline{x}_0, \underline{u}_\mathrm{s}) = \ell(\underline{x}_0, \underline{u}_\mathrm{s}) + F(\underline{A}\,\underline{x}_0 + \underline{B}\,\underline{u}_\mathrm{s})$, it follows that $F\big(\underline{A}\,\underline{x}_0 +$

$\underline{B}\underline{u}_s^*(\underline{x}_0)) \leq F(\underline{A}\,\underline{x}_0)$ holds inherently. Case II: $N > 1$. In this case, a clear relation between $F\big(\underline{\varphi}(N, \underline{x}_0, \underline{\mathbf{u}}_s^*(\underline{x}_0))\big)$ and $F\big(\underline{\varphi}(N, \underline{x}_0, \underline{0}_{mN})\big)$ does not exist. However, the additional constraint $F(\underline{A}\,\underline{x}_0 + \underline{B}\underline{u}_s) \leq F(\underline{A}\,\underline{x}_0)$ in the definition of $\mathcal{U}_N(\underline{x}_0)$ ensures that $F\big(\underline{A}\,\underline{x}_0 + \underline{B}\underline{u}_s^*(\underline{x}_0)\big) \leq F(\underline{A}\,\underline{x}_0)$ holds for all $\underline{x}_0 \in X$. Recall that $\underline{0}_{mN} \in \mathcal{U}_N^s(\underline{x}_0)$ satisfies this additional constraint for all $\underline{x}_0 \in X$. The optimizer either finds a better constant control sequence or simply resorts to $\underline{0}_{mN}$. Since $\alpha_\ell(\cdot), \alpha_f(\cdot) \in \mathcal{K}_\infty$, $F(\cdot)$ is a Lyapunov function in X, global exponential stability follows from Theorem 3.3.1 with $\alpha_1(\|\underline{x}_0\|) = \alpha_3(\|\underline{x}_0\|) = \alpha_\ell(\|\underline{x}_0\|) = b_1\|\underline{x}_0\|_{\underline{Q}}^2, b_1 \in (0, 1]$, and $\alpha_2(\|\underline{x}_0\|) = \alpha_f(\|\underline{x}_0\|) = b_2\|\underline{x}_0\|_{\underline{P}}^2, b_2 \geq 1$, for all $\underline{x}_0 \in X$. □

4.3. Example: Mass-Spring-Damper System

The second-order linear differential equation $M_{sd}\,\ddot{y}(t) - D_{sd}\,\dot{y}(t) - C_{sd}\,y(t) = \sigma(t)$ describes the well-known mass-spring-damper system with the output $y : \mathbb{R}_0^+ \mapsto \mathbb{R}$ and the input $\sigma : \mathbb{R}_0^+ \mapsto \mathbb{R}$. In this numerical example, the mass $M_{sd} = 10$, the spring and damping constants $C_{sd} = 35$ and $D_{sd} = 12$, respectively, are considered without units. The differential equation is transformed into the following state space representation with $\underline{\chi}(t) = \big(\chi_1(t) = y(t), \chi_2(t) = \dot{y}(t)\big)^\intercal$:

$$\begin{pmatrix}\dot{\chi}_1(t)\\ \dot{\chi}_2(t)\end{pmatrix} = \underbrace{\begin{pmatrix}0 & 1\\ -C_{sd}/M_{sd} & -D_{sd}/M_{sd}\end{pmatrix}}_{=\underline{A}_c}\begin{pmatrix}\chi_1(t)\\ \chi_2(t)\end{pmatrix} + \underbrace{\begin{pmatrix}0\\ 1/M_{sd}\end{pmatrix}}_{=\underline{B}_c}\sigma(t). \qquad (4.3.1)$$

When implementing piecewise constant controls with the step time $T_s = 2^{-5}\,$s, the continuous-time representation (4.3.1) can be transformed into the discrete-time representation $\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{B}u(k)$, and vice versa, based on (3.4.9). Hence, control actions generated in the discrete-time domain are translated into the continuous-time domain using a zero-order hold element. The eigenvalues of the resulting matrix $\underline{A}$ lie strictly inside the unit circle. Both MPC and SFMPC implement quadratic form cost functions as defined in (3.1.4), considering the following configuration:

$$\underline{Q} = \mathrm{diag}(1, 0.2),\ \underline{R} = 0,\ \underline{A}^\intercal\underline{P}\,\underline{A} + \underline{Q} = \underline{P},\ N = 10,\ |u(k)| \leq 15. \qquad (4.3.2)$$

Note that $\mathbb{X} = \mathbb{X}_f = X$ applies for both control concepts. The chosen Lyapunov function does not impose any requirements on the definiteness of the matrix $\underline{R}$. In this example, the principle of recursive elimination is used to generate the corresponding NLPs for MPC and SFMPC. Figure 4.1 shows the open-loop system evolution and the closed-loop control performances of MPC and SFMPC in the continuous-time domain, starting from two different initial states $\underline{x}_{\mu,0} = (-0.8, 0)^\intercal$ (solid lines) and $\underline{x}_{\mu,0} = (-0.5, 0.8)^\intercal$ (dashed lines). As established in the Special Case III, both MPC and SFMPC stabilize the origin with the open-loop stable system. Notice that with $\underline{\mu}\big(\underline{x}_\mu(n)\big) := 0$ for all $n \in \mathbb{N}_0$, the open-loop system evolution can be considered as a closed-loop system evolution. The subplots in the left plot of Figure 4.1 show that the costs are decreasing in the sense of Lyapunov for all three cases. Here, the cost evolutions are based on the evaluation of $\underline{x}_\mu^\intercal(n)\,\underline{P}\,\underline{x}_\mu(n)$ at each time instant

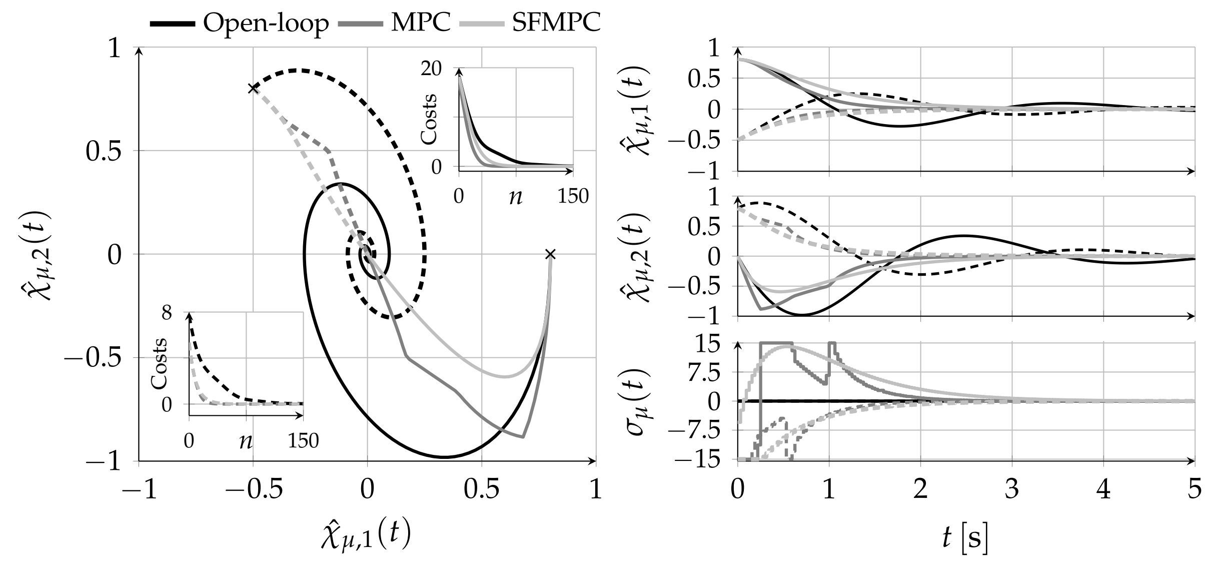


Figure 4.1.: Open- and closed-loop control of the mass-spring-damper system (open-loop stable). Left: Phase portrait. System evolutions starting from two different initial states represented by a cross. The subplots show the corresponding cost evolutions. Right: Closed-loop control performance over time for different predictive controllers. Note: Solid and dashed lines are used to distinguish between system evolutions starting from different initial states.

$n \in \mathbb{N}^{[0,150]}$. For visualization, the cost graphs are connected continuously. Compared to the open-loop system evolution, both MPC and SFMPC suppress oscillation and enable fast transitions to the origin in this particular example. Figure 4.1 highlights the suppression of oscillation in the state space plot on the left and the time plots on the right. Therefore, both approaches reduce costs significantly faster than the controller with $\underline{\mu}(\cdot) := 0$ (open-loop behavior). Though MPC results in a non-smooth closed-loop control performance (since $\underline{R} = 0$), the costs decrease slightly faster compared to SFMPC. Figure 4.2 serves as a guide for choosing a suitable horizon length for the mass-spring-damper system with the presented configuration. Throughout this dissertation, the normalized closed-loop control performance criterion $\bar{J}_{\mathrm{cl}}$ is defined by:

$$\bar{J}_{\mathrm{cl}} := \left(1 - \frac{\left[\sum_{n=0}^{l-1}\left(\|\underline{x}_\mu(n)\|^2_{\underline{Q}} + \|\underline{\mu}(\underline{x}_\mu(n))\|^2_{\underline{R}}\right)\right]_{\mathrm{MPC/SFMPC}}}{\left[\sum_{k=0}^{l-1}\left(\|\underline{\varphi}(k, \underline{x}_\mu(0), \underline{\mathbf{u}}_{\mathrm{ref}})\|^2_{\underline{Q}} + \|\underline{u}_{\mathrm{ref}}(k)\|^2_{\underline{R}}\right)\right]_{\mathrm{Open\text{-}loop}}}\right) 100\,\%. \qquad (4.3.3)$$

Here, the definition $\underline{\mathbf{u}}_{\mathrm{ref}} := \left(\underline{u}_{\mathrm{ref}}(0), \underline{u}_{\mathrm{ref}}(1), ..., \underline{u}_{\mathrm{ref}}(l-1)\right) \in \mathbb{U}^l$ is introduced. In the current mass-spring-damper example, the upper evaluation limit is $l = 300$ and the open-loop system evolution serves as the reference performance such that $\underline{\mathbf{u}}_{\mathrm{ref}} = \underline{0}_{ml}$ applies. Therefore, Figure 4.2 shows the improvement of the system transition to the origin with MPC and SFMPC compared to the oscillating open-loop system evolution with $\bar{J}_{\mathrm{cl}} \in [0\,\%, 100\,\%)$. For MPC, the closed-loop control performance first increases as the horizon length increases. However, the performance saturates since the open-loop and closed-loop trajectories converge towards each other as the horizon increases. Note that the OCP formulations (3.1.6) and (4.1.4) implement a similar quadratic form

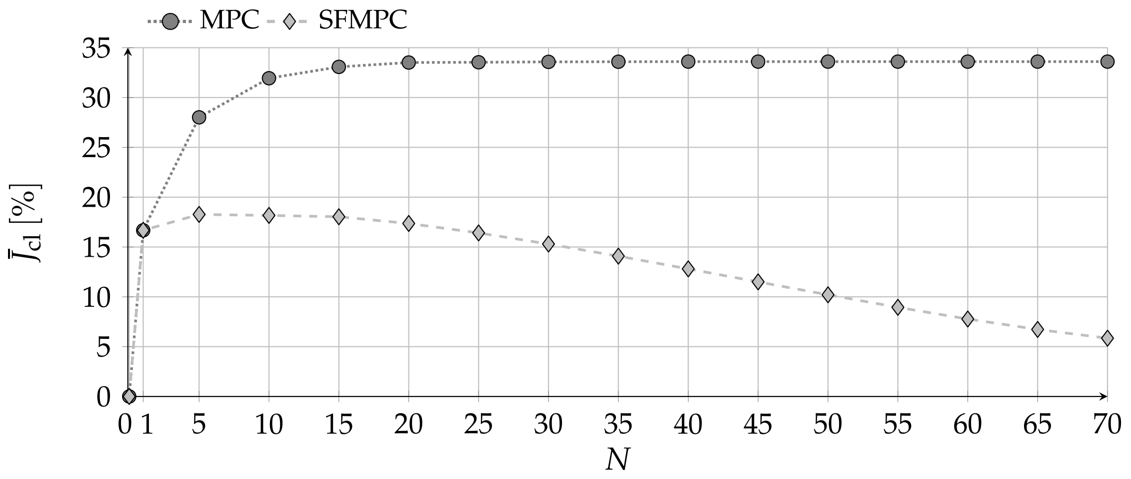


Figure 4.2.: Evaluation of the closed-loop control performance metric $\bar{J}_{\text{cl}}$ for MPC and SFMPC over horizon length N. The open-loop stable mass-spring-damper system is initialized at $\underline{x}_{\mu,0} = (-0.8, 0)^\mathsf{T}$ and is to be transferred to the origin. The closed-loop control performances are normalized to the open-loop system evolution, which is here indicated by $N = 0$.

cost function as it is used in the numerator or denominator of (4.3.3). In the case of SFMPC, the improvement of the closed-loop control performance reaches a maximum at approximately $N = 5$. Beyond that point, the closed-loop control performance decreases again and converges to $\bar{J}_{\text{cl}} \to 0\,\%$ as $N \to \infty$. Since SFMPC has only a single degree of freedom in control, the open-loop and closed-loop trajectories diverge from each other as the horizon length increases. Note that for a large horizon, only $\mathbf{\underline{u}}_{\text{s}}^*(\underline{x}_0) = \underline{0}_{mN}$ satisfies the stability constraint $F\big(\underline{A}\,\underline{x}_0 + \underline{B}\underline{u}_{\text{s}}^*(\underline{x}_0)\big) \leq F(\underline{A}\,\underline{x}_0)$ and transfers (open-loop) the system exactly to the origin. In summary, it can be stated that choosing $N > 1$ in case of SFMPC can increase the control performance. However, due to the extreme input move-blocking, there exists an upper bound beyond which the closed-loop performance deteriorates. This design guide will be important in the rest of this dissertation, especially for state constrained systems.

4.4. Input Domain Discretization

In a high-dimensional Euclidean space, derivative-based optimization algorithms are remarkably efficient since they include local directional information. Newton-type solvers, for example, generate increments based on the local first- and second-order derivative information and seek for points that satisfy the KKT conditions, see, for example, [NW06]. The basic SFMPC formulation presented in Section 4.1 reduces the parameter space to a single control vector $\underline{u}_{\text{s}}$ and thus eliminates the need to determine derivative information. With such a low dimensional parameter space, it suffices to generate and evaluate control samples in the sense of a sampling-based optimization. The main idea in SFMPC is to first place samples in the constraint input set $\mathbb{U}$ by implementing a suitable discretization method. This procedure forms the input sample space. Then, a suitable search algorithm seeks the optimal sample out of the finite set of control vectors. In this dissertation, the exhaustive search algorithm simply

evaluates all resulting control candidates. In principle, there exist two possibilities on how to integrate the sample space into the system theoretical context. Suppose that sampling is an attribute of the optimization algorithm. Then, closed-loop properties can only be ensured if the sample space contains, at least, the optimal solution $\underline{u}_{\mathrm{s}}^*(\underline{x}_0)$ to OCP (4.1.4) with $\bar{\mathbb{P}} := \mathbb{U}$ for all $\underline{x}_0 \in \mathcal{X}_N^{\mathrm{s}}$. Recall that the stability analyses in Section 3.3 and Section 4.2 are based on the evolution of the optimal cost function. Since there is no guarantee that an optimization algorithm (either sampling-based or smooth) can determine the global optimum, unless the OCP is convex, stability analysis needs to build on stabilizing warm-starts in the framework of suboptimal MPC [Sco+99; Pan+11; All+17; Raw+20; BL17]. Another possibility is to include the sample space into the formulation of OCP (4.1.4) by redefining the place holder set $\bar{\mathbb{P}}$. Under mild assumptions, this procedure facilitates stability analysis. Assume that the combinatorial optimizer is fast enough to evaluate all control candidates and thus always determines the global optimum of the sample space. This section pursues the latter option and formalizes the sample space in the following.
A finite set of control vectors $\mathbb{D} \subset \mathbb{U}$ is compact by its definition and has a finite cardinality $c = |\mathbb{D}| \in \mathbb{N}$. To generate a finite set, a uniform discretization can be applied along each dimension $j \in \mathbb{N}^{[1,m]}$ with a fixed step width $\Delta u_j \in \mathbb{R}^+$:

$$\mathbb{D}_j := \{u_{\min,j}, u_{\min,j} + \Delta u_j, u_{\min,j} + 2\Delta u_j, ..., u_{\max,j}\},\ \forall j \in \mathbb{N}^{[1,m]}. \tag{4.4.1}$$

Here, the finite set $\mathbb{D}_j$ has the cardinality $c_j = |\mathbb{D}_j| \leq c$. The finite set of control vectors $\mathbb{D}$ then results from combining all dimensions:

$$\mathbb{D} := \mathbb{D}_1 \times \mathbb{D}_2 \times ... \times \mathbb{D}_m \subset \mathbb{U}. \tag{4.4.2}$$

The definition of $\bar{\mathbb{P}} := \mathbb{D}$ in OCP (4.1.4) now allows to implement combinatorial optimization as, for example, the exhaustive search. It shall be noted that SFMPC processes the control candidates starting from $\underline{u}_{\min} = \min \mathbb{U}$. If the solution $\underline{u}_{\mathrm{s}}^*(\underline{x}_0)$ is not unique, SFMPC selects the value that is closer to $\underline{u}_{\min}$. However, as it can be seen from (4.4.2), the combinatorial complexity grows exponentially with the input dimension m. The cardinality of the finite set of control vectors is $c = |\mathbb{D}| = \prod_{j=1}^{m} |\mathbb{D}_j|$. Therefore, SFMPC addresses systems with preferably not more than two inputs, in particular single-input systems. This requirement appears to be very restrictive, however, underlying systems such as electromagnetic actuators often have a single input. Even if there exists a solution to OCP (4.1.4) with $\bar{\mathbb{P}} := \mathbb{D}$, it might be inferior compared to the case when $\bar{\mathbb{P}} := \mathbb{U}$. A straightforward approach to increase the control performance is to decrease the discretization step widths Δu_j for all $j \in \mathbb{N}^{[1,m]}$. However, decreasing the step widths, in turn, increases the combinatorial complexity. Parallel computing counters the combinatorial complexity since the finite set formulation permits parallelization in the evaluation of the individual control candidates $\underline{u}_{\mathrm{s}} \in \mathbb{D}$. However, the application of parallelization imposes special requirements on the low-level hardware, such as the usage of FPGAs (see, e.g., [Joo+11]), or imposes a computational overhead for efficient memory management if a conventional processing architecture is employed. In this dissertation, an adaptive and sparse discretization method is used to increase the control performance while allowing efficient serial processing of time-varying finite control sets.

4.5. Time-Varying Finite Control Sets

This section introduces a generic formulation of time-varying finite control sets, designed to always contain the control candidates required to ensure stabilizing closed-loop properties. The basic idea is to embed the current solution at time instant n into the formulation of the next OCP at time instant $n+1$ for all $n \in \mathbb{N}_0$. Let the placeholder set $\mathbb{P}(\underline{x}_0, n) \subset \mathbb{U}$ be the only OCP ingredient that depends, in addition to the initial state $\underline{x}_0$, on the integer closed-loop time n. However, then OCP (4.1.4) becomes time-varying, leading to the following formulation:

$$V_N^{\mathrm{s}}(\underline{x}_0, n) := \min_{\underline{u}_{\mathrm{s}} \in \mathbb{P}(\underline{x}_0, n)} J_N\big(\underline{x}_0, \underline{\Theta}_N^{\mathrm{s}}(\underline{u}_{\mathrm{s}})\big) \quad \text{subject to} \quad \underline{\Theta}_N^{\mathrm{s}}(\underline{u}_{\mathrm{s}}) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0). \tag{4.5.1}$$

The optimal solution is now denoted by $\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)$ and yields the control sequence $\underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0, n) := \big(\underline{u}^*(0, \underline{x}_0, n) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n), \underline{u}^*(1, \underline{x}_0, n) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n), ..., \underline{u}^*(N-1, \underline{x}_0, n) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)\big) = \underline{\Theta}_N^{\mathrm{s}}\big(\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)\big) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0)$. Since the system dynamics and the cost functions are still time-invariant, the optimal control sequence always starts at $k=0$. However, the optimal values $\underline{u}^*(k, \underline{x}_0, n)$ with $k \in \mathbb{N}^{[0,N-1]}$ now also depend on the closed-loop time n since their corresponding value set $\mathbb{P}(\underline{x}_0,\, n)$ is time-varying. The implicit control law is now given by $\underline{\mu}(\underline{x}_0, n) := \underline{u}^*(0, \underline{x}_0, n) = \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)$ and the closed-loop system evolves as follows:

$$\underline{x}_0^+ := \underline{x}_\mu(n+1) = \underline{f}\Big(\underline{x}_\mu(n), \underline{\mu}(\underline{x}_\mu(n), n)\Big), \underline{x}_\mu(0) := \underline{x}_{\mu,0}. \tag{4.5.2}$$

Exponential Adaptive Domain Discretization

The following discretization method adopts the basic idea that was proposed in [Kel17] during the development of an emergency steering assist. Here, the steering angle is discretized densely in the vicinity of the last applied steering command. This strategy enables smooth vehicle guidance, at least if obstacles are detected at an early stage. The motivation for adaptive discretization is to facilitate smooth closed-loop control when the evolution of the closed-loop system from $\underline{x}_\mu(n)$ to $\underline{x}_\mu(n+1)$ does not impose a large change on the input. Otherwise, at least the maximum and minimum control vectors max $\mathbb{U}$ and min $\mathbb{U}$, respectively, should always be evaluated to ensure dynamic system behavior when required.
The following map $\eta : \mathbb{Z} \times \mathbb{O} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ describes an exponential discretization and is defined by:

$$\eta(i, c, u, u_{\max}, u_{\min}) := \begin{cases} u + 10^{q\left(\frac{-2i}{c-1} - 1\right)}\left(u_{\min} - u\right) & \text{if } i < 0, \\ u & \text{else if } i = 0, \\ u + 10^{q\left(\frac{2i}{c-1} - 1\right)}\left(u_{\max} - u\right) & \text{otherwise.} \end{cases} \tag{4.5.3}$$

Here, the relation $u_{\min} \leq u \leq u_{\max}$ applies. The larger the exponent $q \in \mathbb{N}$, the finer the sampling is in the vicinity of u. Let $\mathbb{A}(n) := \mathbb{A}_1(n) \times \mathbb{A}_2(n) \times .... \times \mathbb{A}_m(n) \subset \mathbb{U}$ be a time-variant finite set of control vectors with the constant cardinality $c = |\mathbb{A}(n)| = \prod_{j=1}^m c_j = \prod_{j=1}^m |\mathbb{A}_j(n)|$. Let the definition $\mathbb{A}(0) := \mathbb{D}$ hold at time instant $n=0$. For

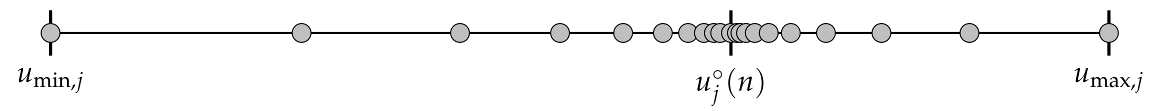


Figure 4.3.: Location of control samples $u_j \in \mathbb{A}_j(n+1)$ on the interval $[u_{\min,j}, u_{\max,j}]$ after applying exponential discretization with $q = 2$ and $c_j = 21$.

$n > 0$, the finite control sets are linked to the temporal evolution of the closed-loop system (4.5.2). The finite control set for the j-th input dimension at the next time instant $n+1$ with $\underline{u}^\circ(n) = \big(u_1^\circ(n), u_2^\circ(n), ..., u_m^\circ(n)\big) := \underline{\mu}(\underline{x}_\mu(n), n)$ is thus given by:

$$\mathbb{A}_j(n+1) := \bigcup_{i=-(c_j-1)/2}^{(c_j-1)/2} \Big\{\eta\big(i, c_j, u_j^\circ(n), u_{\max,j}, u_{\min,j}\big)\Big\}, \ \forall\, j \in \mathbb{N}^{[1,m]}, \ \forall\, n \in \mathbb{N}_0. \quad (4.5.4)$$

By restricting the first argument of $\eta(\cdot)$ to $i \in \mathbb{Z}^{[-(c_j-1)/2,(c_j-1)/2]}$, the resulting finite control set $\mathbb{A}(n+1)$ is fully embedded into the input constraint set $\mathbb{U}$ such that $\min \mathbb{A}(n+1) = \min \mathbb{U} = \underline{u}_{\min}$ and $\max \mathbb{A}(n+1) = \max \mathbb{U} = \underline{u}_{\max}$ hold for all $\underline{x}_0 \in \mathcal{X}_N^{\mathrm{s}}$. Note that the cardinality c_j needs to be an odd number with $c_j \in \mathbb{O}_{\geq 3}$ to provide the same number of samples to the left and the right of the control input value $u_j^\circ(n) := \mu_j(\underline{x}_0, n)$. For $q = 2$ and $c_j = 21$, Figure 4.3 visualizes the resulting distribution of sampling points on the interval $[u_{\min,j}, u_{\max,j}]$. Here, $u_j^\circ(n)$ is chosen randomly. Another input domain discretization, which builds on piecewise polynomial functions, can be found in Appendix A.5.

Remark 4.5.1: *Previous control input vector.* The exponential discretization method embeds the previously applied control input vector $\underline{\mu}(\underline{x}_\mu(n-1), n-1)$ into the current finite set of control vectors $\mathbb{A}(n)$ for all $n > 0$. This property facilitates recursive feasibility issues when $N > 1$ (see Ch. 8).

The optimizer does not solve NLP (3.2.8) when searching for the optimal solution of OCP (4.5.1). In contrast to conventional constrained optimization, the combinatorial optimizer does not search for the KKT points by determining and evaluating the derivative information. Here, the optimizer simply evaluates all control candidates following the pseudo-code presented in Algorithm 4.1. This algorithm starts the roll-out procedure for each control candidate $\underline{u}_{\mathrm{s}} \in \mathbb{A}(n)$ in program line 9. If the predicted state trajectory does not satisfy the state constraints at any point on the horizon (program line 12), the algorithm terminates the current roll-out (program line 15) and proceeds with the next control candidate. If the last predicted state vector satisfies the terminal conditions (program line 16), the algorithm appends the current control candidate and the corresponding cost function value to the set of admissible candidates (program line 18). The least-cost tuple in program line 19 represents the global optimum to OCP (4.5.1) with $\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n) \in \mathbb{A}(n)$. A theoretical analysis of the runtime complexity reveals that the big O notation for Algorithm 4.1 is $\mathcal{O}(c \cdot N)$. The space complexity can be approximated by $\mathcal{O}(c \cdot N)$. However, if only two state vectors $\underline{x}(k)$ and $\underline{x}(k+1)$ are buffered instead of the whole state trajectory and only the current best solution is stored in the program line 18, the space complexity is constant. Hence, the big O notation for the memory complexity of Algorithm 4.1 is $\mathcal{O}(1)$.

Algorithm 4.1.: Procedure to solve OCP (4.5.1) with time-varying finite control sets

```
1: procedure SolveOCP($\underline{x}_\mu(n)$,$\mathbb{A}(n)$)
2:     $\underline{K}$, $\underline{P} \leftarrow$ Optional: Solve Riccati/Lyapunov equation online    ▷ See Sec. 3.3
3:     $\mathcal{S} \leftarrow \emptyset$    ▷ Initialize set of admissible control candidates
4:     $\underline{x}_0 \leftarrow \underline{x}_\mu(n)$
5:     for all candidates $\underline{u}_\mathrm{s} \in \mathbb{A}(n)$ do    ▷ Parallelizable
6:         $k \leftarrow 0$
7:         $\underline{x}(k) \leftarrow \underline{x}_0$
8:         $J \leftarrow 0$
9:         while $k < N$ do    ▷ Roll-out of selected control candidate
10:            $\underline{x}(k+1) \leftarrow \underline{f}\big(\underline{x}(k), \underline{u}_\mathrm{s}\big)$    ▷ Internal model, see (3.1.1)
11:            $J \leftarrow J + \ell\big(\underline{x}(k), \underline{u}_\mathrm{s}\big)$    ▷ Stage costs, see (3.1.3)
12:            if $\underline{\mathcal{G}}\big(\underline{x}(k+1)\big) \preceq \underline{0}$ then    ▷ Check if $\underline{x}(k+1) \in \mathbb{X}$, see (3.2.2)
13:                $k \leftarrow k+1$
14:            else
15:                break
16:        if $k = N$ and $\underline{\mathcal{F}}\big(\underline{x}(N)\big) \preceq \underline{0}$ then    ▷ Check if $\underline{x}(N) \in \mathbb{X}_\mathrm{f}$, see (3.2.7)
17:            $J \leftarrow J + F\big(\underline{x}(N)\big)$    ▷ Terminal costs, see (3.1.3)
18:            $\mathcal{S} \leftarrow \mathcal{S} \cup \{(J, \underline{u}_\mathrm{s})\}$    ▷ Append admissible tuple
19:    $\big(V_N^\mathrm{s}(\underline{x}_0, n), \underline{u}_\mathrm{s}^*(\underline{x}_0, n)\big) \leftarrow$ Select least-cost tuple from $\mathcal{S}$    ▷ Determine global optimum
20:    $\mathbb{A}(n+1) \leftarrow$ Adaptive discretization    ▷ Evolution of finite control set, see (4.5.4)
21:    return $\underline{u}_\mathrm{s}^*(\underline{x}_0, n), \underline{\mathbf{u}}_\mathrm{s}^*(\underline{x}_0, n) = \underline{\Theta}_N^\mathrm{s}\big(\underline{u}_\mathrm{s}^*(\underline{x}_0, n)\big), \mathbb{A}(n+1)$
```

Domain Discretization Error

The benchmark configuration from Example 4.3 serves as the basis for evaluating the discretization error. However, the configuration includes the following modifications:

$$N = 1,\ \underline{R} = 0.0006,\ \underline{P} = \underline{A}^\intercal \underline{P}\,\underline{A} - (\underline{A}^\intercal \underline{P}\,\underline{B})(\underline{R} + \underline{B}^\intercal \underline{P}\,\underline{B})^{-1}(\underline{B}^\intercal \underline{P}\,\underline{A}) + \underline{Q}. \qquad (4.5.5)$$

With $N = 1$, input move-blocking does not apply. With the modifications above and $\bar{\mathbb{P}} := \mathbb{U}$ in OCP (4.1.4), SFMPC reproduces the LQR such that $\underline{\mu}(\underline{x}_0) = \underline{K}\,\underline{x}_0$ holds for all $\underline{x}_0 \in X$ (see Prop. 4.2.2). The matrix $\underline{K}$ follows from (3.3.19) with $\rho = 1$. Therefore, deviations between the closed-loop performance of SFMPC, which rests upon the solutions to OCP (4.5.1) with $\mathbb{P}(\underline{x}_0, n) := \mathbb{A}(n)$ and $N = 1$, and the closed-loop performance of the LQR occur solely due to input domain discretization. Note that in the absence of input and state constraints, it follows that $\mathcal{U}_1^\mathrm{s}(\cdot) = U^N$. Thus, for some arbitrary finite set of control vectors $\mathbb{P}(\underline{x}_0, n) \subset U$, every sample $\underline{u}_\mathrm{s} \in \mathbb{P}(\underline{x}_0, n)$ results in a finite cost function value of the continuous function $J_N\big(\underline{x}_0, \underline{\Theta}_N^\mathrm{s}(\underline{u}_\mathrm{s})\big) \in \mathbb{R}_0^+$ for all $\underline{x}_0 \in X$. Therefore, OCP (4.5.1) with $\mathbb{P}(\underline{x}_0, n) := \mathbb{D}$ or $\mathbb{P}(\underline{x}_0, n) := \mathbb{A}(n)$ is also feasible for all $\underline{x}_0 \in \mathcal{X}_N^\mathrm{s}$ with $n \in \mathbb{N}_0$. However, the proposed adaptive discretization requires input bounds $\underline{u}_\mathrm{min}$ and $\underline{u}_\mathrm{max}$ in order to place samples in some neighborhood of the origin. Although there are no input constraints, SFMPC discretizes the input on the interval $[-15, 15]$. The weight $\underline{R}$ is increased from $\underline{R} = 0$ to $\underline{R} = 0.0006$ to ensure that the LQR implicitly adheres to the input box-constraints with $|u(k)| \leq 15$. In addition, the Riccati equation (3.3.18) assumes that the weighting matrices are positive definite. To

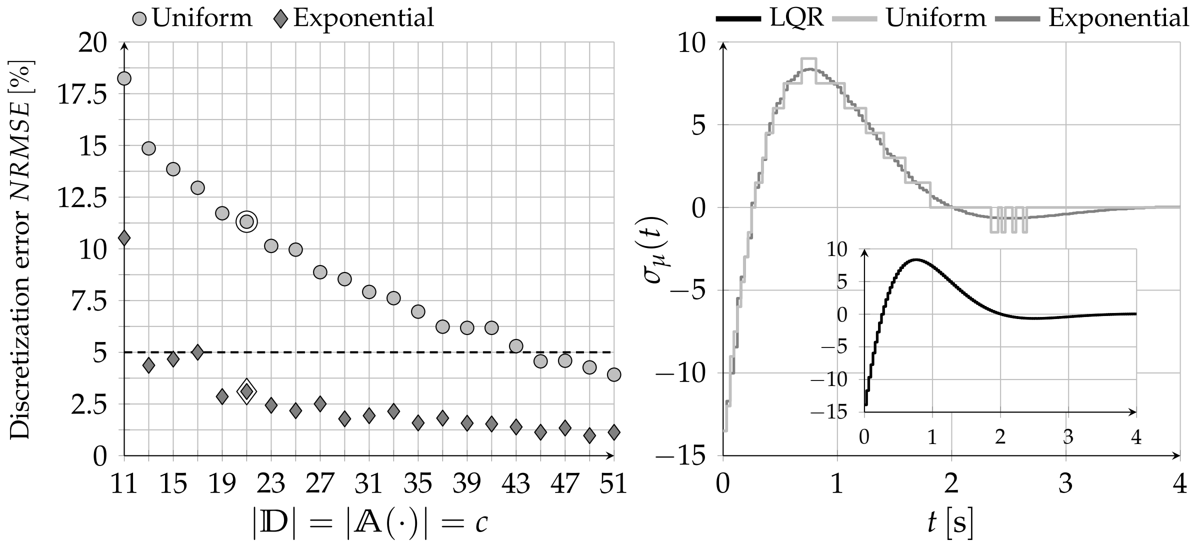


Figure 4.4.: Comparison of input discretization approaches using the mass-spring-damper system from Example 4.3. Left: Discretization error in the control input trajectory over cardinality c. Right: Closed-loop time performance with SFMPC and the uniform and exponential discretization with $c = |\mathbb{D}| = |\mathbb{A}(\cdot)| = 21$ and $q = 3$. The subplot shows the reference closed-loop control performance of the LQR.

compare the one-dimensional control input trajectory of SFMPC with that of the LQR, the normalized root mean square error (NRMSE) is evaluated over 128 closed-loop steps ($t \in [0\,\mathrm{s}, 4\,\mathrm{s}]$). Refer to Appendix A.4 for the NRMSE definition. At time $n = 0$, the mass-spring-damper system is initialized with $\underline{x}_{\mu,0} = (0.8, 0)^\mathsf{T}$. The left plot of Figure 4.4 shows that the discretization error decreases approximately quadratically with a uniform discretization for an increasing number of control candidates. Only with a cardinality of $c = |\mathbb{D}| = 45$, the discretization error drops below $NRMSE = 5\,\%$. In contrast, the exponential discretization requires only a cardinality of $c = |\mathbb{A}(\cdot)| = 13$ to provide a discretization error below $NRMSE = 5\,\%$. The right plot of Figure 4.4 shows the closed-loop control input trajectories for SFMPC with $\mathbb{P}(\cdot) := \mathbb{D}$ and $\mathbb{P}(\cdot) := \mathbb{A}(\cdot)$. Here, the cardinality is set to $c = |\mathbb{D}| = |\mathbb{A}(\cdot)| = 21$. The control trajectory that results from uniform discretization exhibits large quantization steps. In contrast, the exponential discretization provides a smooth control input trajectory, which is fairly similar to that of the LQR (see subplot). Hence, adaptive discretization aims to smooth the control input trajectory although building on a sparse sampling. The time-varying and local clustering of samples is designed to increase the control performance and to decrease the computational effort at the same time. For fast closed-loop control switching, all finite control sets always include the boundary values $\min \mathbb{U}$ and $\max \mathbb{U}$. Note that the exact configuration of the discretization approach depends on the system characteristics and the interval lengths $[u_{\min,j}, u_{\max,j}]$ for all $j \in \mathbb{N}^{[1,m]}$. The more sensitive the system is to input variations and the larger the interval $[u_{\min,j}, u_{\max,j}]$, the more samples are required to establish a high closed-loop control performance. Recall that at time $n = 0$, the definition $\mathbb{A}(0) := \mathbb{D}$ applies with $c = |\mathbb{A}(0)| = |\mathbb{D}|$. Though the exponential discretization reduces the discretization error significantly, a finite error remains. For a cardinality of $c = 101$, uniform discretization results in an error of $NRMSE = 2.13\,\%$, while the exponential discretization provides an error of only $NRMSE = 0.55\,\%$. The next subsection investigates whether this finite error impacts the previously derived stability results in Section 4.2.

Stability Analysis with Finite Control Sets

Although the finite control set $\mathbb{D}$ contains the origin, as required by Assumption 3.1.3, there is no guarantee that it contains the control vector $\underline{u}_\mathrm{s} = \underline{\kappa}(\underline{x})$ if $\underline{x} \in \mathbb{X}_\mathrm{f}$, thus satisfying Assumption 3.3.2. For the finite set $\mathbb{A}(n)$, it cannot even be ensured that it contains the origin. The following formulation addresses the problem of control set finiteness by including further qualified samples with $\underline{u}_\mathrm{s}^0 \in \mathbb{U}$ and $\underline{\Gamma}(\cdot) : U \mapsto \mathbb{U}$:

$$\tilde{\mathbb{A}}\big(\underline{x}_\mu(n), n\big) := \mathbb{A}(n) \cup \big\{\underline{\Gamma}\big(\underline{\kappa}\big(\underline{x}_\mu(n)\big)\big)\big\},\ \forall\, \underline{x}_\mu(n) \in \mathcal{X}_N,\ \forall\, n \in \mathbb{N}_0. \tag{4.5.6}$$

The saturation function $\underline{\Gamma}(\cdot)$ enforces the values of the additional control candidate $\underline{\kappa}(\underline{x})$ to lie between $\max \mathbb{U}$ and $\min \mathbb{U}$ if $\underline{x} \notin \mathbb{X}_\mathrm{f}$. With this formulation, the extended finite set of control vectors $\tilde{\mathbb{A}}\big(\underline{x}_\mu(n), n\big)$ does not contain the origin for all times, however, it contains the origin when the closed-loop system reaches the origin with $\underline{x}_\mu(n) = \underline{0}_p$. Note that none of the constraint sets, such as $\mathcal{U}_N^\mathrm{s}(\underline{x}_0)$ or $\mathcal{X}_N^\mathrm{s}$, consider finite control sets in their formulations. To establish theoretical closed-loop properties with finite control sets, the following mild assumption is required.

Assumption 4.5.1: *Initial control sample.* At closed-loop time $n = 0$, there is an arbitrary but admissible sample $\underline{u}_\mathrm{s}^0 \in \mathbb{A}(0) \subset \mathbb{U}$ such that $\mathbf{u}_\mathrm{s} = \underline{\Theta}_N^\mathrm{s}(\underline{u}_\mathrm{s}^0) \in \mathcal{U}_N^\mathrm{s}\big(\underline{x}_\mu(0)\big)$ holds.

This technical assumption is similar to the "oracle" in [BL17], generating the initial admissible solution. In general, the elements of the set $\mathbb{A}(0)$ can be chosen randomly as long as $\mathbb{A}(0) \subset \mathbb{U}$ and $\underline{u}_\mathrm{s}^0 \in \mathbb{A}(0)$. If $\mathbb{A}(0) := \mathbb{D}$ and the initial sample $\underline{u}_\mathrm{s}^0$ is not available, then the discretization step widths Δu_j in (4.4.1) can be decreased until $\underline{u}_\mathrm{s}^0 \in \mathbb{D}$. With $N = 1$, the following stability properties can already be derived.

Corollary 4.5.1: *SFMPC with finite control sets emulates conventional MPC.* Suppose Assumptions 3.1.1-3.1.3, 3.3.1-3.3.2, and 4.5.1 hold. Define $\mathbb{P}(\underline{x}_0, n) := \tilde{\mathbb{A}}(\underline{x}_0, n)$ and the control law $\underline{\mu}(\underline{x}_0, n) := \underline{u}_\mathrm{s}^*(\underline{x}_0, n)$ for all $\underline{x}_0 \in \mathcal{X}_1^\mathrm{s}$, all $n \in \mathbb{N}_0$. Then, the optimal cost function $V_1^\mathrm{s}(\cdot)$ in (4.5.1) represents a time-varying Lyapunov function in the set $\mathcal{X}_1^\mathrm{s}$ for the closed-loop system (4.5.2) with $\alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}_\infty$ for all $\underline{x}_0 \in \mathcal{X}_1^\mathrm{s}$, all $n \in \mathbb{N}_0$:

$$\alpha_1(\|\underline{x}_0\|) \leq V_1^\mathrm{s}(\underline{x}_0, n) \leq \alpha_2(\|\underline{x}_0\|),\ V_1^\mathrm{s}\Big(\underline{f}\big(\underline{x}_0, \underline{\mu}(\underline{x}_0, n)\big), n+1\Big) \leq V_1^\mathrm{s}(\underline{x}_0, n) - \alpha_1(\|\underline{x}_0\|). \tag{4.5.7}$$

Therefore, the origin is asymptotically stable in the positive invariant set $\mathcal{X}_1^\mathrm{s}$ for the closed-loop system (4.5.2).

The detailed proof is given in Appendix B.1. However, since the finite set $\tilde{\mathbb{A}}(\underline{x}_0, n)$ contains the local control law sample $\underline{\kappa}(\underline{x}_0)$ for all $\underline{x}_0 \in \mathbb{X}_\mathrm{f}$ and all $n \in \mathbb{N}_0$, $V_1^\mathrm{s}(\underline{x}_0, n) \leq F(\underline{x}_0) \leq \alpha_\mathrm{f}(\|\underline{x}_0\|)$ also holds for all $\underline{x}_0 \in \mathbb{X}_\mathrm{f}$ and all $n \in \mathbb{N}_0$. This inclusion of the local control law sample into $\tilde{\mathbb{A}}(\underline{x}_0, n)$ ensures $V_1^\mathrm{s}(\cdot)$ is continuous at the origin (required for upper and lower bounds of Lyapunov function) and that the terminal cost function $F(\cdot)$ is still a CLF in $\mathbb{X}_\mathrm{f}$ (see Asm. 3.3.2).

Corollary 4.5.2: *SFMPC with finite control sets reproduces LQR.* Proposition 4.2.2 also holds if SFMPC is based on OCP (4.5.1) with $\mathbb{P}(\underline{x}_0, n) := \tilde{\mathbb{A}}(\underline{x}_0, n)$ for all $\underline{x}_0 \in X$ and all $n \in \mathbb{N}_0$.

Proof. The finite control set $\tilde{\mathbb{A}}(\underline{x}_0, n)$ contains the global optimum $\underline{u}_s^*(\underline{x}_0, n) = \underline{K}\,\underline{x}_0$ for all $\underline{x}_0 \in X$ and $n \in \mathbb{N}_0$ such that $F(\cdot)$ still serves as the global Lyapunov function. □

Corollary 4.5.3: *SFMPC with finite control sets for open-loop stable systems.* Proposition 4.2.3 also holds if SFMPC is based on OCP (4.5.1) with $\mathbb{P}(\underline{x}_0, n) := \tilde{\mathbb{A}}(\underline{x}_0, n)$ for all $\underline{x}_0 \in X$ and all $n \in \mathbb{N}_0$.

Proof. The finite control set $\tilde{\mathbb{A}}(\underline{x}_0)$ contains the zero vector $\underline{\kappa}(\underline{x}_0) := \underline{0}_m$ for all $\underline{x}_0 \in X$ and $n \in \mathbb{N}_0$ such that $F(\cdot)$ still serves as the global Lyapunov function. □

4.6. Discussion

The MPC formulation introduced in this chapter with a single degree of freedom in control reduces the controller complexity significantly and motivates the development of an adaptive and sparse discretization of the input domain. As a result of the input domain discretization, the OCPs are subject to finite control sets. The resulting combinatorial complexity is sufficiently small such that the exhaustive search algorithm qualifies for solving the corresponding OCPs. Algorithm 4.1 illustrates the simple implementation, which dispenses with external optimization libraries and solvers. For three special system configurations, basic SFMPC already stabilizes the origin with the closed-loop system. Even though two of the three configurations are rather theoretical, they form the basis for the systematic analysis of basic SFMPC. The investigations of the Special Case I in Proposition 4.2.1 and Corollary 4.5.1 identify which control vectors the finite control sets have to contain to ensure recursive feasibility and asymptotic stability. Under the mild Assumption 4.5.1, combined with the strategy of embedding the local controller law $\underline{\kappa}(\cdot)$ into the definition of the time-varying finite control set, SFMPC emulates conventional MPC in the absence of input move-blocking and in the presence of input discretization errors. The introduction and examination of the second Special Case II in Proposition 4.2.2 and Corollary 4.5.2 enable the isolated analysis of the input domain discretization error. With a horizon length of $N = 1$, SFMPC reproduces the conventional LQR closed-loop control performance. Any deviation compared to the LQR thus arises only because of the input domain discretization. However, with the proposed straightforward exponential discretization, the resulting error is negligible for the chosen benchmark system. The analysis of the third Special Case III in Proposition 4.2.3 and Corollary 4.5.3 addresses practical applications and prepares a general design guide for choosing the horizon length N for SFMPC. While a moderate horizon length with $N > 1$ increases the closed-loop control performance, it drops with a long horizon. Because of the single degree of freedom in control, the predicted system evolution differs increasingly from the actual evolution of the closed-loop system as the horizon N increases. The investigation of an open-loop stable system enables the limit value analysis with $N \to \infty$ since only in this case SFMPC does not destabilize the origin with a long horizon.

5 Comparative Numerical Analysis

This chapter investigates the applicability of basic SFMPC to constrained nonlinear systems. The objective is to highlight SFMPC as an advanced but at the same time intuitive low-level control concept for nonlinear systems with small input dimensions, small to mid-sized state dimensions, and simple box-constraints. In particular, the objective is to narrow the scope of basic SFMPC and to point out its practical advantages such as the handling of non-smooth functions. Parts of this chapter have been published in [Mak+17; Mak+18d; Mak+18e].

5.1. Controller Configuration for Constrained Systems

The analysis in the previous Chapter 4 focuses on three special system configurations for which theoretical closed-loop stability guarantees can be derived. This chapter explores whether basic SFMPC applies to constrained nonlinear systems, even when no theoretical stability guarantees exist. Established configuration guidelines for MPC with or without terminal conditions are designed for conventional MPC with full degree of freedom in control and are thus not directly applicable to input move-blocked MPC. For this reason, the numerical and experimental investigations in this and the next chapter, respectively, simply assume that the infinite horizon cost function $F(\underline{x}) = \underline{x}^\mathsf{T}\,\underline{P}\,\underline{x}$ of the linearized system is suitable to serve as a general direction guide for the nonlinear system, even outside the terminal set $\mathbb{X}_\mathrm{f} := \mathrm{lev}_\pi F$. The following simulations combine a horizon length of $N > 1$ with quadratic form cost functions from (3.1.4), where $\underline{P}$ is the solution to the Riccati equation (3.3.18). The terminal region is given by $\mathbb{X}_\mathrm{f} = \mathbb{X}$. Thus, the terminal set does not satisfy Assumption 3.3.2.

Remark 5.1.1: *LQR with control saturation.* Consider quadratic cost functions according to (3.1.4) with $F\big(\underline{x}(N)\big) := \underline{x}^\mathsf{T}(N)\underline{P}\,\underline{x}(N)$ and assume that $\underline{P}$ solves the Riccati equation (3.3.18). If the solution $\underline{P}$ exists, then the origin is stabilizable in X with the linearized system (see Def. 3.3.9). The shorter the time horizon N, the less the evolution of the constrained system contributes to the sum of stage costs in OCP (3.1.6), OCP (4.1.4), and (4.5.1). Therefore, with $N = 1$, the resulting closed-loop control performances with MPC and SFMPC are fairly similar to the evolution of $\underline{x}_\mu(n+1) = \underline{f}\big(\underline{x}_\mu(n), \underline{\Gamma}\big(\underline{K}\,\underline{x}_\mu(n)\big)\big)$, where the gain matrix $\underline{K}$ is defined according to (3.3.19). However, combining a short horizon with $N \geq 1$ and the terminal cost function $F\big(\underline{x}(N)\big) := \underline{x}^\intercal(N)\,\underline{P}\,\underline{x}(N)$ enables compliance with state constraints above a certain horizon length. This special controller configuration, however, assumes that the evolution of the linearized system is similar to the evolution of the nonlinear system.

Recall that in conventional MPC with a control invariant terminal set $\mathrm{lev}_\pi F$, the feasible set $\mathcal{X}_N$ is positive invariant on the one hand but it also represents the region of attraction of the origin for the closed-loop system (3.3.2) or (4.5.2). With the setting $\mathbb{X}_\mathrm{f} = \mathbb{X}$, the feasible sets $\mathcal{X}_N$ and $\mathcal{X}_N^\mathrm{s}$ do not necessarily represent the regions of attraction anymore. Therefore, the numerical analysis in this chapter rests upon the assumption that MPC and SFMPC can steer the system in some positive invariant terminal set $\mathcal{T} := \mathcal{X}_N \cap \mathcal{X}_N^\mathrm{s} \cap \mathcal{B}_\delta$ with some small $\delta > 0$. As soon as the closed loop system enters this terminal set, it never leaves the terminal set $\mathcal{T}$ again under the implicit control law (see [GP17, Def. 2.15]). The experimental stability analysis in Appendix B.3 builds on this formulation and shows an alternative and automated way for investigating (practical) closed-loop stability for the following benchmark system, when no theoretical stability guarantees can be derived from the chosen controller configuration.

5.2. Example: Van der Pol Oscillator

To follow up on the example in Section 4.3, this section also investigates a common second-order dynamical system, albeit with nonlinear damping. The following nonlinear ordinary differential equation describes the well-known Van der Pol oscillator [Pol26] with the input $\sigma : \mathbb{R}_0^+ \mapsto \mathbb{R}$ and the output $y : \mathbb{R}_0^+ \mapsto \mathbb{R}$:

$$\ddot{y}(t) - \big(1 - y^2(t)\big)\dot{y}(t) + y(t) = \sigma(t). \tag{5.2.1}$$

This input-affine nonlinear system can be transformed into the following state space representation with $\underline{\chi}(t) = \big(\chi_1(t) = y(t), \chi_2(t) = \dot{y}(t)\big)^\intercal$:

$$\dot{\underline{\chi}}(t) = \begin{pmatrix} \dot{y}(t) \\ \big(1 - y^2(t)\big)\dot{y}(t) - y(t) + \sigma(t) \end{pmatrix}. \tag{5.2.2}$$

The input of the system $\sigma(t)$ is assumed to be piecewise constant on the fixed time grids $t_{k+1} = t_k + T_\mathrm{s}, k \in \mathbb{N}^{[0,N-1]}, t_0 = 0\,\mathrm{s}$ (open-loop) and $t_{n+1} = t_n + T_\mathrm{s}, n \in \mathbb{N}_0, t_0 = 0\,\mathrm{s}$ (closed-loop) with $T_\mathrm{s} = \Delta t_\mathrm{s}$. According to the sampled data formulation in Section 3.4, the internal discrete-time model includes the continuous-time representation (5.2.2) and the Runge-Kutta method of the fourth order with $\Delta t_\mathrm{s} = 2^{-5}\,\mathrm{s}$. The following list summarizes all related parameters and sets:

$$\underline{A} \approx \begin{pmatrix} 0.9995 & 0.0317 \\ -0.0317 & 1.0312 \end{pmatrix}, \underline{B} \approx \begin{pmatrix} 0.0005 \\ 0.0317 \end{pmatrix}, \underline{Q} = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.25 \end{pmatrix}, \underline{P} \approx \begin{pmatrix} 34.012 & 7.427 \\ 7.427 & 12.471 \end{pmatrix},$$
$$\underline{R} = 0.1, \mathbb{U} = \{u \in \mathbb{R} \mid |u| \le 0.9\}, \mathbb{X} = \{\underline{x} \in X \mid (-1, -0.5)^\intercal \preceq \underline{x} \preceq (1, 0.5)^\intercal\}.$$

The origin is open-loop unstable for the linearized system $\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{B}\,\underline{u}(k)$ since the complex conjugate eigenvalues of the system matrix $\underline{A}$ lie outside the unit

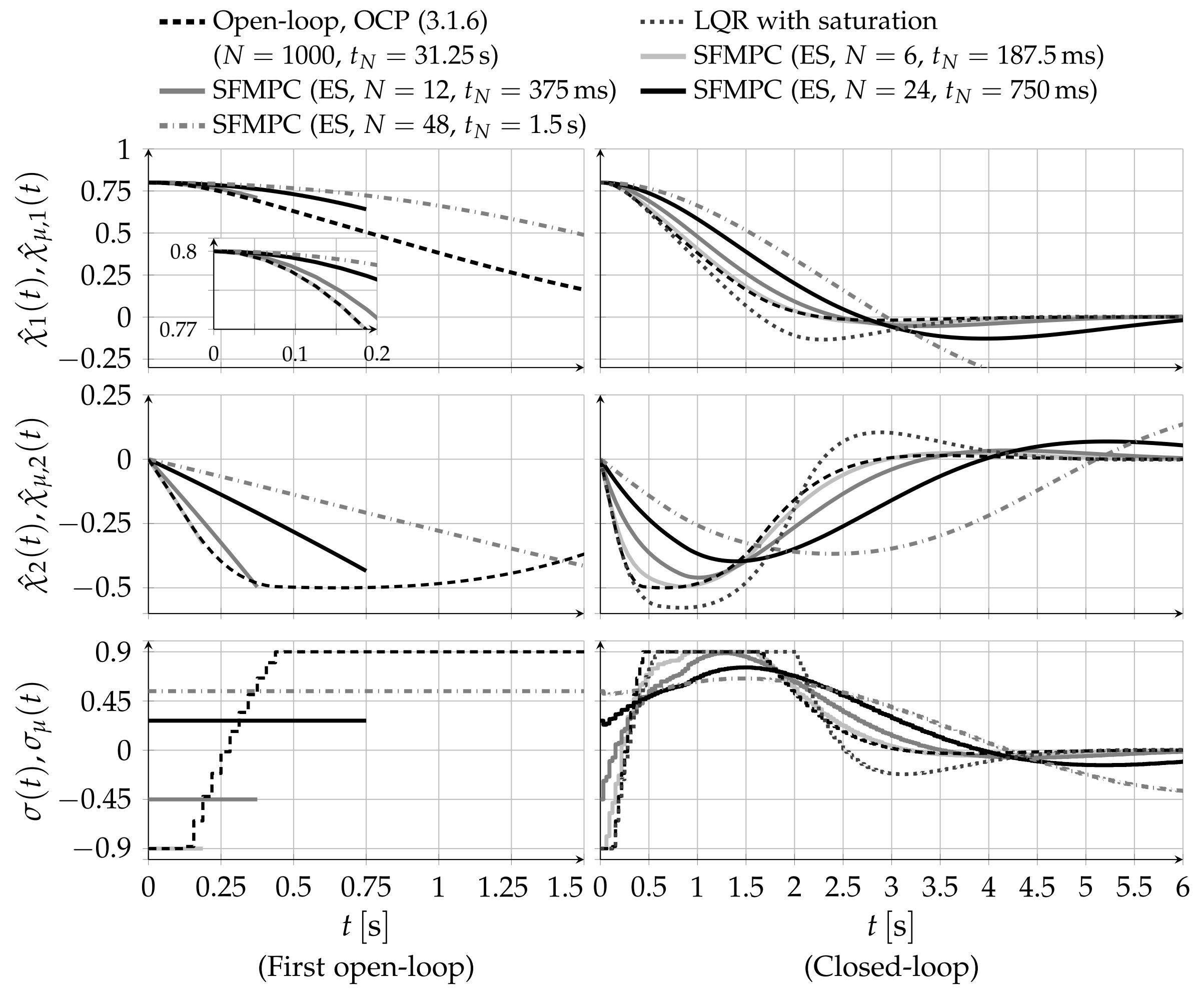


Figure 5.1.: Open- and closed-loop predictive control of the Van der Pol oscillator with different horizon lengths. Left: Open-loop control performance at time instant $n = 0$. Right: Closed-loop control performance. For reference, the dashed graphs show the solution to OCP (3.1.6) with $N = 1000$.

circle. All steady states of the nonlinear system satisfy $\underline{x}_\mathrm{f} = (u_\mathrm{f}, 0)^\intercal$ with $u_\mathrm{f} \in \mathbb{U}$. With MPC, full discretization is used to convert OCP (3.1.6) into NLP (3.2.9), which is then solved numerically by the IPOPT algorithm [WB06]. SFMPC implements the exponential discretization according to (4.5.3) with a cardinality of $c = |\mathbb{A}(\cdot)| = 21$. Algorithm 4.1 is used to solve OCP (4.5.1) with $\mathbb{P}(\underline{x}_0, n) := \mathbb{A}(n)$. Figure 5.1 shows the open-loop and closed-loop control performances of SFMPC for increasing horizon lengths. The black dashed graphs visualize the optimal point-to-point reference solution to which classical MPC would converge with increasing horizon length. The reference solution clearly illustrates the advantages of the full-degree-of-freedom OCP (3.1.6). The bottom plots show that the optimal reference solution first drives the system with the minimum input of $\sigma(t) = u(k) = -0.9$ and then switches to the maximum input of $\sigma(t) = u(k) = 0.9$ to not violate the velocity constraint $|\hat{\chi}_2(t)| = |x_2(k)| \leq 0.5$. Hence, the lower the weighting parameter $\underline{R}$, the more similar becomes the reference solution to a bang-bang control. Because of the full degree of freedom in control, the optimizer exploits the limits of the system efficiently during prediction. In contrast, the second plot on the left shows that a long horizon in SFMPC complicates the handling of state constraints. The longer the horizon, the closer the solution is to $\sigma(t) = u(k) = 0.8$ in the left bottom plot. This value is required to hold the system at its initial state of $\underline{x}_{\mu,0} = (0.8, 0)^\mathsf{T}$. Since SFMPC only predicts a constant control sequence, it chooses a low system excitation to keep the entire open-loop state trajectory inside the feasible state set $\mathcal{X}_N^{\mathrm{s}}$. As long as the predicted system evolution is far away from crossing the velocity limit of $|x_2(k)| \leq 0.5$, the predicted control trajectory is similar to its corresponding section of the reference optimal point-to-point trajectory. In simplified terms, for short horizons, the initial prediction is carried out as if the velocity limit did not exist. However, as the closed-loop system approaches the velocity limit, a minimal horizon length of $N = 3$ is required to prevent the prediction from violating the velocity constraint. Notice that the graphs with $N = 6$ are partially covered by the dashed reference graphs. The right side of Figure 5.1 shows oscillatory closed-loop control performances for SFMPC with $N = 24$ and $N = 48$. Here, $\underline{\sigma}_\mu(t) := \underline{\mu}(\underline{x}_\mu(n), n),\ \forall\, t \in \left[nT_\mathrm{s}, (n+1)T_\mathrm{s}\right),\ n \in \mathbb{N}_0$ applies. This detrimental relationship between long prediction horizons and decreasing closed-loop control performances coincides with the observations for state unconstrained linear systems in Figure 4.2. However, with $N = 6$ and even $N = 12$, the closed-loop control performances are fairly similar to the reference solution though implementing SFMPC includes only a few lines of code (see Alg. 4.1). To follow up the discussion in Remark 5.1.1, finally, a simple LQR with a subsequent input limitation is applied to the nonlinear Van der Pol oscillator (dotted graphs). For this benchmark system, the saturation of control inputs does not destabilize the origin. However, the dotted graph in the second plot on the right side of Figure 5.1 clearly violates the velocity constraint. Recall that the LQR can only adhere to input and state constraints implicitly by increasing the weights in $\underline{Q}$ and $\underline{R}$ and thus by damping closed-loop dynamics.

Finally, the following Table 5.1 provides a quantitative comparison of execution times and closed-loop control performances for SFMPC and classical MPC. The statistical evaluation of execution times is based on 100 cold-started solutions to OCP (3.1.6) or OCP (4.5.1) at time instance $n = 0$. For the numerical evaluation in this dissertation, a common personal computer is used, which does not support real-time computing (see App. A.3 for specification). Depending on how the scheduler of the operating system distributes the internal tasks, outliers can occur when evaluating the computational effort. Therefore, the statistical investigations consider the 0.5-quantile, also known as the median $t_\mathrm{m} \in \mathbb{R}_0^+$, and the 0.95-quantile $t_{95} \in \mathbb{R}_0^+$. The median does not depend on the specific distribution and is therefore known to be robust to outliers. The 0.95 quantile excludes only the fourth largest outliers out of 100 samples, thus covering some of the variation in execution times. In addition, the measured run times also include the computation times required to linearize the system at the origin and to solve the Riccati equation (3.3.18). All optimization variables are always initialized to zero. Since the state constraints listed above only apply from a certain horizon length ($N > 6$) at closed-loop time $n = 0$, the state constraint set is now redefined as $\mathbb{X} := X$ to provide a fair comparison of computational loads between different horizon lengths.

Table 5.1.: Evaluation of closed-loop control performances and execution times for (SF)MPC applied to the Van der Pol Oscillator. The values of the first part of the table are generated based on a pure MATLAB implementation that interfaces with the IPOPT algorithm. The second part evaluates the runtime performance of generated and compiled C code. Abbreviations: Recursive Elimination (RE), Full Discretization (FD), Exhaustive Search (ES).

	MATLAB	$N=3$		$N=6$		$N=12$		$N=24$		$N=48$	
	MPC	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]
IPOPT	RE	**−8.37**	**12.61**	−10.4	24.57	−8.54	48.56	−1.33	186.58	0.00	754.59
IPOPT	FD	−8.32	26.04	−10.39	46.16	−8.53	80.89	−1.33	168.18	**0.00**	**377.16**
	SFMPC	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]	$\bar{J}_{\text{cl}}$ [%]	t_{m} [ms]
	IPOPT RE	−3.68	6.75	−1.95	7.57	−5.15	9.08	−24.05	14.14	−175.91	22.34
	ES $c=21$	−3.82	3.52	**−2.02**	**4.93**	−5.01	7.43	−23.8	13.01	−175.21	23.98
	ES $c=101$	−3.68	10.55	−1.94	16.05	−5.14	28.14	−24.05	50.06	−175.91	94.11
	C, MPC	t_{m} [μs]	t_{95} [μs]	t_{m} [μs]	t_{95} [μs]	t_{m} [μs]	t_{95} [μs]	t_{m} [μs]	t_{95} [μs]	t_{m} [ms]	t_{95} [ms]
SQP	RE	**34.8**	**35.95**	50.5	58.05	125.75	143.6	446.5	511.45	**2.39**	**2.51**
SQP		[μs]	[μs]	[μs]	[μs]	[ms]	[ms]	[ms]	[ms]	[ms]	[ms]
SQP	FD	110.85	151.75	300.25	348.6	1.24	1.4	7.24	7.69	26.27	27.28
	SFMPC	t_{m} [μs]	t_{95} [μs]	t_{m} [μs]	t_{95} [μs]	t_{m} [μs]	t_{95} [μs]	t_{m} [μs]	t_{95} [μs]	t_{m} [μs]	t_{95} [μs]
	SQP RE	31.55	34.4	36.1	37.5	39.6	43.4	74.5	77.4	87.9	110.9
	ES $c=21$	8.3	8.4	**9.6**	**10.05**	10.7	11.7	34.3	42.35	35.7	36.1
	ES $c=101$	14.6	14.75	22.7	30.05	41.1	41.2	74.4	76.6	142.7	143.2

The generic and modular software framework developed for this dissertation is implemented in MATLAB and includes the sparse IPOPT optimization library [WB06] via a pre-compiled interface and the SQP method described in [Rös19, Appx. E.2.2.]. This SQP algorithm represents an extended version of the backtracking line search SQP algorithm from [NW06, Alg. 18.3] and supports C code generation of the entire software framework. Task execution times are thus evaluated in MATLAB (interpreted language) and in C (compiled language). The open-source IPOPT library serves as an established and numerically robust reference optimization framework. The SQP implementation is adopted from [Rös19] and re-implemented from scratch in MATLAB. Implementing both SFMPC with exhaustive search and SFMPC/MPC with an SQP method from scratch facilitates a fair comparison of computational and implementation efforts. Clearly, both the combinatorial and the smooth optimization can be improved in terms of computational efficiency by investigating, for example, tailored combinatorial search strategies or advanced Hessian matrix approximations, respectively. The IPOPT and SQP algorithm both estimate first-order derivative information based on sparse finite differences. In this dissertation, both implementations approximate the Hessian of the Lagrangian by an iterative method. The IPOPT framework [WB06] implements the limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) method that accounts for large-scale optimization problems [Noc80]. The SQP implementation, in contrast, uses the damped BFGS method from [NW06, Proc. 18.2]. For the following example, the SQP algorithm integrates the sparse QP solver OSQP [Ste+20]. In principle, sparsity or structure detection could be outsourced to an offline pre-processing step, as the sparsity pattern, or matrix structure in general, is constant with a receding horizon. However, the time required to exploit the NLP structure dynamically is negligible compared to the core optimization time, at least for the benchmarks in this dissertation. For further information regarding the implementation, refer to Appendix A.3. The closed-loop control performances are evaluated based on the normalized closed-loop control performance criterion $\bar{J}_{\text{cl}}$ from (4.3.3) with $l = 1000$, where the reference is the initial solution to OCP (3.1.6) with $N = 1000$, $\underline{\mathbf{u}}_{\text{ref}} := \underline{\mathbf{u}}^*(\underline{x}_{\mu,0})$ and $\underline{x}_{\mu,0} = (0.8, 0)^\intercal$. Here, $\bar{J}_{\text{cl}} \in (-\infty\,\%, 0\,\%]$ applies, where $\bar{J}_{\text{cl}} < 0\,\%$ indicates a degradation of the closed-loop control performance compared to the optimal point-to-point reference.

The first section of Table 5.1 focuses on the IPOPT interface. As claimed in the beginning of the section, the first two rows reveal that the closed-loop state and input trajectories converge towards the open-loop reference solution as the horizon length increases. For $N = 48$, the closed-loop control performance is similar to the reference performance with $\bar{J}_{\text{cl}} \approx 0\,\%$. Apart from minor numerical differences, MPC with recursive elimination and MPC with full discretization yield the same closed-loop control performances. However, for a low number of optimization parameters ($N = 3$, $N = 6$, and $N = 12$), the additional continuity equality constraints in case of full discretization increase the NLP dimensions such that recursive elimination outperforms full discretization in terms of computation time. However, as the horizon increases, the IPOPT algorithm benefits from efficient NLP structure exploitation. For $N = 48$, the high dimensional but sparse NLP in case of full discretization outperforms the smaller but more dense NLP resulting from recursive elimination. In the following three rows, the degrees of freedom in control are limited to a single degree of freedom. Here, the execution times are significantly lower compared to classical MPC with full degree of freedom in control. Notice that the closed-loop control performances cannot be compared directly between SFMPC and MPC for a particular horizon length. Due to the extreme input move-blocking, the open-loop performances between SFMPC and MPC differ strongly. Hence, the same weighting parameters result in different closed-loop control trajectories. In this example, the closed-loop control performances with $N = 3$, $N = 6$, and $N = 12$ happen to be higher for SFMPC compared to MPC. However, MPC can also provide a similar closed-loop control performance by adjusting the elements of the weighting matrices $\underline{Q}$ and $\underline{R}$ while maintaining the previous open-loop reference performance in the denominator of (4.3.3). More important is the effect, already observed in Figure 5.1, that for SFMPC, the closed-loop control performance decreases as the horizon increases. Further, the fourth and fifth row reveal that the exponential input domain discretization results in closed-loop control trajectories that are similar to the one resulting from smooth optimization. The closed-loop control performances in the third, fourth, and fifth row are almost identical. Increasing the cardinality from $c = 21$ to $c = 101$ does not increase the closed-loop control performances significantly, whereas the computation times exceed the effort resulting from smooth optimization. Hence, adaptive discretization of the input domain provides an application scope for

SFMPC. With $N = 6$, the horizon length has some distance to the minimum horizon length of $N = 3$ to ensure experimental recursive feasibility (see Fig. 5.1). Furthermore, with a small cardinality of $c = 21$, SFMPC reaches a closed-loop control performance of already $\bar{J}_{\mathrm{cl}} = -2.02\,\%$ and is approximately 35 % faster than SFMPC with smooth optimization. In addition, it should be noted that SFMPC with finite control sets is based on the simple implementation in Algorithm 4.1. Since the low-level programming language C is usually used in hardware-oriented software development, where memory and computing power play an important role, the second part of table 5.1 analyzes the computation times resulting from the execution of generated and compiled C code. Though the SQP algorithm integrates the sparse QP solver OSQP [Ste+20], recursive elimination outperforms the implementation of full discretization in terms of runtime performance. The reason for this observation is the implementation of the dense and damped Hessian approximation from [NW06, Proc. 18.2], which exhibits poor convergence properties if the full step BFGS update results in a Hessian of the Lagrangian that is not positive definite. However, the location parameters resulting from recursive elimination in the first row of the second part are fairly similar to the MPC benchmark results presented in, for example, [Rös+18a]. The analysis of the runtime performance of SFMPC in the C environment shows comparatively the same results as when executing interpreted code. A small cardinality of $c = 21$ provides a computational advantage for SFMPC with exhaustive search compared to SFMPC with smooth optimization. With $N = 6$, SFMPC with exhaustive search offers a median of only $t_{\mathrm{m}} = 10.05\,\mu\mathrm{s}$ and thus clearly motivates the application of SFMPC with finite control sets to fast systems. The evaluation of the 0.95-quantile t_{95} reveals that the execution time measurements in the upper range are not distorted by many outliers since the values of t_{m} and t_{95} do not differ strongly for both smooth and combinatorial optimization. In this particular example, SFMPC with finite control sets and exhaustive search outperforms SFMPC with smooth optimization in terms of computational as well as the implementation effort.

5.3. Softening State Constraints

The NLPs introduced in (3.2.8) and (3.2.9) are subject to hard constraints. In practice, however, it is crucial that the MPC controller generates (stabilizing) control inputs when these NLPs are infeasible. Infeasibility may occur due to model mismatch, measurement errors, external disturbances or simply due to the limited time for solving the NLPs numerically. If there exists an upper bound for all perturbations, MPC can be designed in such a way that the predicted trajectories even adhere to the hard constraints in the presence of the worst case perturbation. However, with robust MPC the controller design and the runtime complexity increase, whereby there is nevertheless no guarantee in practice that all perturbations remain below the specified bound for all times (e.g., [Raw+20, Ch. 3]). Another approach, which is commonly used in practice, is a soft constraint formulation with slack variables. For MPC with soft constraints, refer, for example, to [ZC92; ZM95; SR99], [Mac02, Sec. 3.4], and [DB94; KM00b]. In the case of soft constraints, the cost function includes further weighted (smooth) terms for penalizing state or output constraint violations. However, the NLP dimensions increase since the individual slack variables are subject to optimization. Since SFMPC integrates a combinatorial optimization, the following description only deals with implementing a non-smooth penalty function. The authors of [All+16], for example, point out that the ability to violate state constraints temporarily is essential to ensure inherent robustness with (suboptimal) MPC. The same authors suggest to soften hard state constraints by implementing an exact penalty function. A general exact penalty function transforms the constrained NLP (3.2.1) into an unconstrained problem without altering the local optima (e.g., [NW06, Sec. 15.1]). Therefore, the local optima of the original and the unconstrained problem coincide exactly. Note that the polyhedral input constraints in NLP (3.2.8) and NLP (3.2.9) are linear and thus not challenging for the optimization algorithm. In contrast, the state constraints integrate the nonlinear state dynamics. The following function evaluates the polyhedral state constraints with $\underline{x} \in X$:

$$\underline{h}_{\mathrm{ps}}(\underline{x}) = \big(h_{\mathrm{ps},1}(\underline{x}), h_{\mathrm{ps},2}(\underline{x}), ..., h_{\mathrm{ps},l_{\mathrm{x}}}(\underline{x})\big)^{\mathsf{T}} := \underline{G}_{\mathrm{x}}\underline{x} - \underline{h}_{\mathrm{x}} \preceq \underline{0}. \tag{5.3.1}$$

Hard state constraints can now be softened by choosing the following finite horizon cost function with some penalty parameter $\beta > 0$ (e.g., [KM00b; Byr+08]):

$$\tilde{J}_N(\underline{x}_0, \underline{\mathbf{u}}) := J_N(\underline{x}_0, \underline{\mathbf{u}}) + \beta \sum_{k=1}^{N} \sum_{i=1}^{l_{\mathrm{x}}} \max\Big(0, h_{\mathrm{ps},i}\big(\underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{u}})\big)\Big). \tag{5.3.2}$$

The last penalty term in (5.3.2) is continuous but not differentiable at each point. The softened version of OCP (3.1.6) is given by:

$$\tilde{V}_N(\underline{x}_0) := \min_{\underline{\mathbf{u}} \in \mathcal{U}_N(\underline{x}_0)} \tilde{J}_N(\underline{x}_0, \underline{\mathbf{u}}). \tag{5.3.3}$$

Here, $\mathbb{X} = X$ applies in the definition of $\mathcal{U}_N(\underline{x}_0)$. The optimal solution is again denoted by $\underline{\mathbf{u}}^*(\underline{x}_0) \in \mathcal{U}_N(\underline{x}_0)$. Consequently, applying recursive elimination or multiple shooting yields a NLP where hard constraints only ensure the adherence to the input polytope, the terminal region, and the continuity of the state trajectory. Note that the penalty cost function $\tilde{J}_N(\cdot)$ in (5.3.2) is continuous and satisfies Assumption 3.1.2. Therefore, the closed-loop properties of conventional MPC, summarized in Section 3.3, still hold. If there exists a solution to OCP (3.1.6), it is also the solution to OCP (5.3.3). This is an advantage compared to soft constraint formulations where smooth penalty functions superimpose the cost function $J_N(\cdot)$ on the whole parameter space. However, the solution to (5.3.3) is not necessarily the solution to OCP (3.1.6). In other words, the feasible set $\mathcal{X}_N$ is larger with $\mathbb{X} = X$ compared to $\mathbb{X} \subset X$. Since the penalty function $\tilde{J}_N(\cdot)$ is not differentiable at every point, applying derivative-based optimization algorithms is not straightforward. A practical approach for solving general unconstrained optimization problems with an ℓ_1 penalty function approximates the non-smooth function by a linear-quadratic model in the neighborhood of the current parameter vector (e.g., [Fle00]). This procedure is then executed iteratively similar to the SQP method. However, the penalty parameter β significantly affects the convergence rate of the optimization and is thus subject to an iterative parameter update strategy (e.g., [Byr+08]). Obviously, the integration of non-smooth cost functions in conventional MPC leads to

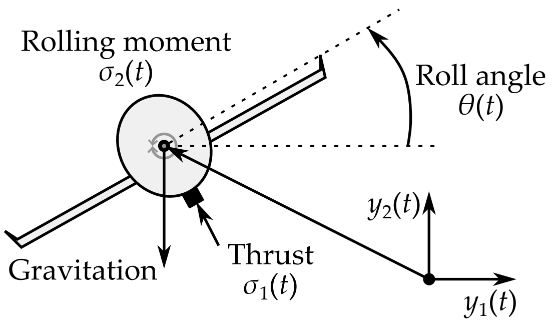


Figure 5.2.: Schematic of the planar vertical takeoff and landing aircraft system adopted from [Sas99, Fig. 10.10.].

an elevated level of implementation and runtime complexity. The softened version of OCP (4.5.1) is defined by:

$$\tilde{V}_N^{\mathrm{s}}(\underline{x}_0, n) := \min_{\underline{u}_{\mathrm{s}} \in \mathbb{P}(\underline{x}_0, n)} \tilde{J}_N\big(\underline{x}_0, \underline{\Theta}_N^{\mathrm{s}}(\underline{u}_{\mathrm{s}})\big) \quad \text{subject to} \quad \underline{\Theta}_N^{\mathrm{s}}(\underline{u}_{\mathrm{s}}) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0). \quad (5.3.4)$$

The optimal solution is again denoted by $\mathbf{\underline{u}}_{\mathrm{s}}^*(\underline{x}_0, n) = \underline{\Theta}_N^{\mathrm{s}}\big(\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)\big) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0)$. When OCP (5.3.4) is subject to a finite set of control vectors with $\mathbb{P}(\underline{x}_0, n) := \mathbb{A}(n)$ or $\mathbb{P}(\underline{x}_0, n) := \tilde{\mathbb{A}}(\underline{x}_0, n)$, the procedure for solving the corresponding OCP follows directly from Algorithm 4.1. Only program line 11 needs to be modified as follows:

$$11: J \leftarrow J + \ell\big(\underline{x}(k), \underline{u}_{\mathrm{s}}\big) + \beta \sum_{i=1}^{l_{\mathrm{x}}} \max\Big(0, h_{\mathrm{ps},i}\big(\underline{x}(k+1)\big)\Big).$$

Here, the penalty parameter β does not have to be adjusted iteratively as in the case of smooth optimization. Assume an arbitrary large $\beta \to \infty$. If OCP (4.5.1) is infeasible, SFMPC simply selects the control candidate that exhibits the lowest violation of constraints and thus the smallest value for $\tilde{J}_N(\cdot)$. The work in [Rat+18] (parallelizable MPC) follows a similar strategy and also chooses, in case of infeasibility, the control sequence that least violates the state constraints. The work in [Ham95] also includes penalty functions that only distort the cost function in case of constraint violations.

5.4. Example: Planar Takeoff and Landing Aircraft

This section introduces the nonlinear planar vertical takeoff and landing aircraft since it is a descriptive system with two inputs that do not simply superimpose on each other. The motivation is to evaluate SFMPC for a multi-input system and to investigate the softened state constraint formulation from Section 5.3. The control task of hovering a real vertical takeoff and landing aircraft can be treated as a low-level control task. Moreover, for this benchmark system, SFMPC clearly outperforms the LQR with control saturation an thus addresses Remark 5.1.1.
The following description of the planar vertical takeoff and landing aircraft is adopted from [Sas99, Sec. 10.4.2]. This system also serves as an MPC benchmark system in [Eng+19]. Figure 5.2 shows the schematic of this planar aircraft. The two-dimensional

position of the center of mass of the planar aircraft at time $t \in \mathbb{R}_0^+$ is denoted by $\big(y_1(t), y_2(t)\big)$. The variable $\theta : \mathbb{R}_0^+ \mapsto \mathbb{R}$ represents the roll angle. The inputs $\sigma_1(t)$ and $\sigma_2(t)$ are the thrust at the bottom of the aircraft and the rolling moment, respectively. Consider the following nonlinear continuous-time mathematical state space model with the state vector $\underline{\chi}(t) = \big(y_1(t), \dot{y}_1(t), y_2(t), \dot{y}_2(t), \theta(t), \dot{\theta}(t)\big)^\mathsf{T}$ [Sas99, Sec. 10.4.2]:

$$\dot{\underline{\chi}}(t) = \begin{pmatrix} \dot{y}_1(t) \\ \ddot{y}_1(t) \\ \dot{y}_2(t) \\ \ddot{y}_2(t) \\ \dot{\theta}(t) \\ \ddot{\theta}(t) \end{pmatrix} = \begin{pmatrix} \chi_2(t) \\ -\sin\big(\chi_5(t)\big)\,\sigma_1(t) + \tilde{\varepsilon}\cos\big(\chi_5(t)\big)\,\sigma_1(t) \\ \chi_4(t) \\ \cos\big(\chi_5(t)\big)\,\sigma_1(t) + \tilde{\varepsilon}\sin\big(\chi_5(t)\big)\,\sigma_2(t) - 1 \\ \chi_6(t) \\ \sigma_2(t) \end{pmatrix}. \tag{5.4.1}$$

The gravitational acceleration is included by subtracting minus one in the fourth row of (5.4.1). The higher the parameter $\tilde{\varepsilon} \in \mathbb{R}_0^+$, the stronger is the coupling between the rolling moment and the lateral acceleration [Sas99]. In this example, the parameter is set to $\tilde{\varepsilon} = 0.1$. For a detailed derivation of the planar aircraft model, refer to [Sas99, Sec. 10.4]. Because of the fast system dynamics, the input of the system $\underline{\sigma}(t)$ is assumed to be piecewise constant on the fixed time grids $t_{k+1} = t_k + T_\mathrm{s}, k \in \mathbb{N}^{[0,N-1]}, t_0 = 0\,\mathrm{s}$ (open-loop) and $t_{n+1} = t_n + T_\mathrm{s}, n \in \mathbb{N}_0, t_0 = 0\,\mathrm{s}$ (closed-loop) with $T_\mathrm{s} = \Delta t_\mathrm{s} = \Delta t_\mathrm{s} = 2^{-5}\,\mathrm{s}$. According to the sampled data formulation in Section 3.4, the internal discrete-time model includes the continuous-time representation (5.4.1) together with the Runge-Kutta method of the fourth order. Because of safety aspects, the roll angle is restricted to $|\theta(t)| \leq \pi/4$, where π represents the circle constant. The constraint sets are thus given by $\mathbb{X} = \{\underline{x} \in X \,|\, |x_5| \leq \pi/4\}$, $\mathbb{X}_\mathrm{f} = \mathbb{X}$, and $\mathbb{U} = \{\underline{u} \in U \,|\, (0,-2)^\mathsf{T} \preceq \underline{u} \preceq (2,2)^\mathsf{T}\}$. Quadratic form cost functions according to (3.1.4) include the weighting matrices $\underline{\underline{Q}} = \underline{\underline{I}}_6$ and $\underline{\underline{R}} = \mathrm{diag}(0.5, 0.5)$. The control task is the transition of the planar aircraft from $\underline{x}_0 = \big(10, 0, 30, 0, 0, 0\big)^\mathsf{T}$ to $\underline{x}_\mathrm{f} = \big(0, 0, 15, 0, 0, 0\big)^\mathsf{T}$ with $\underline{u}_\mathrm{f} = (1, 0)^\mathsf{T}$. SFMPC is configured to solve OCP (4.5.1) repeatedly, using time-varying finite control sets with $\mathbb{P}(\underline{x}_0, n) := \mathbb{A}(n)$ and a cardinality of $c = |\mathbb{A}(\cdot)| = 21^2$. MPC rests upon the solutions to OCP (3.1.6).

The left side of Figure 5.3 shows the evolution of the vertical and horizontal position $y_1(t)$ and $y_2(t)$ of the planar aircraft using different controllers. The black dashed graph visualizes the optimal point-to-point reference solution (open-loop) with a long horizon of $N = 1000$. Obviously, with a horizon of $N = 20$, the closed-loop control performance of MPC deviates from the optimal point-to-point reference solution. However, the MPC controller transfers the planar aircraft to its desired final state $\underline{x}_\mathrm{f}$ while adhering to the limitation of the roll angle as shown in the third plot on the right side of Figure 5.3 with $\chi_{\mu,5}(t) = \theta(t)$. Because of the single degree of freedom in control, SFMPC tilts the planar aircraft more slowly and maintains this aircraft state for a shorter time. As a result, the closed-loop trajectories of MPC and SFMPC differ slightly, which is clearly visible in the evolution of the position of the center of mass in the left phase portrait in Figure 5.3. As noted in the first example 5.2 of this chapter, MPC can exploit the state constraints more effectively compared to SFMPC, meaning that it drives the system to its state constraints faster and operates the system longer at its input limits. The last three plots from above in Figure 5.3 show

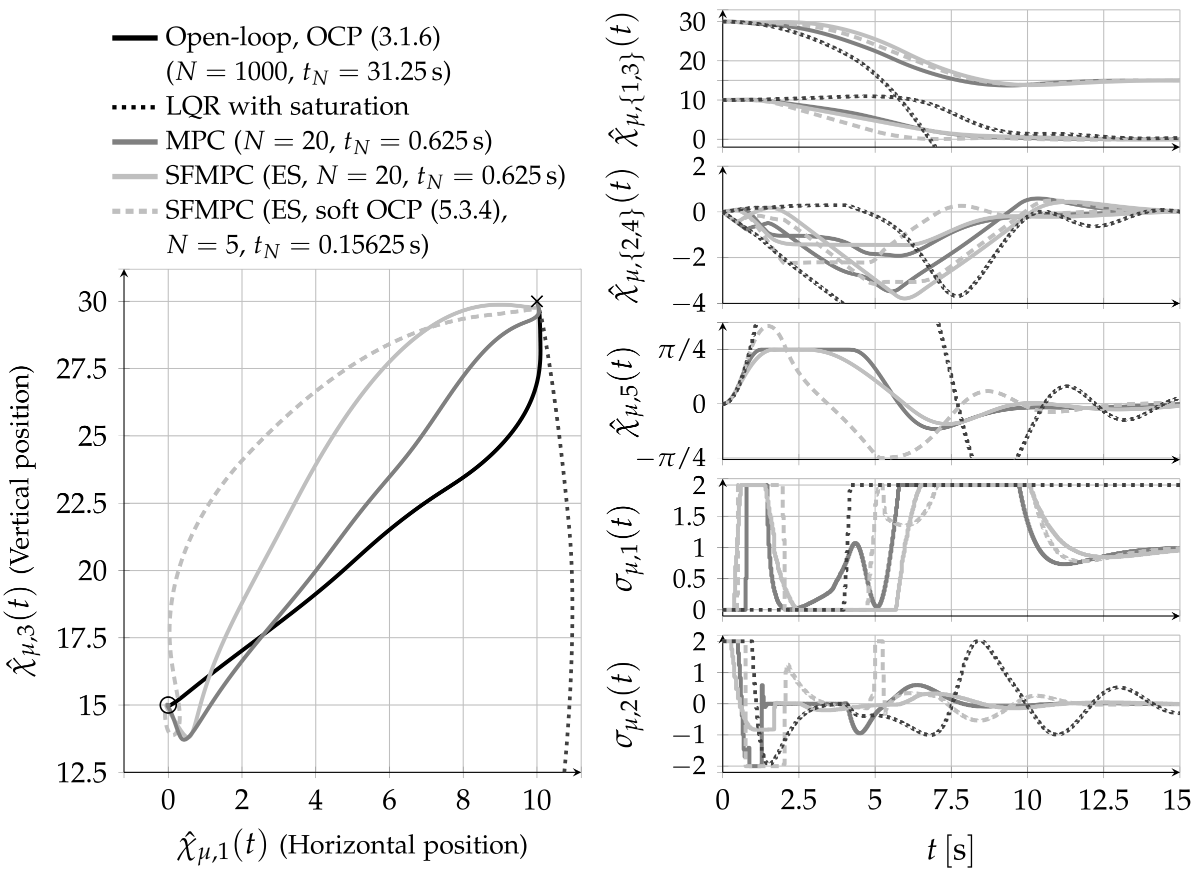


Figure 5.3.: Open- and closed-loop control of the planar vertical takeoff and landing aircraft. Left: Phase portrait. Evolution of the position of the center of mass for different predictive controllers. The cross and the circle denote the start and final positions, respectively. Right: Closed-loop control performance over time.

the constrained state evolution and the two constrained system inputs, highlighting the observed difference of the closed-loop control performance between SFMPC and MPC. However, in this example, both MPC and SFMPC stabilize the steady state $\underset{\sim}{x}_\mathrm{f}$ with the closed-loop system (3.3.2) or (4.5.2). Recall that Remark 5.1.1 claims that there exists a certain similarity of the closed-loop control performances of a saturated LQR with and SFMPC or MPC with a short horizon when implementing the terminal cost function $F\big(\underset{\sim}{\hat{x}}(N)\big) := \underset{\sim}{\hat{x}}^\mathsf{T}(N)\,\underline{P}\,\underset{\sim}{\hat{x}}(N)$. However, in this example, the subsequently input constrained LQR cannot stabilize the planar aircraft model. With the LQR, the planar aircraft clearly violates the maximum roll angle (see third plot on the right) and leaves the admissible position value range ($y_2(t) < 0$, see first plot on the right). For the presented configuration, numerical analysis reveals that SFMPC requires a minimum horizon length of $N = 12$ to ensure experimental recursive feasibility if the implicit control law is driven by the solutions to OCP (4.5.1). However, if SFMPC rests upon the softened OCP (5.3.4), it can transfer the planar aircraft to the steady state $\underset{\sim}{x}_\mathrm{f}$ and stabilize it afterwards with a horizon of only $N = 5$. However, at approximately $t = 1.5\,\mathrm{s}$ the closed-loop system violates the roll angle constraint such that $1/4\,\pi \le \hat{\chi}_{\mu,5}(t) = \theta(t) \le 3/8\,\pi$ applies for a short time period. As described in Section 5.3, outside the feasible set $\mathcal{X}_N^\mathrm{s}$, the softened version of SFMPC with $\beta \to \infty$

Table 5.2.: Evaluation of closed-loop control performances and computation times for (SF)MPC applied to the planar vertical takeoff and landing aircraft model. Abbreviations: Recursive Elimination (RE), Full Discretization (FD), Exhaustive Search (ES).

$N = 20$			IPOPT/SQP			ES (for SFMPC)		
Environment	Solver		MPC RE	MPC FD	SFMPC RE	$c = 11^2$	$c = 21^2$	$c = 31^2$
MATLAB	IPOPT/ES	$\bar{J}_{\text{cl}}$	$-1.08\,\%$	$-1.08\,\%$	$-7.46\,\%$	$-7.42\,\%$	$-7.43\,\%$	$-7.46\,\%$
MATLAB	IPOPT/ES	t_{m}	901.75 ms	**444.29** ms	64.31 ms	83.82 ms	265.45 ms	**554.41** ms
C	SQP/ES	t_{m}	24.89 ms	1.73 s	**213.85** μs	**382.6** μs	1.28 ms	2.68 ms
C	SQP/ES	t_{95}	25.56 ms	1.77 s	**222.45** μs	**415.1** μs	1.55 ms	3.13 ms

selects the control candidate that leads to the lowest state constraint violation during prediction. This example of the planar aircraft demonstrates the functionality of the softened version of SFMPC as a low-level controller.

The quantitative evaluation in Table 5.2 further narrows the scope of SFMPC with the exhaustive search algorithm. Again, the closed-loop control performances are evaluated based on the normalized closed-loop control performance criterion $\bar{J}_{\text{cl}}$ from (4.3.3) with $l = 1000$. Here, the reference represents the initial solution to OCP (3.1.6) with $N = 1000$. The first line of Table 5.2 reveals that with MPC, recursive elimination and full discretization provide (as expected) the same closed-loop control performance with $\bar{J}_{\text{cl}} = -1.08\,\%$. Because of the adaptive and exponential input domain discretization, SFMPC reaches nearly the same closed-loop performance with the exhaustive search algorithm and a cardinality of only $c = 11^2$ as SFMPC with smooth optimization. The error that results from discretization is below 0.05 %. Recall that a direct comparison between MPC and SFMPC is not representative if both controller use the same cost function weights. Evaluating the execution time of the interpreted MATLAB code already indicates that SFMPC is faster with smooth optimization as SFMPC with the exhaustive search algorithm. Again, the IPOPT library demonstrates that it processes sparse optimization problems efficiently such that MPC with full discretization outperforms MPC with recursive elimination and even SFMPC with the exhaustive search algorithm and a cardinality of $c = 31^2$. In this example, the SQP implementation embeds the numerically robust optimization framework $mpcActiveSetSolver(\cdot)$ from MATLAB for solving the underlying QPs. This framework realizes the (QP)KWIK algorithm, which represents an active-set method [SB94]. Though this optimization framework cannot process structure/sparsity information, it is robust to numerical inaccuracies resulting from applying finite differences and Hessian approximations. For a purely dense optimization algorithm, recursive elimination is clearly the preferred method for converting OCP (3.1.6) to an NLP, when considering the resulting optimization run time. Also, when evaluating compiled C code, smooth optimization outperforms the simple exhaustive search in terms of execution time. Hence, regarding computational effort, this example demonstrates that SFMPC is especially suitable for single-input systems with small- to mid-sized state dimensions. For systems with large state dimensions, simultaneous discretization methods provide sparse NLPs, addressing efficient sparse solvers. In contrast, exhaustive search is configured to only process the control candidates resulting from adaptive input domain discretization. However, even in this example, SFMPC with exhaustive search still offers low computational effort, a straightforward implementation according to Algorithm 4.1, and a straightforward procedure for softening state constraints as introduced in Section 5.3.

5.5. Discussion

By systematically comparing the control performances and corresponding computation times of MPC and SFMPC for two common nonlinear benchmark systems, this chapter identifies and narrows the scope of basic SFMPC, exploiting simple derivative-free combinatorial optimization. The design guide for selecting a horizon length elaborated for linear and open-loop stable systems in Section 4.2 and Example 4.3 also applies to state constrained systems. For SFMPC, a short to mid-sized horizon length combined with the cost approximation for an infinite horizon, borrowed from linear system theory, accounts for both a high closed-loop control performance and the experimental compliance with state box-constraints. The resulting closed-loop control performance ranges between the control performance achievable with a subsequently input constrained LQR and conventional MPC. Compared to the LQR, SFMPC explicitly adheres to input and state constraints and considers nonlinear system dynamics. Using the example of the planar aircraft model, SFMPC completes the point-to-point motion and stabilizes the set-point, while the saturated LQR destabilizes the set-point. Because of the single degree of freedom in control, the internal model cannot be operated at its state and input box-constraints as efficiently as with a full degree of freedom in control. As a result, the SFMPC controller is more pessimistic, resulting in a more damped closed-loop control performance compared to MPC. However, for simple state box-constraints, the resulting closed-loop control performance is defensible, especially when considering the straightforward implementation of SFMPC with finite control sets and the exhaustive search algorithm. Since the combinatorial complexity grows exponentially with the input dimension, the runtime statistics clearly show that SFMPC should not be considered for systems with more than two inputs. Obviously, the conventional and not input move-blocked MPC with smooth optimization is more powerful and can also be executed quickly when sparsity and structure information are considered. For this purpose, external optimization libraries, such as IPOPT [WB06], are well-suited. The introduction and evaluation of the presented SQP method should highlight that it is challenging to develop and implement a numerically robust and efficient optimization algorithm from the bottom up. In order to make a completely fair assessment of the realization effort for smooth optimization, the underlying QP solver would also have to be implemented from scratch. Either suitable external libraries are required or a very high effort must be made to realize an efficient optimization. SFMPC with derivative-free optimization is suitable for low-level control when the last two points do not apply. Besides the straightforward implementation and low computational effort, SFMPC allows to implement non-smooth cost functions and system dynamics. The next Chapter 6 makes use of the non-smooth cost function to account for state constraints in the presence of model mismatch and external disturbances.

6 Model Predictive Low-Level Control of Mechatronic Systems

This chapter focuses on the implementation and evaluation of SFMPC as a low-level controller for real systems, namely a directional control valve and a servo-motor. The objective is to cover large operating ranges with small sets of intuitive controller parameters. An important part concerns the development of a real-time capable strategy to compensate the offset since internal models generally suffer from model mismatches. In addition, this chapter demonstrates the clear advantages of derivative-free optimization for industrial applications such as deterministic worst-case computation times and handling of non-smooth objective functions. Parts of this chapter have been published in [Mak+17; Mak+18d; Mak+18e].
The paper [Xu+20] reviews recent algorithms and technologies concerned with the operation of electro-hydraulic control valves. Therein, the authors list the work in [Mak+17] as the only paper dealing with MPC for control valves.
Preliminary note: In general, a continuous signal $S_\mathrm{g}(t)$ is defined to be a function of time $t \in \mathbb{R}_0^+$ with the function definition $S_\mathrm{g} : \mathbb{R}_0^+ \mapsto \mathbb{R}$. Formal definitions of signals that are not directly relevant to the control tasks in this chapter are omitted.

6.1. Directional Control Valve

Complex mechatronic systems like construction and manufacturing machines integrate directional control valves to control hydraulic actuators, which offer fast dynamics and a high force density (e.g., [Ewa+03]). The directional control valve 4WRPEH 6 of the company Bosch Rexroth is a directly operated and fast-responding 4/4-way valve that provides a high control performance [Bos10]. The valve cross-section in Figure 6.1 provides a better understanding of the operating principle. The valve's objective is to route oil from the pressure port P to the control ports A or B. An electromagnetic actuator exerts force by reducing the air gap inside the solenoid. The moving armature adjusts the position of the guided piston and thus the opening cross sections of the hydraulic ports. As soon as the valve partially opens a working port, pressure relief to the tank T occurs on the other side. A spring on the left outside of the valve generates the restoring force and handles the fail-safe position in case of power failure. The

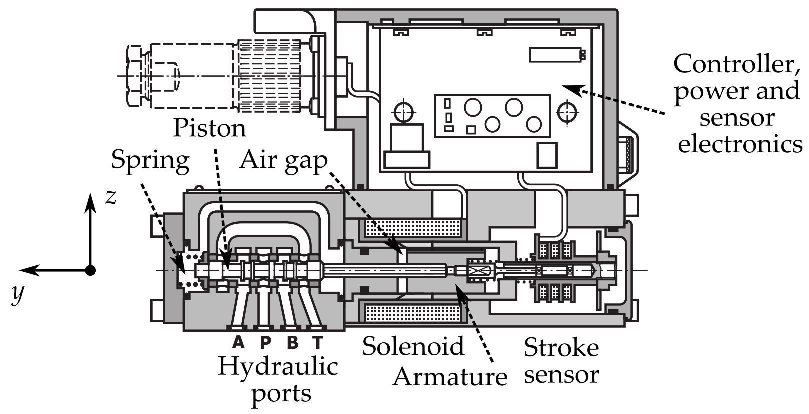


Figure 6.1.: Cross-section of the directional control valve 4WRPEH 6 [Bos10]. The dashed arrows and the corresponding labels are added to indicate important valve components.

position of the piston and therefore of the solenoid stroke $y_\mathrm{s}(t)$ and the pressure $\bar{p}(t)$ at port P mainly define the oil flow rate $Q(t)$. An internal inductive sensor measures the position of the armature and allows position control, or more specifically, stroke control. The objective of such a position controller is to compensate parasitic effects such as nonlinear friction and magnetic and mechanical hysteresis, which would cause positioning inaccuracies in an open-loop control setup (see, e.g., [JK12]). For detailed information on directly operated valves, refer, for example, to [Ewa+03]. The block diagram in Figure 6.2 summarizes the valve's working principle and shows the most important components that mainly define the dynamical behavior of the valve. In the stationary case, the force generated by the proportional solenoid $F_\mathrm{s}(t)$ is almost independent of the armature position $y_\mathrm{s}(t)$. In the stroke range of interest, the force is almost proportional to the excitation current $i_\mathrm{s}(t)$. However, for fast responding valves, eddy currents are not negligible. The temporal variation of the armature position and the time-varying excitation generate eddy currents inside the conductive components. These eddy currents induce opposing magnetic fields and slow down the force build-up (e.g., [Kal+12, Sec. 5.2.4]). The one dimensional motion of all sliding components along the y-axis is similar to the motion of a mass-spring-damper system. However, the forces caused by nonlinear friction effects counteract the driving force $F_\mathrm{s}(t)$. In addition, as soon as oil flows through the valve, a hydraulic flow force $F_\mathrm{h}(t)$ applies to the piston. A reliable control approach for directional control valves rests upon a cascaded structure with an inner current controller enclosed by a complex position controller (e.g., [Ott04; Kre+07]). The current controller processes the reference signal from the position controller $i_\mathrm{ref}(t)$ and the measured current signal $i_\mathrm{m}(t)$. The native power electronics supply a tailored pulse width modulated voltage signal $u_\mathrm{p} : \mathbb{R}_0^+ \mapsto \{-24\,\mathrm{V}, 24\,\mathrm{V}\}$, which minimizes power losses and increases valve dynamics. For more information on driving an electromagnetic valve actuator with a pulse width modulated voltage signal, refer to [Lau90]. In addition, the native electronics are optimized for a compact and robust housing. However, due to customized power electronics and low-level controller, the identification of a continuous state space representation with a small to mid-sized state dimension is a challenging task [Lau90; Mak+15a; Mak+15c]. To meet the high demands placed on the closed-loop control

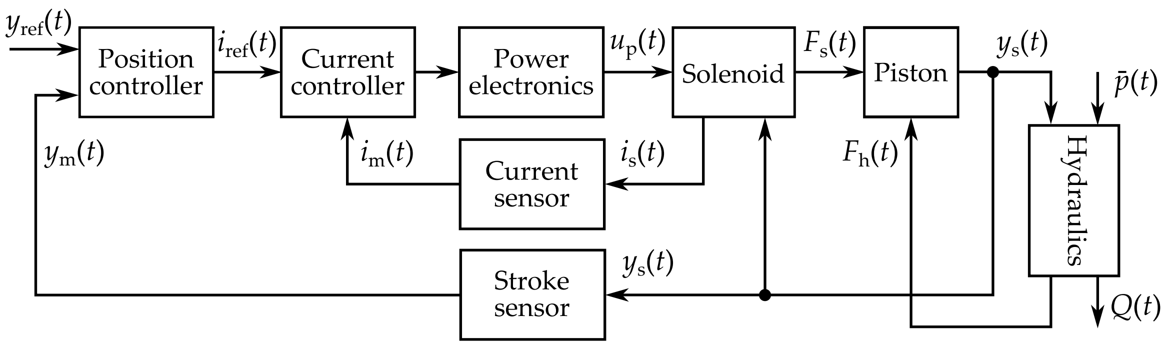


Figure 6.2.: Operating principle of the directional control valve 4WRPEH 6. The power electronics generate a pulse width modulated voltage signal $u_{\mathrm{p}}(t)$.

performance on both the small as well as the large signal range, an augmented proportional-integral position controller is extended by an additional state feedback structure. The proportional and integral gains are defined by piecewise linear functions, where the grid points are subject to optimization (related to gain scheduling). Therefore, the controller amplifications depend on the current position error $y_{\mathrm{ref}}(t) - y_{\mathrm{m}}(t)$, where $y_{\mathrm{ref}} : \mathbb{R}_0^+ \mapsto \mathbb{R}$ and $y_{\mathrm{m}} : \mathbb{R}_0^+ \mapsto \mathbb{R}$ represent the set-point and measured position signals, respectively. In total, the native position controller has 24 free parameters (e.g., [Ott04; Kre+07]). To speed up the controller design, automated tuning of free controller parameters relies on an elaborate evolutionary hardware-in-the-loop optimization [Nic+01; Kre+07]. In [Kre+09], this automated tuning is extended to a multi-objective optimization strategy that further incorporates expert knowledge.

6.1.1. Experimental Setup

The experimental setup in this chapter takes a step away from the series product to facilitate system identification. This section aims at replacing the entire native control concept of the valve by a single predictive controller. However, the detailed integration of MPC into series-ready valve electronics is beyond the scope of this dissertation. Figure 6.3 illustrates the experimental setup for the evaluation of MPC. A performance real-time target machine (Speedgoat, Intel Core i7 with 4.2 GHz, 16bit analog I/O resolution, IO323) implements different algorithms at a fixed sampling time $\Delta t_{\mathrm{s}} \in \mathbb{R}^+$. Let $t_{n+1} = t_n + \Delta t_{\mathrm{s}}$ denote the resulting fixed time grid with $n \in \mathbb{N}_0^+$ and $t_0 = 0\,\mathrm{s}$. Instead of the native power electronics, a high-speed four-quadrant amplifier (Toellner 7610-40) directly drives the solenoid without applying a pulse width modulation. However, the power amplifier approximately provides the same electrical power as the native valve electronics. The real-time target machine closes the loop by reading the sensor information of the native sensor electronics. The host PC is used to develop a control or data logging algorithm, which is then transferred to the real-time target machine to enable real-time computing. During real-time operation, the communication between the host PC and the target machine is based on a TCP/IP Ethernet connection. This asynchronous communication enables quasi-online tuning of controller parameters and processing of measured and commanded signals. The valve is filled with oil and a locking plate seals all hydraulic ports mechanically at

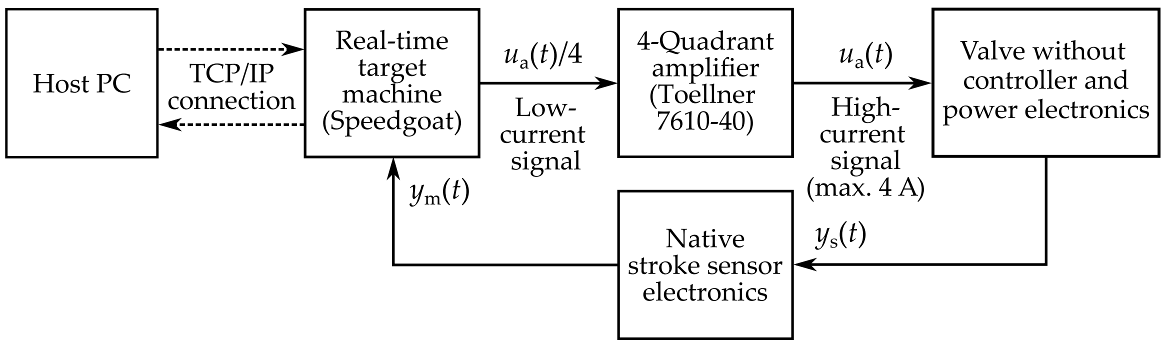

Host PC
TCP/IP connection
Real-time target machine (Speedgoat)
$u_\mathrm{a}(t)/4$
Low-current signal
4-Quadrant amplifier (Toellner 7610-40)
$u_\mathrm{a}(t)$
High-current signal (max. 4 A)
Valve without controller and power electronics
$y_\mathrm{m}(t)$
Native stroke sensor electronics
$y_\mathrm{s}(t)$

Figure 6.3.: Experimental setup for evaluating model predictive valve control. The amplifier generates an analog voltage signal $u_\mathrm{a}(t)$ (not pulse width modulated).

the bottom of the valve. Since there is no oil flow, there are also no flow forces acting on the piston. Clearly, experimental robustness needs to be evaluated when hydraulic actuators are connected to the valve. The sensor output of $y_\mathrm{m}(t) = 0\,\%$ denotes the center piston position in which both hydraulic control ports A and B are closed. At this piston position, no oil can flow through the valve, even when the valve is unsealed from below and a hydraulic actuator is connected. The sensor output of $y_\mathrm{m}(t) = \pm 100\,\%$ shows that one working port A or B is fully opened while the other is connected to the tank port T. In the following, a linear differential equation of the third order is used to approximate the input-output behavior of the valve between the analog input voltage $u_\mathrm{a} : \mathbb{R}_0^+ \mapsto [-24\,\mathrm{V}, 24\,\mathrm{V}]$ and the measured position $y_\mathrm{m}(t)$. A linear model is chosen for three reasons. The major reason is that a linear model enables the implementation of linear MPC (e.g., [Bor+17]), which then can serve as a real-time capable reference approach. Furthermore, a linear model facilitates the observer and offset compensation design. Since the mass-spring-damper system is already a second-order system, a third-order model is at least necessary to account for additional dynamical electromagnetic effects. Let $\sigma : \mathbb{R}_0^+ \mapsto \mathbb{R}$ and $y : \mathbb{R}_0^+ \mapsto \mathbb{R}$ denote the input (voltage) and output (position) signals, respectively. The differential equation $\dddot{y}(t) + a_2\ddot{y}(t) + a_1\dot{y}(t) + a_0\ddot{y}(t) = b_0\sigma(t)$ is transformed into the following state space representation:

$$\begin{pmatrix} \dot{y}(t) \\ \ddot{y}(t) \\ \dddot{y}(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{pmatrix} \begin{pmatrix} y(t) \\ \dot{y}(t) \\ \ddot{y}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ b_0 \end{pmatrix} \sigma(t). \tag{6.1.1}$$

The state vector is denoted by $\underline{\chi}(t) = \big(y(t), v(t) = \dot{y}(t), a(t) = \ddot{y}(t)\big)^\mathsf{T}$, where $v(t)$ and $a(t)$ represent the (stroke) velocity and acceleration, respectively. The output of the state space model is defined by $y(t) = \underline{C}\,\underline{\chi}(t) = (1, 0, 0)\,\underline{\chi}(t)$. The open-loop identification involves a carefully designed amplitude modulated random binary signal (ARBS) to prevent the solenoid armature from hitting the mechanical stops, since the stroke only ranges between a few millimeters. After subtracting the individual mean values of the input voltage stimulus and the measured position signal, optimization-based identification returns values for $a_0, a_1, a_2, b_0 \in \mathbb{R}^+$. For reasons of anonymization, the exact values are not given in this dissertation. However, the identified model is open-loop stable and responds with a decaying oscillation. Figure B.2 gives an insight into the valve behavior and compares the measured and simulated system responses to the designed input stimulus. The model performance achieves a fitness value of $NRMSE = 9.69\,\%$ during identification and a fitness value of $NRMSE = 12.43\,\%$ for the subsequent evaluation of the validation signal (see App. B.4). Note that a linear model cannot reproduce nonlinear friction effects such as static friction. Static friction occurs when the piston crosses the zero velocity and remains in a constant position, at least for a short time. The native valve electronics prevent the stick slip effect (e.g., [Fee+98]) during closed-loop control by superimposing a high frequency dither current signal. The experimental setup in Figure 6.3 does not include a fast-responding current source. Therefore, MPC and SFMPC prevent the stick slip effect, as far as possible, by enforcing a dynamic closed-loop behavior. For real-time operation, the continuous-time representation (6.1.1) is transformed into the discrete-time formulation $\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{B}u(k)$ by applying zero-order-hold discretization as described in (3.4.9). Since the continuous- and discrete-time model coincide at discrete time steps, the following relationship applies with $k \in \mathbb{N}_0, t_0 = 0\,\text{s}$, and $t_{k+1} = t_k + \Delta t_\text{s}$:

$$\begin{pmatrix} y(t_k + \Delta t_\text{s}) \\ v(t_k + \Delta t_\text{s}) \\ a(t_k + \Delta t_\text{s}) \end{pmatrix} = \underline{A} \begin{pmatrix} y(t_k) \\ v(t_k) \\ a(t_k) \end{pmatrix} + \underline{B}\,\sigma(t_k). \tag{6.1.2}$$

The linear and continuous-time system in (6.1.1) and its discretized version in (6.1.2) are both controllable and observable for $\Delta t_\text{s} = 100\,\mu\text{s}$ and $\Delta t_\text{s} = 150\,\mu\text{s}$.

6.1.2. Offset and Computation Time Compensation

As it is well-known from state space control, model mismatch results in a closed-loop control offset, which needs to be compensated. Offset-free model predictive control follows the idea of first augmenting the state space representation by a disturbance model. In the second step, an observer estimates the states as well as the disturbances (e.g., [MB02; PR03; Mae+09; MM12]). A detailed analysis and comparison of common approaches for offset-free tracking MPC is given in [Pan15]. This section follows a straightforward offset compensation procedure by simply exploiting the system structure. Since the second and the third state represent time derivatives of the first state, all steady states $\underline{x}_\text{f}$ are defined by $\underline{x}_\text{f} := \big(y_\text{ref}(t_n), 0, 0\big)^\intercal$ for all admissible $y_\text{ref}(t_n)$. Here, an admissible reference position $y_\text{ref}(t_n)$ lies between the mechanical limits of the valve. The reference input u_f follows from solving the steady state equation $\underline{x}_\text{f} = \underline{A}\,\underline{x}_\text{f} + \underline{B}\,u_\text{f}$ for u_f. To hold the piston at a constant position, the real valve also assumes a stroke velocity and acceleration of zero. Hence, steady state offsets only occur in the position signal. The proposed offset compensation procedure first implements a polynomial filter of the third order with the model function $y_\text{ls}(t_n) := \zeta_1 t_n^2 + \zeta_2 t_n + \zeta_3$ and $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{R}$. At every closed-loop time instant, the polynomial filter solves the least squares problem with the window length $W \in \mathbb{N}$:

$$\text{Step I: } (\zeta_1^*, \zeta_2^*, \zeta_3^*) := \arg \min_{\zeta_1, \zeta_2, \zeta_3} \sum_{l=0}^{W-1} \big(y_\text{m}(t_{n-l}) - y_\text{ls}(t_{n-l})\big)^2 \;\rightarrow\; v_\text{ls}(t_n) := 2\zeta_1^* t_n + \zeta_2^*. \tag{6.1.3}$$

For the time interval $t \in [0, W\Delta t_\mathrm{s})$, the measured output $y_\mathrm{m}(t)$ is assumed to be zero. The larger the time window, the better the smoothing of sensor noise. However, as the window length W increases, the phase error becomes larger. Therefore, this parameter W is subjected to a heuristic tuning. In the following experiments, only a short window length of $W \in \{9, 12\}$ is used to suppress high frequency noise without inducing a noticeable phase error. Finally, the estimated stroke velocity follows by $v_\mathrm{ls}(t_n) := 2\zeta_1^* t_n + \zeta_2^*$. Then, this estimated velocity signal is used to further estimate the position and the acceleration with the help of a linear observer:

$$\text{Step II:} \begin{pmatrix} \hat{y}(t_n) \\ \hat{v}(t_n) \\ \hat{a}(t_n) \end{pmatrix} = \underline{A} \begin{pmatrix} \hat{y}(t_n - \Delta t_s) \\ \hat{v}(t_n - \Delta t_s) \\ \hat{a}(t_n - \Delta t_s) \end{pmatrix} + \underline{B}\, \sigma(t_n - \Delta t_s) + \underline{L}\big(v_\mathrm{ls}(t_n) - v(t_n)\big). \quad (6.1.4)$$

The observer gain matrix $\underline{L}$ is determined by pole placement such that the eigenvalues of $\underline{A} - \underline{L}\,\underline{C}$ are strictly inside the unit circle. The difference $d_\mathrm{y}(t_n) := y_\mathrm{m}(t_n) - \hat{y}(t_n)$ reveals information about the current model error since the observer and the internal model are based on the same state space representation (6.1.2). Recall that the velocity signals are not affected by the position offset error. As proposed in [Tat07, Sec. 3.4.2], this information $d_\mathrm{y}(t_n)$ can be directly used for offset compensation. First, assume that the error $d_\mathrm{y}(t_n)$ is constant over the entire prediction horizon (e.g., [Pan15]). Instead of adding the estimated disturbance to the state space model (6.1.2), the following offset compensation scheme adjusts the position set-point by the estimated disturbance $d_\mathrm{y}(t_n)$:

$$\text{Step III: } \tilde{y}_\mathrm{f}(t_n) := y_\mathrm{ref}(t_n) - \big(y_\mathrm{m}(t_n) - \hat{y}(t_n)\big) = y_\mathrm{ref}(t_n) - d_\mathrm{y}(t_n). \quad (6.1.5)$$

For this adjustment step, the linear system of equations describing the behavior of the system in a steady state has to be solved:

$$\begin{aligned} \text{Step IV: } \tilde{\underline{x}}_\mathrm{f}(n) &:= \begin{pmatrix} \tilde{y}_\mathrm{f}(t_n) \\ 0 \\ 0 \end{pmatrix} = \underline{A} \begin{pmatrix} \tilde{y}_\mathrm{f}(t_n) \\ 0 \\ 0 \end{pmatrix} + \underline{B}\, \sigma(t_n), \\ \tilde{u}_\mathrm{f}(n) &:= \sigma(t_n) = \underline{B}^{\#}\big(\tilde{\underline{x}}_\mathrm{f}(n) - \underline{A}\, \tilde{\underline{x}}_\mathrm{f}(n)\big). \end{aligned} \quad (6.1.6)$$

The offset-compensation technique, as outlined in (6.1.6), yields some time-varying virtual reference pair $\big(\tilde{\underline{x}}_\mathrm{f}(n), \tilde{u}_\mathrm{f}(n)\big) := \big((\tilde{y}_\mathrm{f}(t_n), 0, 0)^\mathsf{T}, \sigma(t_n)\big)$ although the set-point $y_\mathrm{ref}(t_n) = y_\mathrm{f} \in \mathbb{R}$ might be constant on some time interval.
Besides the offset-compensation, a predictive controller needs to compensate non-negligible computation times. In contrast to the simulation, where a computation time of zero is assumed, during real-time operation, the finite time for solving a NLP affects closed-loop control performance. The handling of finite computation times in MPC was first analyzed in [Che+00; FA04]. Both contributions develop computation time compensation techniques that rest upon stabilizing terminal ingredients. The RTI scheme also compensates finite computation times by distributing a tailored SQP approach over two consecutive sampling intervals [Die+05b]. In this dissertation, a straightforward computation time compensation procedure accounts for asymptotically constant references that can be switched during real-time operation (see

Rem. 3.3.4). This procedure solves the OCP respectively NLP for the closed-loop time instant $n+1$ at time instant n similar to [FA04]. Since the sampled-data formulation links the solutions of the discrete-time and continuous-time domain, the time between two time instances n and $n+1$ is well-defined by Δt_s and is used to solve the OCP. Let $\underline{e} : \mathbb{N}_0 \mapsto \mathbb{R}^p$ and $\underline{w} : \mathbb{N}_0 \mapsto \mathbb{R}^p$ be some bounded non-deterministic measurement and modeling errors, respectively. The measurable or observable state vector at the next time instant is defined by (see [All+17]):

$$\hat{\underline{x}}_\mu(n+1) := \underline{x}_\mu(n+1) - \underline{e}(n+1) = \underline{f}\Big(\underline{x}_\mu(n), \underline{\mu}\big(\underline{x}_\mu(n)\big)\Big) + \underline{w}(n) - \underline{e}(n+1). \quad (6.1.7)$$

Then, the closed-loop state vector at the next time instant $\underline{x}_\mu(n+1)$, which initializes the OCPs (3.1.6), (4.5.1), and (5.3.4), is not known exactly in advance at time instant n. However, the compensation procedure approximates the next closed-loop state vector by a one-step prediction $\underline{x}_\mu(n+1) \approx \underline{f}\big(\hat{\underline{x}}_\mu(n), \underline{\mu}\big(\underline{x}_\mu(n)\big)\big)$. At the very first iteration $n=0$, the initial control vector is defined by $\underline{\mu}\big(\underline{x}_\mu(0)\big) := \underline{0}_m$. In the setup of valve control, the estimated state vector is defined by $\hat{\underline{x}}_\mu(n) := \big(\hat{y}(t_n), \hat{v}(t_n), \hat{a}(t_n)\big)^\mathsf{T}$.

6.1.3. Controller Design

The following model predictive controller directly applies the discrete-time state space representation $\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{B}u(k)$ as the internal model, which results from zero-order hold discretization with $\Delta t_\mathrm{s} = 100\,\mu\mathrm{s}$ or $\Delta t_\mathrm{s} = 150\,\mu\mathrm{s}$. The coordinate transformations $\breve{\underline{x}}(k) := \underline{x}(k) - \tilde{\underline{x}}_\mathrm{f}(n)$ and $\breve{\underline{u}}(k) := \underline{u}(k) - \tilde{u}_\mathrm{f}(n)$ account for the time-varying reference pair $\big(\tilde{\underline{x}}_\mathrm{f}(n), \tilde{u}_\mathrm{f}(n)\big)$ at closed-loop time n. The quadratic cost functions are now given by $\ell\big(\breve{\underline{x}}(k), \breve{\underline{u}}(k)\big) = \|\breve{\underline{x}}(k)\|^2_{\underline{Q}} + \|\breve{\underline{u}}(k)\|^2_{\underline{R}}$ and $F\big(\breve{\underline{x}}(N)\big) = \|\breve{\underline{x}}(N)\|^2_{\underline{P}}$ with $\underline{Q} = \mathrm{diag}(q_1, q_2, q_3)$, $\underline{R} = r_1$, and $q_1, q_2, q_3, r_1 \in \mathbb{R}^+$. Because of the linearity of the internal model, the Riccati equation only needs to be solved once to determine the matrix $\underline{P}$. Although nominal theoretical stability cannot be guaranteed in this chapter due to input move-blocking, the solution of the Riccati equation is nevertheless used to approximate the infinite horizon costs. The constraint sets are given by $\mathbb{X} = X$ or $\mathbb{X} = \{\underline{x} \in X \,|\, |x_2| \leq 2000\}$, $\mathbb{X}_\mathrm{f} = \mathbb{X}$, and $\mathbb{U} = \{u \in U \,|\, |u| \leq 24\}$. In the next subsection, linear MPC (LMPC) serves as the reference control approach. The transformation of OCP (3.1.6) into a QP requires the constraint sets $\mathbb{X}, \mathbb{X}_\mathrm{f}$, and $\mathbb{U}$ to be polyhedral sets, which is satisfied with input and state box-constraints. For more information on LMPC, refer, for example, to [Bor+17]. To ensure a numerically stable real-time operation, LMPC operates at a higher sampling time of $\Delta t_s = 150\,\mu\mathrm{s}$ ($f_\mathrm{s} \approx 6.7\,\mathrm{kHz}$) with a filter window length of $W = 9$. The LMPC implementation uses the numerically robust framework *mpcActiveSetSolver*$(\cdot)$ from MATLAB to solve the recurring QPs. The real-time target machine is configured to tolerate a pre-defined number of real-time violations. In this case, the real-time machine waits for the QP solver to terminate and then proceeds again at its fixed sampling time. The QP solver either terminates in case of convergence or if it reaches its maximum number of optimization iterations (suboptimal or infeasible solution). However, the LMPC implementation uses the first control value that results at the time the QP solver terminates for closed-loop control. SFMPC, in contrast, solves OCP (5.3.4) with $\mathbb{P}(\underset{\sim}{x}_0, n) := \mathbb{A}(n)$ recurrently at a closed-loop sampling time of $\Delta t_\mathrm{s} = 100\,\mu\mathrm{s}$ ($f_\mathrm{s} = 10\,\mathrm{kHz}$) and uses a filter window length of $W = 12$. SFMPC further builds on the exponential discretization in (4.5.3) with a cardinality of $|\mathbb{A}(\cdot)| = 101$. This cardinality is chosen generously, the available execution time would even allow for a much higher value. However, the high closed-loop control performance is preserved even with a significantly lower cardinality (see Fig. 4.4). If the objective is to limit the opening speed of working ports at $v_\mathrm{lim} = 2000\,\%\,\mathrm{s}^{-1}$, the constraint state space is given by $\mathbb{X} = \{\underset{\sim}{x} \in X \,|\, |x_2| \leq 2000\}$. In simulation, hence for the nominal case, both approaches require at least a horizon of $N = 3$ to ensure experimental recursive feasibility with a velocity limit of $v_\mathrm{lim} = 2000\,\%\,\mathrm{s}^{-1}$. During real-time operation, both SFMPC and LMPC implement a horizon of only $N = 5$. The maximum velocity that the piston can reach is approximately $v_\mathrm{max} \approx 16000\,\%\,\mathrm{s}^{-1}$. Note that a demanded accuracy of only 1 % already justifies the high closed-loop control frequency around $f_\mathrm{s} = 10\,\mathrm{kHz}$. The series valve even provides a higher accuracy. For both approaches, the controller design is based on manual tuning of the free parameters q_1, q_2, q_3, r_1. During parameterization, a compromise is made between the criteria rise time, settling time, and overshoot with respect to the position signal. The following experiments are not about determining the Pareto set from which the control engineer can choose an individual (cf. [Kre+09]). The idea is to achieve a high control performance in just a few tuning steps and then to compare the closed-loop performances qualitatively.

6.1.4. Performance Evaluation

The upper plot in Figure 6.4 compares the tracking performances of LMPC and SFMPC for some piecewise constant reference vectors $\underset{\sim}{x}_\mathrm{f} := \big(y_\mathrm{ref}(t), 0, 0\big)^\intercal$. Note that in the following plots, all signals that are actually defined on the fixed time grid $t_{n+1} = t_n + \Delta t_\mathrm{s}$ are connected continuously such that the continuous time variable $t \in \mathbb{R}_0^+$ is introduced on the x-axis. In addition, the x-axis is always normalized to the time duration of the presented experiment $t_\mathrm{max} \in \mathbb{R}^+$ such that $\bar{t} := t/t_\mathrm{max}$ applies. Without a velocity limit, the step responses are almost identical. Both approaches provide short rise times, low overshooting, and smooth position signals. With the proposed offset-compensation strategy, the offset is less than one percent. The small subplot shows, exemplary for $\underset{\sim}{x}_\mathrm{f} = (50, 0, 0)^\intercal$, the close-up of the performance in the local neighborhood of the steady state. As it can be seen from the second plot in Figure 6.4, LMPC and SFMPC operate the solenoid at its input limits every time another position reference $y_\mathrm{ref}(t) = y_\mathrm{f}$ is commanded. The third plot in Figure 6.4 shows the evolution of the virtual reference $\tilde{y}_\mathrm{f}(t)$. For steering the real valve to the commanded reference $y_\mathrm{ref}(t)$, the linear model needs to be transferred to $\tilde{y}_\mathrm{f}(t)$. Both the qualitative difference and the different value ranges between $\tilde{y}_\mathrm{f}(t)$ in the third plot and $y_\mathrm{ref}(t)$ in the first plot of Figure 6.4 are due to model mismatch. The model has to be driven around $\tilde{y}_\mathrm{f}(t) = 450\,\%$ such that the steady state equation (6.1.6) yields a reference value $\tilde{u}_\mathrm{f}(n) := \sigma(t_n)$ for all $n \in \mathbb{N}_0$ that is high enough to hold the real valve piston at some constant position level. As soon as the real valve piston position is in some close neighborhood of the reference position $y_\mathrm{ref}(t) = y_\mathrm{f}$, the offset compensation

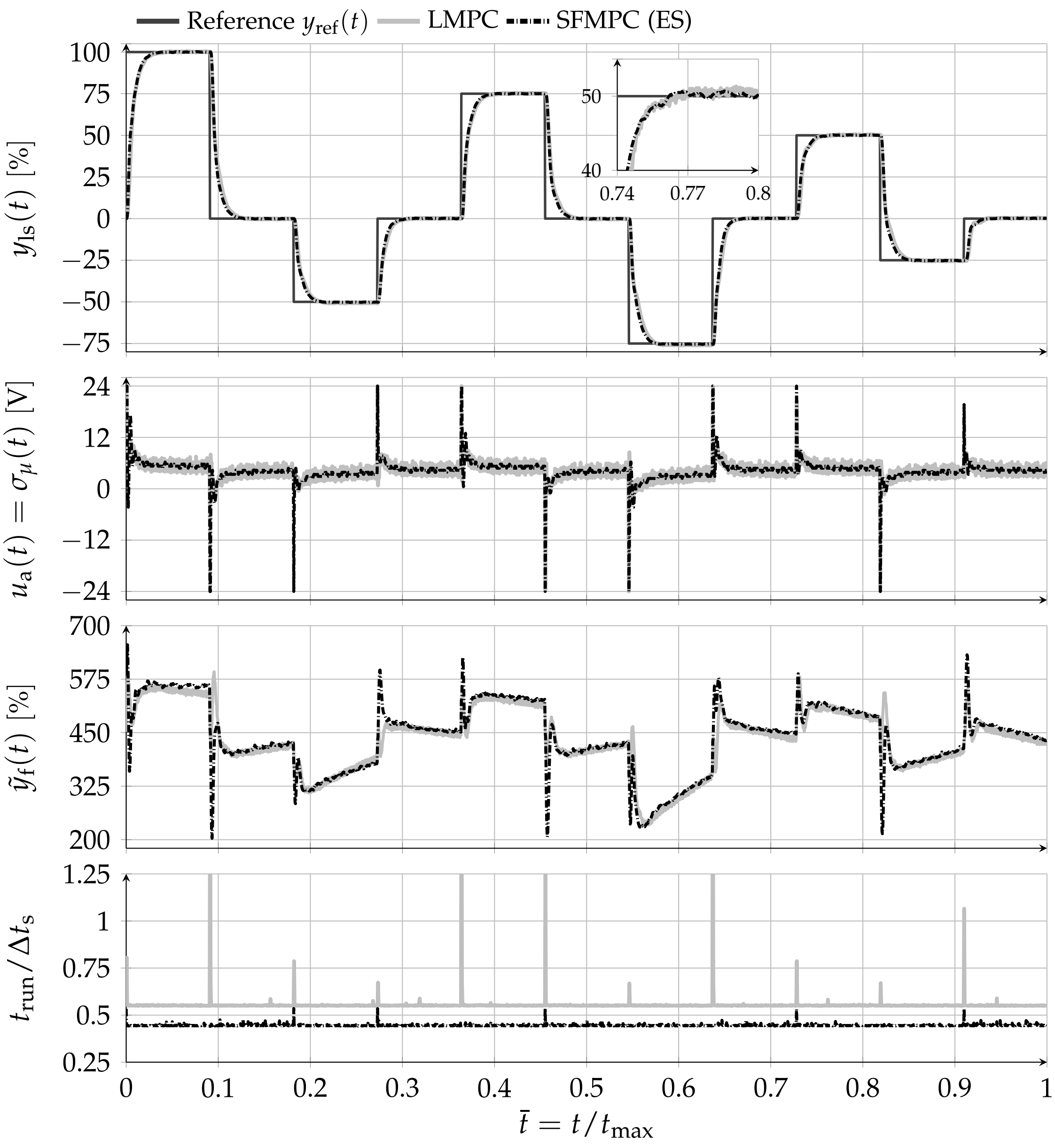


Figure 6.4.: Valve closed-loop real-time control: Constant reference tracking with an **inactive** stroke **velocity limit**. First: Stroke time performance. Second: Solenoid input voltage. Third: Reference adaptation due to offset compensation. Bottom: Normalized execution time.

scheme increases or decreases the virtual reference such that the valve reaches its final position (similar to an integral controller). The bottom plot in Figure 6.4 visualizes the normalized execution time $t_{\text{run}}/\Delta t_{\text{s}}$, where $t_{\text{run}} \in \mathbb{R}^{+}$ is the measured run time. SFMPC performs at a nearly constant task execution time level and never violates the real-time constraints. LMPC also operates at an almost constant task execution time level. However, some outliers clearly violate the real-time constraints. Variations in execution times with smooth optimization occur due to measurement and model errors. For example, the warm-starts of primal and also dual variables might be poor such that the optimizer requires a longer time to converge. As soon as the reference position $\tilde{y}_{\text{f}}(t)$ changes its value, the QP solver cannot warm-start the active inequalities.

As noted in Remark 3.3.4, switching the reference is similar to a cold-start, which potentially involves the highest computational overhead. Figure B.7 and Figure B.8 in Appendix B.6 show the closed-loop control performance of LMPC and SFMPC for another reference profile that also forces the piston to cross the zero position.

The major advantage of SFMPC and also LMPC over the native valve control concept is the significant reduction of parameter complexity. At a comparable closed-loop control performance, the number of free controller parameters reduces from 24 to initially five parameters. The horizon length N can be determined during simulation considering recursive feasibility aspects. Since only the main diagonal of the weighting matrix $\underline{Q}$ is used, the weight q_1 can be set to $q_1 = 1$. Then, the weights q_2 and q_3 can be tuned in the sense of a relative weighting. Here, the weights q_2 and q_3 penalize the velocity and acceleration of the piston, respectively, while r_1 penalizes the control input. This clear relationship between the remaining three weighting parameters and the control performance enables intuitive online controller design. This tuning procedure addresses industrial applications in particular. SFMPC outperforms LMPC from the perspective of implementation. While LMPC requires a numerically robust QP solver, SFMPC comprises only a few lines of code, as shown in Algorithm 4.1.

Another major advantage of SFMPC over LMPC is its ability to directly handle non-smooth cost functions, such as those that arise when state constraints are softened (see Sec. 5.3). Figure 6.5 shows a similar experiment as in Figure 6.4 but with the active velocity limit $v_{\text{lim}} = 2000\,\%\,\text{s}^{-1}$. The third plot results from applying forward finite differences to some carefully filtered position signal. Since the polynomial filter in (6.1.3) is only designed to suppress high-frequency noise of the stroke sensor, the position signal $y_{\text{ls}}(t_n)$ is further subject to a zero-phase offline filtering procedure (see *filtfilt*$(\cdot)$ in MATLAB). This procedure passes the position signal through a low-pass filter with some cutoff frequency in forward and backward direction such that no phase error occurs. The resulting smooth position signal then serves as the basis for numerical differentiation. The velocity $v_{\text{filt}} : \mathbb{R}_0^+ \mapsto \mathbb{R}$ is finally considered as the ground truth signal. Since the polynomial filter involves only a small window length W to avoid large phase errors, the velocity profile $v_{\text{filt}}(t_n)$ is much smoother than $v_{\text{ls}}(t_n)$. Offline filtering, in contrast, is suitable for suppressing even lower-frequency noise without distorting the phase evolution. The third plot in Figure 6.5 visualizes that the velocity signals of LMPC and SFMPC clearly violate the velocity constraint. This violations occur because of the present model mismatch and the limited inherent robustness of the nominal MPC setting (see Def. 3.3.1). If there exist upper bounds for the model uncertainties and external disturbances, the controller design of LMPC needs to follow the (non-trivial) guidelines of robust linear MPC (e.g., [Bor+17, Ch. 15]). In [Zei+14], for example, the authors combine a soft constraint approach with robust LMPC and prove important closed-loop stability properties. The first plot and its subplot in Figure 6.5 show a non-smooth position signal for LMPC. Hence, when applying nominal LMPC with hard constraints for valve control, the tracking performance drops when constraint violations occur. The execution time for LMPC in the bottom plot fluctuates and violates the real-time constraints as soon as the piston velocity approaches the velocity limit. With SFMPC, however, the valve features an almost constant opening and closing behavior of the working ports and no deviation in steady state. This behavior

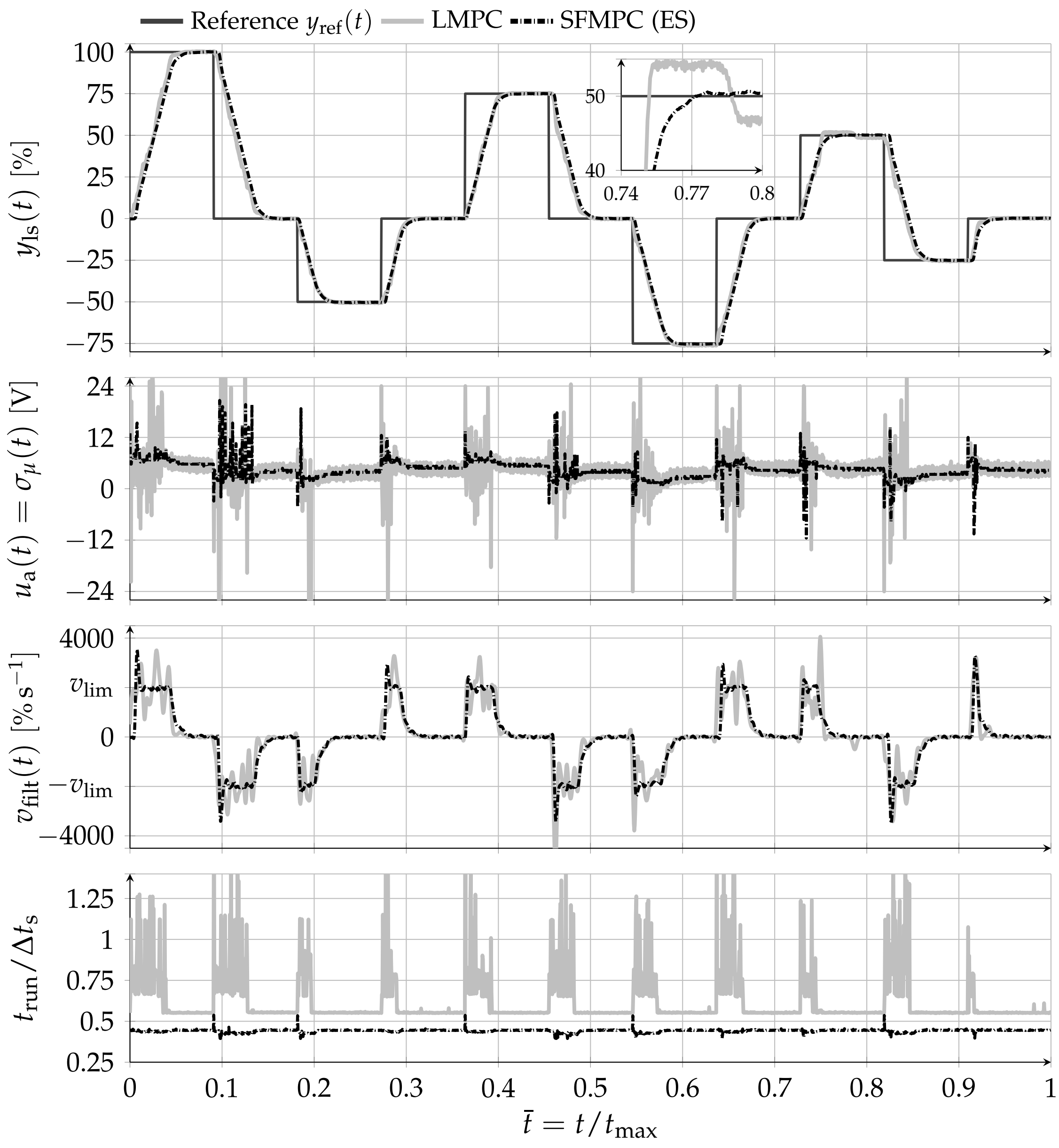


Figure 6.5.: Valve closed-loop real-time control: Constant reference tracking with an **active** stroke **velocity limit**. First: Stroke time performance. Second: Solenoid input voltage. Third: Estimated stroke velocity. Bottom: Normalized execution time.

rests upon the non-smooth penalty function introduced in Section 5.3. Hence, if all input candidates violate the velocity constraint, SFMPC simply chooses the input candidate that reaches the lowest value for the cost function in (5.3.2). The bottom plot in Figure 6.5 underlines the almost deterministic execution time behavior of SFMPC even when constraint violations occur. The limit value v_{lim} can be added as an additional intuitive controller parameter, suitable, for example, to decouple the small signal from the large signal range performance. Hence, if v_{lim} is active, q_2 can be decreased to speed up closed-loop behavior in the small signal range. However, strict adherence to the speed limit would require a higher model accuracy and a robust design of SFMPC, which is beyond the scope of this dissertation.

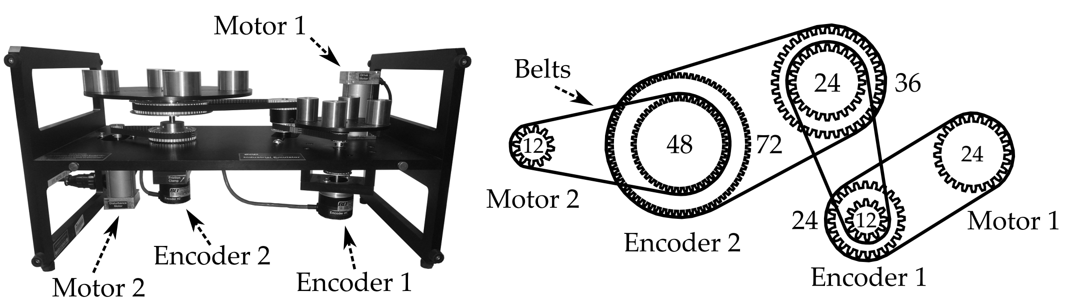


Figure 6.6.: ECP Industrial Plant Emulator Model 220. Left: Picture of the experimental system. Right: Schematic top-view with information on the individual gear ratios.

6.2. Industrial Plant Emulator

The Industrial Plant Emulator Model 220 [ECPa] is particularly suitable for investigating and evaluating classical and novel control concepts for electric drive systems under real operating conditions such as nonlinear friction. The left side of Figure 6.6 shows a picture of the experimental test bench. The schematic top-view on the right side of Figure 6.6 illustrates that the system has five parallel axes of rotation. The inner axes are each equipped with two idler pulleys of different diameters. Adjacent axles are connected to each other via transmission belts. Two brushless DC motors each drive a load plate, which in turn are connected to each other by an idler shaft and transmission belts. Two incremental encoders supply pulses as soon as the load plates rotate. An external digital signal processor (DSP) estimates the angular positions and velocities of the axes on the basis of the temporal resolution of the measured ticks.

6.2.1. Experimental Setup

In this chapter, the Industrial Plant Emulator is configured to be a single-input system. While motor 1 serves as the active actuator, motor 2 is either a passive element, which increases the rotational inertia, or a source of disturbance. Figure 6.7 shows the experimental setup for this chapter. The square-wave output signals of the three Hall sensors $i_{\mathrm{h},1}(t), i_{\mathrm{h},2}(t), i_{\mathrm{h},3}(t)$ indicate the position of the rotor and control the commutation. The internal electronics measure two phase currents $i_{\mathrm{p},1}(t)$ and $i_{\mathrm{p},2}(t)$, while the third phase current results from the three-phase star connection as the negative sum of the other two currents. Two internal proportional-integral controllers process a reference current signal $i_{\mathrm{ref},1} : \mathbb{R}_0^+ \mapsto [-1, 1]$ A and drive the rotor by applying three different time-varying phase voltages $P_1(t), P_2(t)$, and $P_3(t)$ to the four pole stator windings. For further information on the internal structure of the brushless DC motors, refer, for example, to [ECPb]. The control task is the angular positioning of the load plate, below which the second encoder is located. Note that the system transmits the torques generated by the individual motors $M_1(t)$ and $M_2(t)$ with a gear ratio of four to this axis. As soon as the drive torque exceeds the moment of inertia and all friction torques, the load plates start to rotate. The encoder converts the angular velocity $v_\mathrm{a}(t)$ into a square-wave signal and accumulates the measured ticks. Finally, the DSP transforms the encoder signal $E_{\mathrm{dsp}}(t)$ into a measurable angular position signal $y_{\mathrm{dsp}} : \mathbb{R}_0^+ \mapsto \mathbb{R}$

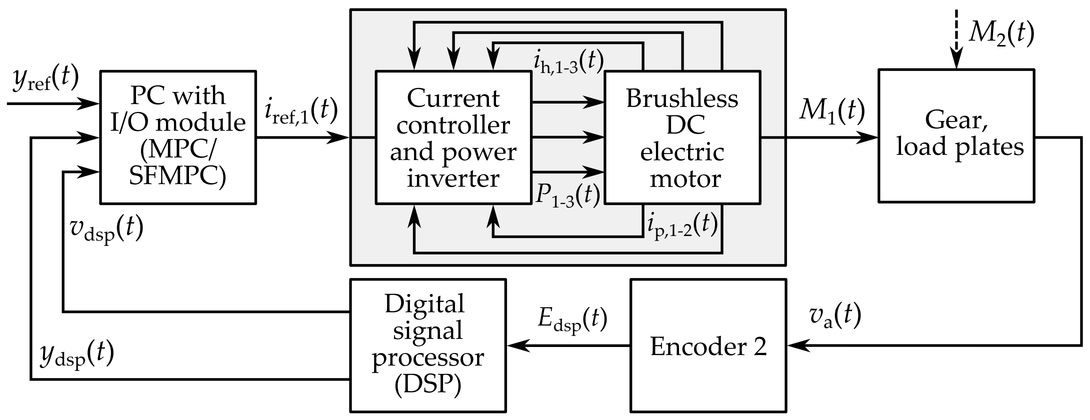


Figure 6.7.: Experimental setup for evaluating model predictive servo motor control. The second motor can be used optionally as a source of disturbance.

and a measurable angular velocity signal $v_{\mathrm{dsp}} : \mathbb{R}_0^+ \mapsto \mathbb{R}$. For the identification procedure and the first experiment, the input signal of the second motor is defined by $i_{\mathrm{ref},2}(t) := 0\,\mathrm{A}$ such that $M_2(t) = 0\,\mathrm{Nm}$ follows for all $t \in \mathbb{R}_0^+$. Again, let $\sigma : \mathbb{R}_0^+ \mapsto \mathbb{R}$ and $y : \mathbb{R}_0^+ \mapsto \mathbb{R}$ denote the input (current) and output (angular position) signals, respectively. The model structure is based on the following second-order differential equation derived from the mechanical equilibrium:

$$I_{\mathrm{ecp}}\,\ddot{y}(t) + C_{\mathrm{ecp}} \tanh\big(b\,\dot{y}(t)\big) + D_{\mathrm{ecp}}\,\dot{y}(t) = 4\,K_{\mathrm{ecp}}\,\sigma(t). \tag{6.2.1}$$

Here, I_{ecp} is the total load inertia about the axis of the second encoder, C_{ecp} is the Coulomb friction torque, D_{ecp} is a damping constant, and K_{ecp} is a motor constant. Since the dry friction representation is not continuous at the origin, the hyperbolic tangent approximates the transition from the negative to the positive angular velocity to enable smooth optimization. The larger the parameter $b \in \mathbb{R}^+$, the larger the gradient at the origin. Note that the differential equation (6.2.1) does not consider static friction, as this would result in a non-smooth function. In this case, conventional MPC could not serve as the reference approach. The physically motivated differential equation (6.2.1) is transformed into the following mathematical state-space representation:

$$\begin{pmatrix} \dot{y}(t) \\ \ddot{y}(t) \end{pmatrix} = \begin{pmatrix} \dot{y}(t) \\ -p_1 \tanh\big(p_2\,\dot{y}(t)\big) - p_3\,\dot{y}(t) + p_4\,\sigma(t) \end{pmatrix}. \tag{6.2.2}$$

The state vector is denoted by $\underline{\chi}(t) = \big(y(t), v(t) = \dot{y}(t)\big)^{\mathsf{T}}$, where $v(t)$ represents the (angular) velocity. The output of the state space model is defined by $y(t) = \underline{C}\,\underline{\chi}(t) = (1, 0)\,\underline{\chi}(t)$. The identification process of the parameters $p_1, p_2, p_3, p_4 \in \mathbb{R}^+$ relies on an ARBS as the input stimulus and global optimization. Figure B.4 compares the estimated position and velocity signals $y_{\mathrm{dsp}}(t)$ and $v_{\mathrm{dsp}}(t)$, respectively, with the simulated signals $y(t)$ and $v(t)$. The model identification process results in a fitness value of $NRMSE = 7.45\,\%$ for the position signal and a fitness value of $NRMSE = 7.49\,\%$ for the velocity signal. The subsequent validation of the model results in a fitness value of $NRMSE = 13.42\,\%$ for the position signal and a fitness value of $NRMSE = 10.55\,\%$ for the velocity signal (see App. B.4).

6.2.2. Controller Design

The controller design in this section follows the same motivation as the controller design in Section 6.1.3. In the following, the sampled-data formulation introduced in Section 3.4 includes the continuous-time state space representation (6.2.2) into the discrete-time formulation of the predictive controller. Here, the input signal $\sigma(t)$ is assumed to be piecewise constant on a fixed time grid with a sampling time of $\Delta t_\mathrm{s} = 0.01\,\mathrm{s}$ and the IVPs are solved approximately by the explicit Runge-Kutta method of the fourth order. The integral behavior of the real system ($u_\mathrm{f} = 0$) and the state space representation (6.2.2) reduce positioning offset errors such that an offset compensation approach is not required. The quadratic cost functions are given by $\ell\big(\underline{\check{x}}(k), \underline{u}(k)\big) = \|\underline{\check{x}}(k)\|^2_{\underline{Q}} + \|\underline{u}(k)\|^2_{\underline{R}}$ and $F\big(\underline{\check{x}}(N)\big) = \|\underline{\check{x}}(N)\|^2_{\underline{P}}$ with $\underline{\check{x}}(k) := \underline{x}(k) - \underline{x}_\mathrm{f}$, $\underline{Q} = \mathrm{diag}(q_1, q_2)$, $\underline{R} = r_1$, and $q_1, q_2, r_1 \in \mathbb{R}^+$. The following model predictive controllers linearize the corresponding sampled data system at the steady state $(\underline{x}_\mathrm{f}, u_\mathrm{f})$ with $\underline{x}_\mathrm{f} = \big(y_\mathrm{ref}(t_n), 0, 0\big)^\mathsf{T}$ and $u_\mathrm{f} = 0$ and resolve the Riccati equation (3.3.18) as soon as the reference position $\underline{x}_\mathrm{f}$ changes its value. Constraint sets are formulated as $\mathbb{X} = X$ or $\mathbb{X} = \{\underline{x} \in X \,|\, |x_2| \leq 8\}$, $\mathbb{X}_\mathrm{f} = \mathbb{X}$, and $\mathbb{U} = \{u \in U \,|\, |u| \leq 1\}$. For MPC, recursive elimination transforms OCP (3.1.6) into NLP (3.2.8), which then is subject to smooth optimization. The solution to this NLP is obtained numerically using the SQP method described in Section 5.2. Here, the numerically robust framework *mpcActiveSetSolver*$(\cdot)$ from MATLAB solves the underlying QPs. With a horizon of $N = 25$, the SQP algorithm always reaches its termination condition in the specified time of $\Delta t_\mathrm{s} = 0.01\,\mathrm{s}$ ($f_\mathrm{s} = 100\,\mathrm{Hz}$). Again, the MPC implementation also considers suboptimal or even infeasible solutions for closed-loop control. SFMPC rests upon the softened OCP formulation (5.3.4) with $\mathbb{P}(\underline{x}_0, n) := \mathbb{A}(n)$ and also operates at a closed-loop sampling time of $\Delta t_\mathrm{s} = 0.01\,\mathrm{s}$ ($f_\mathrm{s} = 100\,\mathrm{Hz}$). SFMPC further implements the exponential discretization in (4.5.3) with a cardinality of $|\mathbb{A}(\cdot)| = 31$ and predicts the system evolution for a horizon length of $N = 25$. Since the single degree of freedom in control tends to result in a damped closed-loop behavior, manual tuning for SFMPC results in $q_1 = 1, q_2 = r_1 = 0.01$, while the parameter tuning for MPC results in $q_1 = 1, q_2 = r_1 = 0.05$. The controller parameterization follows the motivation that the closed-loop behavior of MPC and SFMPC should be fairly similar. Both approaches are subject to the computation time compensation described in Section 6.1.2 with $\underline{\mu}\big(\underline{x}_\mu(0)\big) := \underline{0}_m$ and $\underline{\hat{x}}_\mu(n) := \big(y_\mathrm{dsp}(t_n), v_\mathrm{dsp}(t_n)\big)^\mathsf{T}$.

6.2.3. Performance Evaluation

This section again deals with the tracking performance of constant references with MPC and SFMPC under real-time constraints. The signals in Figure 6.8 and Figure 6.9 are connected continuously though they are measured and evaluated on the fixed time grid $t_{n+1} = t_n + 0.01\,\mathrm{s}, n \in \mathbb{N}_0, t_0 = 0\,\mathrm{s}$. In the first plot of Figure 6.8, both MPC and SFMPC result in smooth position signals without no significant overshoots. Note that the slightly different closed-loop control performances between MPC and SFMPC results from manual controller tuning. The second plot in Figure 6.8 displays that both controller operate the brushless DC motor at its control limits every time the reference

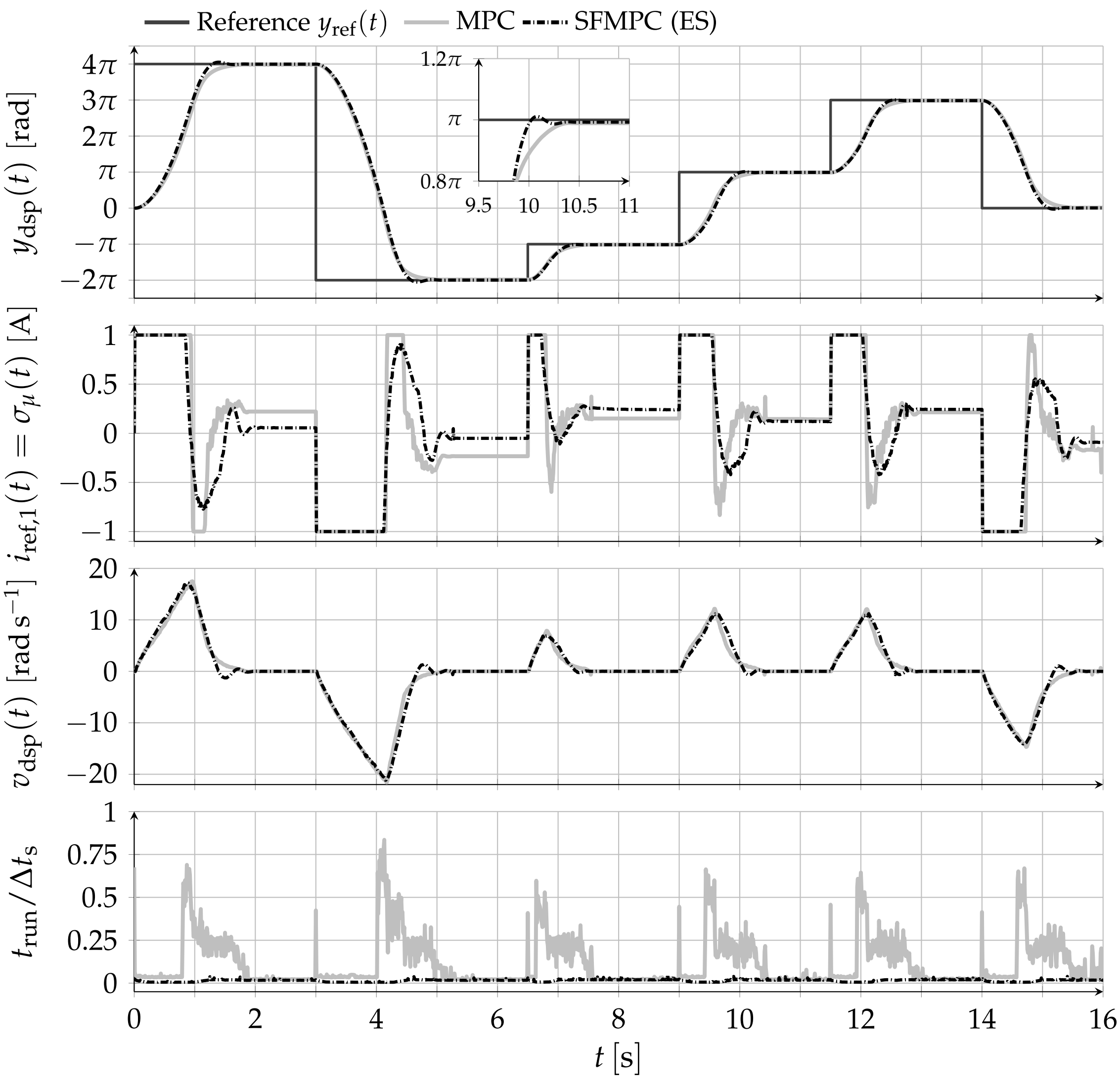


Figure 6.8.: Closed-loop real-time control of the Industrial Plant Emulator: Constant reference tracking. Top: Position time performance. Second: Control input (motor 1). Third: Estimated rotational velocity (DSP). Bottom: Normalized task execution time.

$\underline{x}_{\mathrm{f}}$ switches. The first close-up magnifies the transition to the reference $\underline{x}_{\mathrm{f}} = (\pi, 0, 0)^{\mathsf{T}}$. Because of the integral behavior of the Industrial Plant Emulator, only a small position offset occurs although the model deviation is not negligible (see Fig. B.4). The second plot in Figure 6.8 indicates the existing model mismatch. As soon as the load plate remains at a constant angular position for some time steps, for example between $t = 8\,\mathrm{s}$ and $t = 9\,\mathrm{s}$, the angular velocity $v_{\mathrm{dsp}}(t)$ decreases to zero as shown in the third subplot in Figure 6.8. However, the control input $\sigma_\mu(t) = i_{\mathrm{ref},1}(t)$ in the second subplot does not adopt a value of zero although the reference control vector is $\underline{u}_{\mathrm{f}} = 0$. This offset from $i_{\mathrm{ref},1}(t) = 0\,\mathrm{A}$ is generated by the first element of the predicted control sequence that the internal model requires for getting closer to the reference $\underline{x}_{\mathrm{f}}$ on the prediction horizon. Because of the model mismatch, the real system never reaches the reference $\underline{x}_{\mathrm{f}}$ exactly such that $\underline{x}_0 \neq \underline{x}_{\mathrm{f}}$ applies even when the load plates stop rotating. However, since the internal model does not consider static friction, the first element of

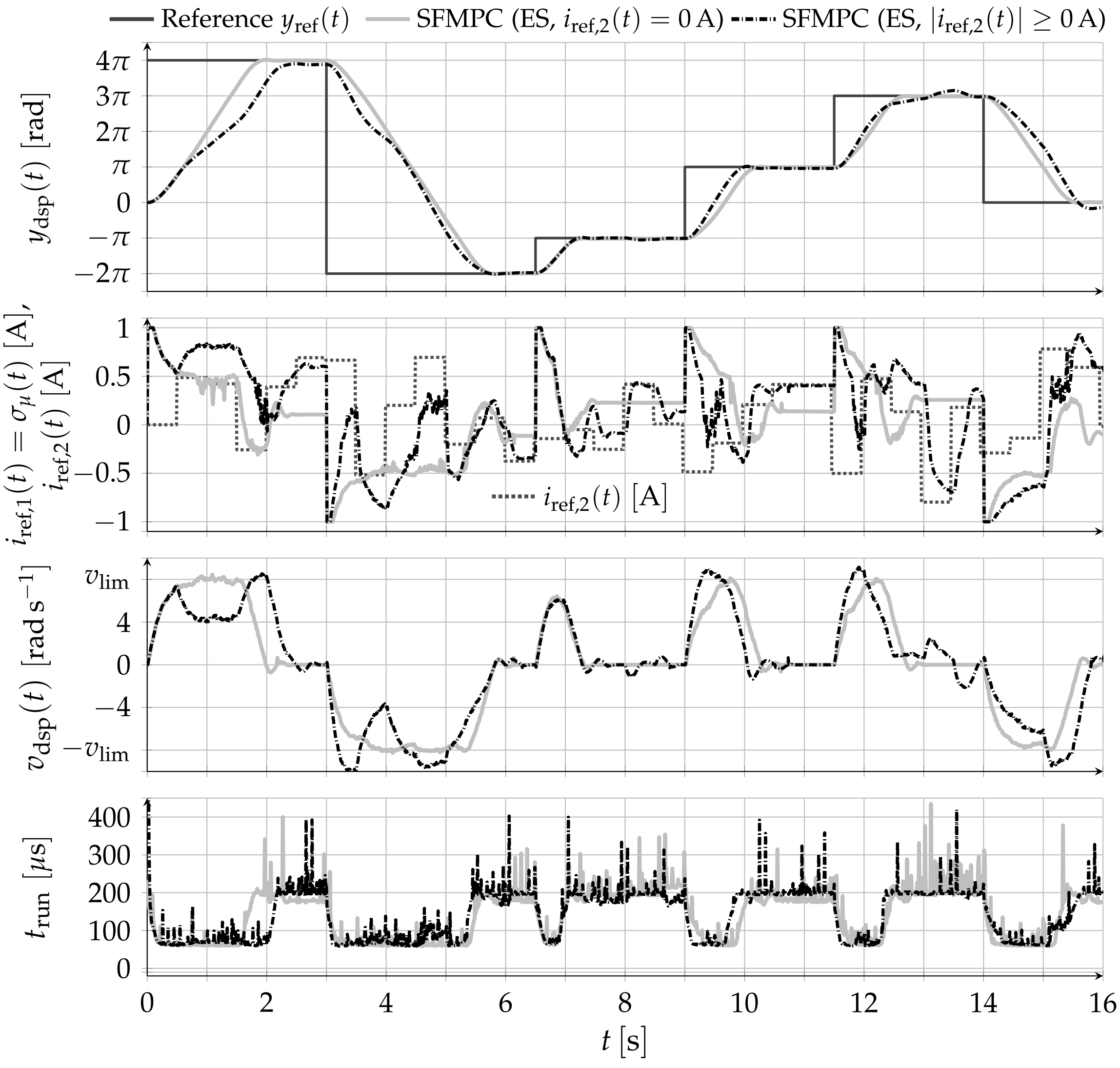


Figure 6.9.: Closed-loop real-time control of the Industrial Plant Emulator: Constant reference tracking with active velocity limit $v_{\mathrm{lim}} = 8\,\mathrm{rad\,s}^{-1}$ and with strong inference torque from motor 2. Top: Position time performance. Second: Control input at motor 1 and disturbance input at motor 2. Third: Estimated rotational velocity (DSP). Bottom: Task execution time.

the predicted control sequence is not high enough to generate a torque that overcomes static friction. The bottom graph again reveals that SFMPC has a significantly lower computational effort compared to MPC, although the closed-loop control performance is fairly similar. In case of MPC, computation times fluctuate and are close to the maximum value every time the reference position $y_{\mathrm{ref}}(t)$ changes. This is because the SQP algorithm cannot reuse the previous solution for warm-starting optimization at the current closed-loop time instant (see Rem. 3.3.4). SFMPC, in contrast, showcases a nearly constant computational effort.

The experiment in Figure 6.9 challenges SFMPC. First, the real-time control is only subject to a velocity limit of $v_{\mathrm{lim}} = 8\,\mathrm{rads}^{-1}$ (gray graphs). The closed-loop position performance in the first plot of Figure 6.9 is similar to the closed-loop position performance in the first plot of Figure 6.8. However, here, SFMPC adheres closely to the velocity constraint. For example, between $t = 1\,\mathrm{s}$ and $t = 1.5\,\mathrm{s}$, the third plot in Figure 6.9 indicates only small constraint violations. These unexpected violations occur because of model mismatch and measurement errors. However, the execution time in the bottom plot is not affected by these constraint violations. The alternating computation time level between $t_{\mathrm{run}} = 70\,\mu\mathrm{s}$ and $t_{\mathrm{run}} = 200\,\mu\mathrm{s}$ is caused by the function $\tanh(\cdot)$. The internal numerical representation of this function requires a higher computational effort when the internal model velocity $v(t_n)$ is in some neighborhood of zero. In the second use case, the second motor generates a strong disturbance torque. The second plot shows the input profile $i_{\mathrm{ref},2}(t)$ for the internal electronics of the second motor (gray dotted profile), which is then translated into a disturbance torque $M_2(t)$. Here, the softened OCP formulation (5.3.4) proves to be extremely beneficial for practical applications. Despite the large disturbance torque, SFMPC tries to transfer the system into the feasible state space while still tracking the constant references (black dash-dotted graphs). At the same time, the controller only violates the constraints slightly and exhibits an almost constant computational effort.

6.3. Discussion

This chapter provides experimental evidence that SFMPC is a suitable control concept for two types of electric motors, at least for position control. Since SFMPC already includes a high level of system knowledge from systematic model identification, the subsequent controller design is very intuitive. During real-time operation, a manual tuning of controller parameters already results in a high closed-loop control performance. The almost deterministic and low computational effort of SFMPC with finite control sets enables smooth closed-loop control of fast mechatronic systems in the upper Kilohertz sampling range. Because of the large number of degrees of freedom in control, conventional MPC outperforms SFMPC in the nominal case, during the numerical analysis in the previous Chapter 5. However, during the real-time experiments, SFMPC and MPC perform fairly similar without state constraints, though smooth optimization varies strongly in the number of required optimization iterations to satisfy convergence criteria. Since the measurement, observer, and especially model errors are commonly not negligible in real-world experiments, MPC must be extended by a soft constraint formulation or/and by robustification strategies. Robustification, however, complicates controller design and imposes a computational overhead. Implementing a non-smooth exact penalty function proves to be a major advantage of SFMPC for real-world applications. As long as an admissible solution exists, SFMPC strictly adheres to the state constraints. If, due to model errors or external disturbances, an admissible solution suddenly no longer exists, SFMPC simply selects a control candidate that transfers the system back into, or at least closer to, the feasible state space. In summary, the controller complexity of SFMPC for a real-world experiment is equal to the controller complexity during ideal simulation. Basic SFMPC addresses, in particular, industrial applications where the stability analysis is limited to experimental and sufficient testing of real-time control scenarios.

7 Suboptimal Input Move-Blocked Model Predictive Control

Before investigating closed-loop stability properties of SFMPC, this chapter introduces a novel formulation for general MBMPC that accounts for recursive feasibility and asymptotic stability. The key contribution of this chapter is the integration of input move-blocking into the suboptimal MPC framework presented in [Raw+20, Sec. 2.7] and [All+17]. Assume that there exists a solution to OCP (3.1.6) and that the chosen optimizer can find the minimum in finite time. By subsequently introducing input move-blocking constraints, the solution space might not contain the original minimizer anymore. In this case, the optimizer cannot find the optimal solution to OCP (3.1.6) such that input move-blocking can be classified as a source of suboptimality. Chen et al. [Che+20] also notice this relationship between the suboptimal formulation and input move-blocking, however, the authors do not further elaborate on this connection. The major part of this chapter has been published in [Mak+22].

7.1. Generating Stabilizing Warm-Starts

Before introducing the extension to input move-blocking, this section summarizes the main ingredients of suboptimal MPC with an adapted nomenclature. First, let $\underline{\tilde{\mathbf{u}}}(\utilde{x}_0) = \big(\underline{\tilde{u}}(0, \utilde{x}_0), \underline{\tilde{u}}(1, \utilde{x}_0), ..., \underline{\tilde{u}}(N-1, \utilde{x}_0)\big) \in \mathcal{U}_N(\utilde{x}_0)$ be some admissible warm-start control sequence. According to [Pan+11, Eq. (5b)] and [All+17, Eq. (12)], the set of all control sequences that result in lower or equal costs than the warm-start is defined by:

$$\mathcal{U}_N^{\dagger}\big(\utilde{x}_0, \underline{\tilde{\mathbf{u}}}(\utilde{x}_0)\big) := \big\{\underline{\mathbf{u}} \in \mathcal{U}_N(\utilde{x}_0) \,|\, J_N(\utilde{x}_0, \underline{\mathbf{u}}) \leq J_N\big(\utilde{x}_0, \underline{\tilde{\mathbf{u}}}(\utilde{x}_0)\big)\big\}. \tag{7.1.1}$$

Suboptimal MPC triggers the optimizer to solve the following OCP:

$$\min_{\underline{\mathbf{u}} \in \mathcal{U}_N^{\dagger}(\utilde{x}_0, \underline{\tilde{\mathbf{u}}}(\utilde{x}_0))} J_N(\utilde{x}_0, \underline{\mathbf{u}}). \tag{7.1.2}$$

The globally optimal solution to OCP (7.1.2) is denoted by $\underline{\mathbf{u}}^*(\utilde{x}_0) \in \mathcal{U}_N^{\dagger}\big(\utilde{x}_0, \underline{\tilde{\mathbf{u}}}(\utilde{x}_0)\big)$. However, the optimizer can also provide an admissible but suboptimal solution that is denoted by $\underline{\mathbf{u}}^{\dagger}(\utilde{x}_0) = \big(\underline{u}^{\dagger}(0, \utilde{x}_0), \underline{u}^{\dagger}(1, \utilde{x}_0), ..., \underline{u}^{\dagger}(N-1, \utilde{x}_0)\big) \in \mathcal{U}_N^{\dagger}\big(\utilde{x}_0, \underline{\tilde{\mathbf{u}}}(\utilde{x}_0)\big)$ with the relation $J_N\big(\underline{x}_0, \underline{\mathbf{u}}^*(\underline{x}_0)\big) \leq J_N\big(\underline{x}_0, \underline{\mathbf{u}}^\dagger(\underline{x}_0)\big)$. Similar to conventional MPC, the first element of the suboptimal control sequence provided by the optimizer is applied for closed-loop control:

$$\underline{x}_0^+ := \underline{x}_\mu(n+1) = \underline{f}\Big(\underline{x}_\mu(n), \underline{u}^\dagger\big(0, \underline{x}_\mu(n)\big)\Big). \tag{7.1.3}$$

Assume that the optimizer cannot find the globally optimal solution, or more generally, assume that the optimizer cannot provide an improvement on the warm start solution $\tilde{\underline{\mathbf{u}}}(\underline{x}_0)$ in terms of costs. Obviously, in this case, the warm-start solution serves as the fall-back level. The authors of [Sco+99; Pan+11; All+17; Raw+20] develop a theoretical framework for designing stabilizing warm-starts that ensure recursive feasibility and asymptotic stability of the origin (exponential stability in [Pan+11]) even without applying additional optimization iterations. Note that the definition in (7.1.1) implies that $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \mathcal{U}_N^\dagger\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)$. Therefore, there always exists a solution to OCP (7.1.2). In [Pan+11, Eq. (5c)] and [All+17, Eq. (11)], the set of all admissible warm-start control sequences is defined by:

$$\tilde{\mathcal{U}}_N(\underline{x}_0) := \{\underline{\mathbf{u}} \in \mathcal{U}_N(\underline{x}_0) \mid J_N(\underline{x}_0, \underline{\mathbf{u}}) \leq F(\underline{x}_0), \text{ if } \underline{x}_0 \in \mathbb{X}_\mathrm{f}\}. \tag{7.1.4}$$

Here, the terminal cost function $F(\cdot)$ is chosen to be a CLF in $\mathbb{X}_\mathrm{f}$ (see Asm. 3.3.2). The introduction of the admissible warm-start set $\tilde{\mathcal{U}}_N(\underline{x}_0)$ ensures the property that when $\|\underline{x}_0\| \to \underline{0}$ it also follows that $\|\underline{\mathbf{u}}^\dagger(\underline{x}_0)\| \to \underline{0}$ [Pan+11; All+17; Raw+20], which is important for asymptotic stabilization as summarized below. In addition to the operator $\underline{\Omega}_\mathrm{sta} : \mathbb{X} \times \mathbb{U}^N \mapsto \mathbb{U}^N$ (shift-truncate-append operator), another operator is required in order to formulate the manual generation of warm-starts. Let $\underline{\Omega}_\kappa : \mathbb{X}_\mathrm{f} \mapsto \mathbb{U}^N$ be the operator that applies the local control law for N times such that $\underline{\Omega}_\kappa(\underline{x}_0) = \big(\underline{\kappa}(\underline{x}_0), \underline{\kappa}\big(\underline{f}\big(\underline{x}_0, \underline{\kappa}(\underline{x}_0)\big)\big), \ldots\big)$ follows. Now, suboptimal MPC generates warm-start solutions according to the following scheme and based on the prediction $\underline{x}_0^+ = \underline{f}\big(\underline{x}_0, \underline{u}^\dagger(0, \underline{x}_0)\big)$ [Pan+11, Eq. (5)], [All+17, Eq. (13)]:

$$\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) = \underline{\Omega}\big(\underline{x}_0, \underline{\mathbf{u}}^\dagger(\underline{x}_0)\big) := \begin{cases} \underline{\Omega}_\kappa(\underline{x}_0^+) & \text{if } \underline{x}_0^+ \in \mathbb{X}_\mathrm{f} \text{ and } J_N\big(\underline{x}_0^+, \underline{\Omega}_\kappa(\underline{x}_0^+)\big) \leq \\ & \leq J_N\big(\underline{x}_0^+, \underline{\Omega}_\mathrm{sta}\big(\underline{x}_0, \underline{\mathbf{u}}^\dagger(\underline{x}_0)\big)\big), \\ \underline{\Omega}_\mathrm{sta}\big(\underline{x}_0, \underline{\mathbf{u}}^\dagger(\underline{x}_0)\big) & \text{otherwise.} \end{cases} \tag{7.1.5}$$

The first case in (7.1.5) ensures that $\underline{\Omega}_\kappa(\underline{x}_0)$ is always a member of the admissible warm-start set $\tilde{\mathcal{U}}_N(\underline{x}_0)$ in (7.1.4). Notice that by Assumption 3.3.2, the following inequality applies for all $\underline{x}_0 \in \mathbb{X}_\mathrm{f}$ [Pan+11, Prop. 9], [All+17, Prop. 8]:

$$J_N\big(\underline{x}_0, \underline{\Omega}_\kappa(\underline{x}_0)\big) = \sum_{k=0}^{N-1} \ell\Big(\underline{x}(k), \underline{\kappa}\big(\underline{x}(k)\big)\Big) + F\big(\underline{x}(N)\big) \leq F(\underline{x}_0),\ \underline{x}(k) = \underline{\varphi}\big(k, \underline{x}_0, \underline{\Omega}_\kappa(\underline{x}_0)\big). \tag{7.1.6}$$

Consequently, it follows that $\underline{\Omega}_\kappa(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$. The authors of [Pan+11; All+17; Raw+20] introduce the extended state $\underline{\vartheta} := \big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big) \in \mathbb{R}^{p+mN}$ and the following difference inclusion, which includes the closed-loop system (7.1.3) [All+17, Eq. (14)]:

$$\begin{aligned} \underline{\vartheta}^+ \in \underline{H}(\underline{\vartheta}) := \big\{\big(\underline{x}_0^+, \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)\big) \mid \underline{x}_0^+ = \underline{f}\big(\underline{x}_0, \underline{u}^\dagger(0, \underline{x}_0)\big), \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) = \underline{\Omega}\big(\underline{x}_0, \underline{\mathbf{u}}^\dagger(\underline{x}_0)\big), \\ \underline{\mathbf{u}}^\dagger(\underline{x}_0) \in \mathcal{U}_N^\dagger\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)\big\}. \end{aligned} \tag{7.1.7}$$

In contrast to the optimal MPC formulation in Section 3.3, it is not clear which solution $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0)$ the optimizer provides in the suboptimal case. Hence, the evolution of the extended state $\underline{\vartheta}^{+}$, and thus of the closed-loop system, is described by the set-valued map $\underline{H}(\cdot)$. The feasible domain of this difference inclusion is defined as follows [Pan+11; All+17]:

$$\mathcal{Z}_N := \big\{ \big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big) \mid \underline{x}_0 \in \mathcal{X}_N \text{ and } \tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0) \big\}. \tag{7.1.8}$$

Since the warm-start generation according to (7.1.5) makes use of the control invariance property of the terminal set $\mathbb{X}_{\mathrm{f}}$ (see Asm. 3.3.2), applying the warm-starts with $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0) := \tilde{\underline{\mathbf{u}}}(\underline{x}_0)$ already renders the set $\mathcal{Z}_N$ positive invariant for the closed-loop system (7.1.7) (e.g., [All+17, Proof of Thm. 14]). Pannocchia et al. [Pan+11] and Allan et al. [All+17] (see also [Raw+20]) show that $J_N(\cdot)$ is a Lyapunov function in the set $\mathcal{Z}_N$ for the closed-loop system (7.1.7). The following Lyapunov inequalities hold with $\alpha_1(\cdot), \alpha_2(\cdot), \alpha_3(\cdot) \in \mathcal{K}_\infty$ and for all $\underline{\vartheta} \in \mathcal{Z}_N$ [All+17, Thm. 14]:

$$\alpha_1(\|\underline{\vartheta}\|) \leq J_N(\underline{\vartheta}) \leq \alpha_2(\|\underline{\vartheta}\|), \quad \sup_{\underline{\vartheta}^{+} \in \underline{H}(\underline{\vartheta})} J_N(\underline{\vartheta}^{+}) \leq J_N(\underline{\vartheta}) - \alpha_3(\|\underline{\vartheta}\|). \tag{7.1.9}$$

The existence of the lower bound $\alpha_1(\cdot)$ follows from Assumption 3.1.1, Assumptions 3.1.2–3.3.1, and the algebraic transformations in [All+17, Prop. 22]. Similar to the optimal formulation in Section 3.3, the upper bound $\alpha_2(\cdot)$ results from Assumptions 3.1.1-3.3.1 and again from [RR17b, Prop. 14]. The second inequality in (7.1.9) indicates the following relation. As long as the worst-case evolution of the extended state satisfies the cost descent property, it automatically applies to all other possible closed-loop evolutions. The existence of the local CLF in Assumption 3.3.2 ensures that the following cost descent holds for all $\underline{\vartheta} \in \mathcal{Z}_N$ [All+17, Proof Thm. 14]:

$$\begin{aligned} J_N(\underline{\vartheta}^{+}) = J_N\big(\underline{x}_0^{+}, \tilde{\underline{\mathbf{u}}}(\underline{x}_0^{+})\big) &\leq J_N\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big) - \ell\big(\underline{x}_0, \underline{u}^{\dagger}(0, \underline{x}_0)\big) \leq \\ &\leq J_N\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big) - \alpha_\ell(\|(\underline{x}_0, \underline{u}^{\dagger}(0, \underline{x}_0))\|). \end{aligned} \tag{7.1.10}$$

Since $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0) \in \mathcal{U}_N^{\dagger}\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)$, it follows that $J_N\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big) \leq J_N\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big) = J_N(\underline{\vartheta})$ [All+17, Proof Thm. 14]. Finally Allan et al. show that there exists a function $\alpha_3(\cdot) \in \mathcal{K}_\infty$ such that $\alpha_3(\|\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)\|) \leq \alpha_\ell(\|(\underline{x}_0, \underline{u}^{\dagger}(0, \underline{x}_0))\|)$ [All+17, Prop. 10, Proof Thm. 14]. It is important to note that the latter step explicitly requires that $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$. Since $J_N(\cdot)$ is a valid Lyapunov function in the positive invariant set $\mathcal{Z}_N$, $\underline{H}(\underline{0}_{p+mN}) = \{(\underline{0}_{p+mN})\}$, asymptotic stability of the origin follows from Proposition 13 in [All+17].

7.2. Integrating Input Move-Blocking Through Buffering

Input move-blocking groups consecutive control vectors on the prediction horizon into blocks that share the same value along each input dimension. This input parameterization can be formalized by introducing an admissible blocking matrix $\check{\underline{B}} \in \mathbb{R}^{N \times M}$ and the Kronecker product $\otimes$ [Cag+07]. Here, $M \leq N$ represents the number of degrees of freedom in control and the matrix $\check{\underline{B}}$ only contains zeros and ones [TJ02]. Recall that in Section 4.1, a special case of input move-blocking was already implemented by introducing the operator $\underline{\Theta}^{\mathrm{s}}_N(\underline{u}_{\mathrm{s}}) := (\check{\underline{B}} \otimes \underline{I}_m)\underline{u}_{\mathrm{s}}$ with $\check{\underline{B}} := \underline{1}_N$ and $M = 1$. For example, a uniform input move-blocking pattern with $M = 2$ and a horizon length of $N = 4$ allows a single switching during prediction and adopts the following structure:

$$\check{\underline{B}} := \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}^{\mathsf{T}}. \tag{7.2.1}$$

According to the Definition 3 in [Cag+07], a blocking matrix is admissible if it contains exactly one element equal to one in each row. Each new block, which follows the previous one in terms of prediction time, is shifted by one column. The Kronecker product expands the chosen input parameterization to all input dimensions [Cag+07]. For general input move-blocking with arbitrary blocking-patterns the input parameterization is defined by:

$$\underline{\mathbf{u}} = \underline{\Theta}_N(\check{\underline{\mathbf{u}}}) := \left(\check{\underline{B}} \otimes \underline{I}_m\right) \check{\underline{\mathbf{u}}}. \tag{7.2.2}$$

Here, the vector field $\underline{\Theta}_N(\cdot)$ maps $U^M \mapsto U^N$ such that the reduced-order sequence is defined by $\check{\underline{\mathbf{u}}} := \left(\check{\underline{u}}(0), \check{\underline{u}}(1), ..., \check{\underline{u}}(M-1)\right) \in U^M$. Notice that $\underline{\Theta}^{\mathrm{s}}_N(\underline{u}_{\mathrm{s}}) = \underline{\Theta}_N(\underline{u}_{\mathrm{s}})$ holds. The setting $M = N$ disables move-blocking such that $\underline{\mathbf{u}} = \underline{\Theta}_N(\check{\underline{\mathbf{u}}}) = \check{\underline{\mathbf{u}}}$ applies. The following optimization problem is subject to naive input move-blocking:

$$\min_{\check{\underline{\mathbf{u}}} \in \mathbb{U}^M} J_N\left(\underline{x}_0, \underline{\Theta}_N(\check{\underline{\mathbf{u}}})\right) \quad \text{subject to} \quad \underline{\Theta}_N(\check{\underline{\mathbf{u}}}) \in \mathcal{U}_N(\underline{x}_0). \tag{7.2.3}$$

The solution to this OCP, if it exists, does not ensure recursive feasibility and asymptotic stability of the steady state. Recall that the admissible input set $\mathcal{U}_N(\underline{x}_0)$ does not consider move-blocking in its formulation. However, it is worth noting that the formulation of input move-blocking with $M \leq N$ seamlessly integrates into the previous derivations on suboptimal MPC. With input move-blocking, OCP (7.2.3) requires the following modification:

$$\min_{\check{\underline{\mathbf{u}}} \in \mathbb{U}^M} J_N\left(\underline{x}_0, \underline{\Theta}_N(\check{\underline{\mathbf{u}}})\right) \quad \text{subject to} \quad \underline{\Theta}_N(\check{\underline{\mathbf{u}}}) \in \mathcal{U}^{\dagger}_N\left(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\right). \tag{7.2.4}$$

The optimal solution is denoted by $\check{\underline{\mathbf{u}}}^*(\underline{x}_0) = \left(\check{\underline{u}}^*(0, \underline{x}_0), \check{\underline{u}}^*(1, \underline{x}_0), ..., \check{\underline{u}}^*(M-1, \underline{x}_0)\right)$, whereas $\check{\underline{\mathbf{u}}}^{\dagger}(\underline{x}_0) = \left(\check{\underline{u}}^{\dagger}(0, \underline{x}_0), \check{\underline{u}}^{\dagger}(1, \underline{x}_0), ..., \check{\underline{u}}^{\dagger}(M-1, \underline{x}_0)\right)$ represents a suboptimal solution to OCP (7.2.4). Since OCP (7.1.2) is enhanced subsequently by input move-blocking constraints, there is no guarantee that the resulting OCP (7.2.4) is feasible for all $\underline{x}_0 \in \mathcal{X}_N$. When applying smooth optimization (e.g., Newton-type solvers) with $N = M$ the optimization algorithm usually accepts only those optimization steps that improve the warm-start in terms of costs. With $M < N$ and M constant over all closed-loop time steps n, the optimizer cannot handle the structure of the warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0)$ since the temporal position of the switching points and the number of switching points do not coincide between $\underline{\Theta}_N\left(\check{\underline{\mathbf{u}}}^{\dagger}(\underline{x}_0^+)\right)$ and $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)$ as the closed-loop system evolves. Assume that OCP (7.2.4) is feasible at time instant $n = 0$ with some $M < N$ and the optimizer provides the globally optimal solution. If the first case in equation (7.1.5) applies, the warm-started control sequence $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)$ at the next time step $n+1$ has $M = N$ degrees of freedom in control. If the second case in (7.1.5) applies, the warm-started control sequence $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+)$ has $M+1$ degrees of freedom in control resulting from the application of the shift-truncate-append operator. The following extension addresses the cases in which OCP (7.2.4) turns infeasible:

$$\underline{\mathbf{u}}^\dagger(\underline{x}_0) := \begin{cases} \Theta_N(\check{\underline{\mathbf{u}}}^\dagger(\underline{x}_0)) & \text{if OCP (7.2.4) is feasible,} \\ \tilde{\underline{\mathbf{u}}}(\underline{x}_0) & \text{otherwise.} \end{cases} \tag{7.2.5}$$

If the optimizer finds a better solution than the warm-start though starting from another point in the parameter space than the warm-start, the first case in (7.2.5) applies. If the first case does not apply, the optimization routine needs to fall back to the manually generated warm-start such that the second case in (7.2.5) applies. Therefore, the manually generated fallback solution $\tilde{\underline{\mathbf{u}}}(\underline{x}_0)$ must be temporarily buffered between two consecutive closed-loop steps. Obviously, in the setting of input move-blocking, the term fallback solution is better suited than the term warm-start. Note that the feasible set $\mathcal{X}_N$ that is included in the definition of the extended set $\mathcal{Z}_N$ does not consider input move-blocking in its formulation. To access the results from suboptimal MPC with $M = N$, the following assumption is necessary.

Assumption 7.2.1: *Admissible initial warm-start*. At time instant $n = 0$, there exists an admissible warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ for all $\underline{x}_0 \in \mathcal{X}_N$.

In [All+17, Alg. 9], the authors implicitly assume that there exists an admissible warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ at time instant $n = 0$. Here, in the unblocked case with $M = N$, every suboptimal solution to the original OCP (3.1.6) can be simply used as the initial warm-start. Input move-blocking, in contrast, reduces the feasible set such that there is no guarantee that there exists a suboptimal solution to OCP (3.1.6) that additionally satisfies the blocking constraints. The authors of [BL17] use a similar assumption in the context of sampling based MPC. Here, an "oracle" generates the initial admissible control sequence, which is then improved systematically by sampling-based optimization.

Proposition 7.2.1: *Suboptimal MBMPC*. Suppose Assumptions 3.1.1-3.3.2 and 7.2.1 hold. Assume that the suboptimal control sequences are generated based on the fallback level (7.2.5) and the suboptimal solutions to OCP (7.2.4). Then, the origin is asymptotically stable for the closed-loop system (7.1.7) in the positive invariant set $\mathcal{Z}_N$.

Proof. Assumption 7.2.1 ensures that an initial and admissible warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ exists for all $\underline{x}_0 \in \mathcal{X}_N$, which is not necessarily subject to input move-blocking. If OCP (7.2.4) is feasible for some $\underline{x}_0 \in \mathcal{X}_N$, every suboptimal solution produces lower costs than the warm-start since $\Theta_N(\check{\underline{\mathbf{u}}}^\dagger(\underline{x}_0)) \in \mathcal{U}_N^\dagger(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0))$. However, the fallback level in (7.2.5) includes the warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ at every closed-loop time instant $n \geq 0$. Since $\underline{\mathbf{u}}^\dagger(\underline{x}_0) = \tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \mathcal{U}_N^\dagger(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0))$, $\underline{x}_0^+ \in \mathcal{X}_N$ holds after applying the suboptimal control vector $\underline{u}^\dagger(0, \underline{x}_0)$ for closed-loop control. From (7.1.5) it follows that $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) = \underline{\Omega}(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)) \in \tilde{\mathcal{U}}_N(\underline{x}_0^+)$ such that $\underline{\vartheta}^+ \in \mathcal{Z}_N$ (see [All+17, Proof Thm. 14]). The rest of the Proof (asymptotic stability) follows from [All+17, Prop. 13 and Proof of Thm. 14] and relies on the Assumptions 3.1.1-3.3.2. □

The rigorous realization of Assumption 7.2.1 might require the inclusion of OCP (3.1.6) that is not subject to input move-blocking. This might imply that input move-blocking can only be enabled for $n \geq 1$. However, in practice, the initial state vectors can be simply restricted to the following closed feasible set:

$$\bar{\mathcal{X}}_M := \{\, \underline{x}_0 \in \mathbb{X} \mid \exists \check{\underline{\mathbf{u}}} \text{ such that } \underline{\Theta}_N(\check{\underline{\mathbf{u}}}) \in \mathcal{U}_N(\underline{x}_0) \}. \tag{7.2.6}$$

Here, $\bar{\mathcal{X}}_M \subseteq \mathcal{X}_N$ holds. If $\underline{x}_0 \in \bar{\mathcal{X}}_M \backslash \mathbb{X}_\mathrm{f}$, then $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) = \underline{\Theta}_N(\check{\underline{\mathbf{u}}}) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ represents a possible warm-start at time step $n = 0$. If $\underline{x}_0 \in \mathbb{X}_\mathrm{f}$, then $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) = \underline{\Omega}_\kappa(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ simply applies (see [All+17, Alg. 9]). In both cases, $(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)) \in \mathcal{Z}_N$ holds at time instant $n = 0$.

Remark 7.2.1: *Blocking inside terminal set*. If input move-blocking is active inside the terminal set $\mathbb{X}_\mathrm{f}$, there is no guarantee that $J_N(\underline{x}_0^+, \underline{\Omega}_\mathrm{sta}(\underline{x}_0, \underline{\Theta}_N(\check{\underline{\mathbf{u}}}^\dagger(\underline{x}_0)))) \leq F(\underline{x}_0^+) \leq \alpha_\mathrm{f}(\|\underline{x}_0^+\|)$ applies with $\underline{x}_0^+ \in \mathbb{X}_\mathrm{f}$. Therefore, suboptimal MPC with input move-blocking relies, in particular, on evaluating $\underline{\Omega}_\kappa(\underline{x}_0^+)$ (see (7.1.5)) as an alternative warm-start.

Remark 7.2.2: *Inherent robustness with input move-blocking*. Allan et al. [All+17] derive inherent robustness properties for suboptimal MPC with $\mathbb{X} = X$ (softened state constraints) and $\mathbb{X}_\mathrm{f} := \mathrm{lev}_\pi F := \{\underline{x} \in \mathbb{X} \mid F(\underline{x}) \leq \pi\}$ with some $\pi > 0$. The results on inherent robustness rely on the stabilizing warm-starts $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ and the basic stability Assumptions 3.1.1-3.3.2. Therefore, with the same configuration of sets, suboptimal MBMPC inherits the robustness results from [All+17] as long as the warm-start sequences are available at every closed-loop time instant $n \in \mathbb{N}_0$.

The greater the restriction on the degrees of freedom in control, the more likely it is that there are no better suboptimal solutions than the warm-start. In such cases, the optimizer cannot improve the closed-loop performance because of the restrictive input parameterization. Moreover, the numerical detection of infeasibility of an OCP involves a high computational effort. Usually, an optimization algorithm detects infeasibility by reaching the maximum number of iterations, for example, during line search. The next section presents a reconfiguration of the optimization problem that allows the optimizer to improve the warm-start incrementally.

7.3. Offset Input Move-Blocking

The following redefinition of the input parameterization is adopted from [OW14; SM15] with some scaling parameter $\lambda \in \mathbb{R}_0^+$. Here, the move-blocked control sequence serves as an offset for the manually generated warm-start:

$$\underline{\mathbf{u}} = \underline{\Theta}_N(\check{\underline{\mathbf{u}}}, \tilde{\underline{\mathbf{u}}}(\underline{x}_0), \lambda) := (\check{\underline{B}} \otimes \underline{I}_m)\, \check{\underline{\mathbf{u}}} + \lambda\, \tilde{\underline{\mathbf{u}}}(\underline{x}_0). \tag{7.3.1}$$

By including the scaling parameter λ as an additional optimization parameter similar to [OW14; SM15], the optimizer can now process the manually generated warm-start:

$$\min_{\check{\underline{\mathbf{u}}} \in \mathbb{U}^M,\, \lambda \in \mathbb{R}_0^+} J_N\Big(\underline{x}_0, \underline{\Theta}_N(\check{\underline{\mathbf{u}}}, \tilde{\underline{\mathbf{u}}}(\underline{x}_0), \lambda)\Big) \text{ subject to } \underline{\Theta}_N(\check{\underline{\mathbf{u}}}, \tilde{\underline{\mathbf{u}}}(\underline{x}_0), \lambda) \in \mathcal{U}_N^\dagger(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)). \tag{7.3.2}$$

A suboptimal solution tuple to OCP (7.3.2) is denoted by $(\check{\underline{\mathbf{u}}}^{\dagger}(\underline{x}_0), \lambda^{\dagger})$. The corresponding suboptimal control sequence follows by:

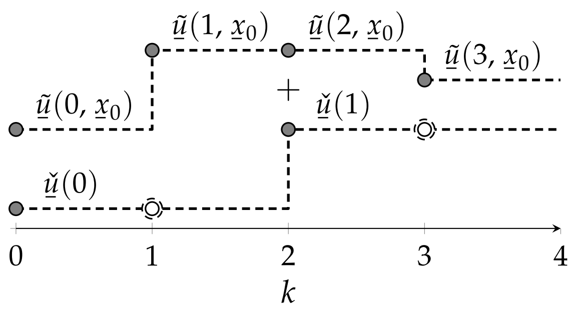
$$\underline{\mathbf{u}}^{\dagger}(\underline{x}_0) := \underline{\Theta}_N\big(\check{\underline{\mathbf{u}}}^{\dagger}(\underline{x}_0), \tilde{\underline{\mathbf{u}}}(\underline{x}_0), \lambda^{\dagger}\big) \tag{7.3.3}$$

Proposition 7.3.1: *Suboptimal offset MBMPC*. Suppose Assumptions 3.1.1-3.3.2 and 7.2.1 hold. Assume that the suboptimal control sequences are generated based on (7.3.3) and the suboptimal solutions to OCP (7.3.2). Then, the origin is asymptotically stable for the closed-loop system (7.1.7) in the positive invariant set $\mathcal{Z}_N$.

Proof. Since $\{0\}^{mM} \subset \mathbb{U}^M$ holds by Assumption 3.1.3, the optimizer can now resort to the warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \tilde{\mathcal{U}}_N(\underline{x}_0)$ autonomously by setting $\lambda^{\dagger} = 1$ and $\check{\underline{\mathbf{u}}}^{\dagger}(\underline{x}_0) = \underline{0}_{mM}$ (see also [SM15, Thm. 1]). Assumption 7.2.1 provides the first admissible warm-start at closed-loop time $n = 0$. Again, since $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0) = \tilde{\underline{\mathbf{u}}}(\underline{x}_0) \in \mathcal{U}_N^{\dagger}\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)$, $\underline{x}_0^{+} \in \mathcal{X}_N$ holds after applying the suboptimal control vector $\underline{u}^{\dagger}(0, \underline{x}_0)$ for closed-loop control. From (7.1.5) it follows that $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^{+}) = \underline{\Omega}\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big) \in \tilde{\mathcal{U}}_N(\underline{x}_0^{+})$ such that $\underline{\vartheta}^{+} \in \mathcal{Z}_N$ (see [All+17, Proof Thm. 14]). Asymptotic stability follows from [All+17, Prop. 13 and Proof of Thm. 14] and includes Assumptions 3.1.1-3.3.2. □

In contrast to the work in [SM15], this section expands on the framework of suboptimal MPC. By directly incorporating the manually generated warm-starts according to (7.1.5), the setting $\lambda^{\dagger} = 1$ and $\check{\underline{\mathbf{u}}}^{\dagger}(\underline{x}_0) = \underline{0}_{mM}$ ensures both recursive feasibility and asymptotic stability. Since the authors of [SM15] only include a fallback level resulting from the shift-truncate-append operator $\underline{\Omega}_{\text{sta}}(\cdot)$, their theoretical derivations mainly guarantee recursive feasibility. The operator $\underline{\Omega}_{\text{sta}}(\cdot)$ does not ensure that $J_N\big(\underline{x}_0, \underline{\Omega}_{\text{sta}}(\tilde{\underline{\mathbf{u}}}(\underline{x}_0))\big) \leq F(\underline{x}_0) \leq \alpha_{\text{f}}(\|\underline{x}_0\|)$ holds for all $\underline{x}_0 \in \text{lev}_{\pi}F$ with some small $\pi > 0$. This property, however, ensures that $J_N\big(\cdot, \underline{\Omega}_{\text{sta}}(\cdot)\big)$ is continuous at the origin. Suboptimal offset MBMPC inherits an important runtime property of classical suboptimal MPC. The following configuration enables a deterministic computation time. Let λ_i and λ_i^{+} be the values of λ after $i \in \mathbb{N}_0$ optimization iterations at time instant n and $n+1$, respectively. Analogous relation applies for $\check{\underline{\mathbf{u}}}_i^{+}$. By manually setting $\lambda_0^{+} := 1$ and $\check{\underline{\mathbf{u}}}_0^{+} := \underline{0}_{mM}$ at time instant n, the optimization algorithm always starts its routine at the warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0)$, which is inherently an element of $\mathcal{U}_N^{\dagger}\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)$. Now, with respect to a maximum execution time, the number of optimization iterations can be simply limited (early termination) without loss of important closed-loop properties. This warm-starting strategy of auxiliary optimization parameters follows the main idea in suboptimal MPC according to [Pan+11; All+17] but assumes that the ideal optimizer only produces feasible iterates. Since the parameter λ only affects the suboptimal solution $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0) \in \mathcal{U}_N^{\dagger}\big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)$, the formulation of the difference inclusion in (7.1.7) is still comprehensive.

Remark 7.3.1: *Shifting blocking pattern*. The input parameterization in (7.3.1) can be also expanded to time-varying blocking matrices whose evolution might follow the shifting strategy in [Cag+07] or in [GR20]. However, shifting the blocking pattern also affects the structure of first- or second-order derivative information. Here, a hypergraph formulation enables an efficient online structure/sparsity exploitation and addresses the use of sparse solvers in combination with sparse finite differences [Rös+18a].


$\tilde{\underline{u}}(1,\underline{x}_0)$
$\tilde{\underline{u}}(2,\underline{x}_0)$
$\tilde{\underline{u}}(3,\underline{x}_0)$
$\tilde{\underline{u}}(0,\underline{x}_0)$
$+$
$\check{\underline{u}}(1)$
$\check{\underline{u}}(0)$
0
1
2
3
4
k

Resulting inequalities:

$\check{\underline{u}}(0) + \lambda \max\{\tilde{\underline{u}}(0,\underline{x}_0), \tilde{\underline{u}}(1,\underline{x}_0)\} \leq \max \mathbb{U}$

$\check{\underline{u}}(0) + \lambda \min\{\tilde{\underline{u}}(0,\underline{x}_0), \tilde{\underline{u}}(1,\underline{x}_0)\} \geq \min \mathbb{U}$

$\check{\underline{u}}(1) + \lambda \max\{\tilde{\underline{u}}(2,\underline{x}_0), \tilde{\underline{u}}(3,\underline{x}_0)\} \leq \max \mathbb{U}$

$\check{\underline{u}}(1) + \lambda \min\{\tilde{\underline{u}}(2,\underline{x}_0), \tilde{\underline{u}}(3,\underline{x}_0)\} \geq \min \mathbb{U}$

Figure 7.1.: Linear inequalities for offset MBMPC with $M = 2$, uniform blocking, $N = 4$, $m = 1$, and input box-constraints. The white filled circles indicate missing degrees of freedom.

Remark 7.3.2: *Regularization.* Problem (7.3.2) can be regularized by including $r_{\mathrm{g}}\,(\lambda - 1)^2$ as an additional cost term with some small weighting parameter $r_{\mathrm{g}} > 0$.

The additional scaling parameter λ results in a dense column vector in the Jacobian of the constraints since the variation of this parameter affects the entire control trajectory. Reusing the previous solution as the warm-start requires the introduction of additional linear inequalities on the control variables such that $\underline{\Theta}_N\big(\check{\underline{\mathbf{u}}}, \tilde{\underline{\mathbf{u}}}(\underline{x}_0), \lambda\big) \in \mathbb{U}^N$ holds. Figure 7.1 visualizes an example of shifted move-blocked control sequences and provides the corresponding linear inequalities in case of input box-constraints.

7.4. Example Continued: Van der Pol Oscillator

This section follows up on the example in Section 5.2. Here, the MBMPC approaches can also inherit the stabilizing properties of the local controller $\underline{\kappa}(\underline{x}) := \underline{K}\,\underline{x}$ with $\underline{x} \in \mathbb{X}_{\mathrm{f}} := \mathrm{lev}_{\pi} F$. To enlarge the feasible sets $\bar{\mathcal{X}}_M$ for different degrees of freedoms $M \geq 1$, the horizon length is increased to $N = 80$. The following numerical investigations only consider uniform move-blocking patterns, such that all blocking intervals contain the same number of constant control steps. All related modifications compared to Section 5.2 are given below:

$$\mathbb{X} = \{\underline{x} \in X \,|\, (-1,-1)^{\mathsf{T}} \preceq \underline{x} \preceq (1,1)^{\mathsf{T}}\}, N = 80, \rho = 1.001, \pi = 0.5174.$$

The spread of the terminal region $\pi = 0.5174$ is determined by applying the heuristic search strategy described in Remark 3.3.3. Figure 7.2 compares the closed-loop control performances of MBMPC and conventional MPC starting from the initial state $\underline{x}_{\mu,0} = (0.8, 0)^{\mathsf{T}}$. The black dashed graph visualizes the initial open-loop solution with two degrees of freedom $M = 2$. Here, the last predicted state satisfies the terminal constraint such that $\underline{\varphi}\big(N, \underline{x}_0, \underline{\Theta}_N\big(\check{\underline{\mathbf{u}}}^*(\underline{x}_0)\big)\big) \in \mathrm{lev}_{\pi} F$ applies with $\underline{x}_0 = \underline{x}_{\mu,0}$. Recall that if common MBMPC does not integrate stabilizing fallback solutions, there is no guarantee that the underlying OCP is at least recursively feasible. Though the naive MBMPC (gray dash-dotted graphs), which is driven by the solutions to OCP (7.2.3), is recursively feasible in this particularly numerical example, the cost function in the upper right plot does not decrease in the sense of Lyapunov. Here, MBMPC is able to stabilize the origin, however, the reduction in degrees of freedom is clearly achieved at

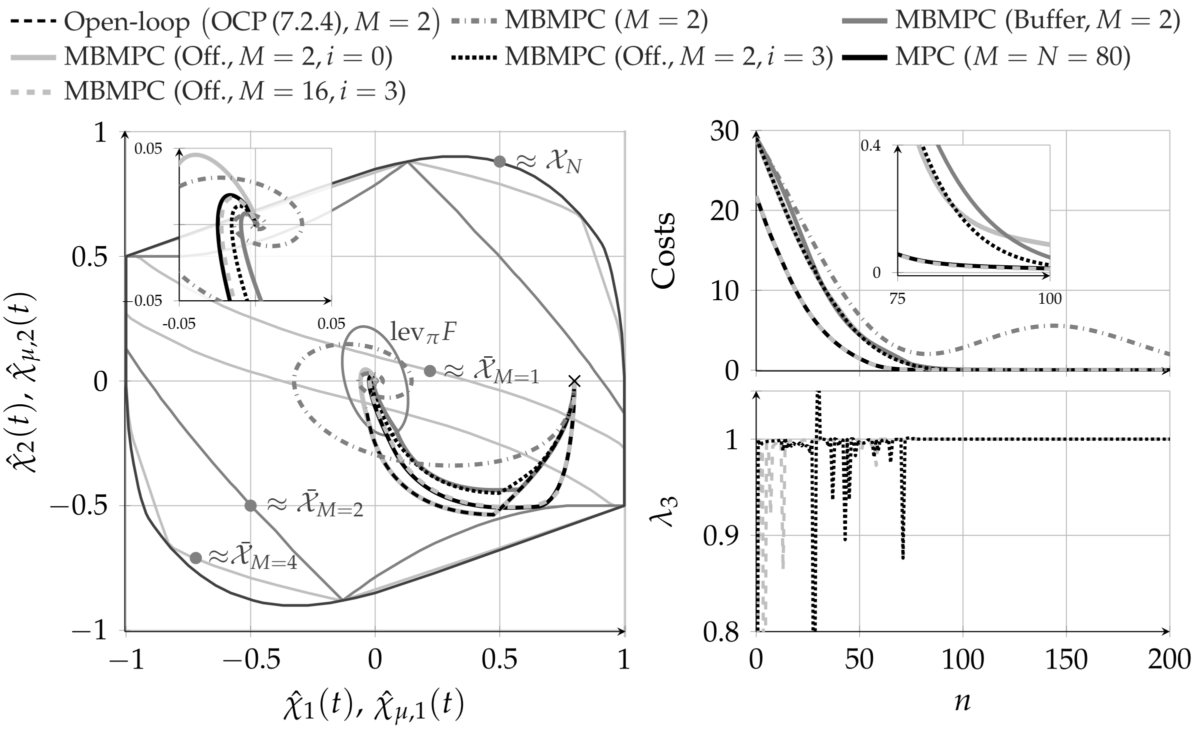


Figure 7.2.: Open- and closed-loop control of the Van der Pol oscillator subject to input move-blocking. Left: Phase-portrait. The subplot shows a close-up of the closed-loop behavior at the origin. Right, top: Evolution of costs over closed-loop steps. Right, bottom: Evolution of the scaling parameter λ after three optimization iterations over closed-loop steps.

the expense of an oscillating behavior and thus poorer closed-loop control performance. The first stabilizing input move-blocking approach holds back fallback solutions in some buffer according to Section 7.2. This straightforward approach already improves the closed-loop control performance (dark gray solid graphs). Note that since the closed-loop trajectory does not match the initially predicted state trajectory, the optimizer also finds solutions that are better than the fallback solutions in terms of costs. Though costs do not converge to zero as fast as with the following approaches, the cost function $J_N(\cdot)$ clearly satisfies Lyapunov's cost descent condition. The second approach applies offset MBMPC according to Section 7.3. First, the maximum number of optimization iterations is restricted to $i = 0$. Hence, the predictive controller simply reuses the open-loop solution until the closed-loop system reaches the terminal set $\text{lev}_\pi F$ (see light gray solid graphs). Then, the predictive controller switches to the local control law, which stabilizes the origin by design (see Asm. 3.3.2). The corresponding cost function evolution in the right upper plot decreases in the sense of Lyapunov. By increasing the maximum number of optimization iterations to $i = 3$, the closed-loop performance improves visibly (black dotted graphs) and the close-up in the right upper plot shows that the costs converge faster to zero compared to the case with $i = 0$. Obviously, conventional MPC with $N = M = 80$ (black solid graphs) outperforms the very restrictive MBMPC with $M = 2$. However, offset MBMPC with an application oriented configuration with $M = 16$ and only $i = 3$ nearly coincides with the performance of MPC (light gray dashed graphs). The lower right plot visualizes that the

optimizer exploits the additional degree of freedom in case of offset MBMPC especially at the beginning of the closed-loop control. By choosing values different from $\lambda_3 = 1$, the optimizer obviously improves the warm-start. As soon as the closed-loop system approaches the terminal region $\mathrm{lev}_\pi F$, the optimizer chooses $\lambda_3 = 1$ as it is very unlikely that a move-blocked control sequence outperforms a control sequence that is generated by iteratively applying the local control law. In addition, Figure 7.2 shows convex or at least compact approximations of the feasible sets for MBMPC and conventional MPC. Recall that by integrating stabilizing terminal conditions, the feasible sets also represent the regions of attraction. The approximations are determined by uniformly sampling the constrained state space $\mathbb{X}$ along each dimension. This procedure results in a uniform grid of 201×201 points. All the initial states $\utilde{x}$ for which an admissible move-blocked control trajectory $\utilde{\Theta}_N(\check{\underline{\mathbf{u}}}) \in \mathcal{U}_N(\utilde{x})$ exists are enclosed by a compact polytope. Here, a convex hull is preferred over a concave hull. However, the difference between the surface area of the exact compact boundary of all feasible grid points and the area resulting from an interpolated boundary, which is enlarged in the direction of the convex hull, is limited to differ in at most 3.5 %. Figure 7.2 verifies the relationship $\check{\mathcal{X}}_1 \subset \check{\mathcal{X}}_2 \subset \check{\mathcal{X}}_4 \subset \mathcal{X}_N = \check{\mathcal{X}}_{80}$ to hold. However, with only four degrees of freedom, the approximation of the feasible set $\check{\mathcal{X}}_4$ is already comparable to the approximation of $\mathcal{X}_N$. It is evident that in this example, a moderate reduction of the degrees of freedom in control ($M \geq 2$) does not decrease the closed-loop performance and the feasible region significantly. However, a single degree of freedom results in a small feasible region, which is not sufficient for real applications.
Table 7.1 summarizes the results of a quantitative analysis of closed-loop control performances and the corresponding computational load for different degrees of freedom in control and a uniform blocking. Again, the closed-loop control performances are evaluated on the basis of criterion $\bar{J}_{\mathrm{cl}}$ from (4.3.3) with $l = 1000$. The reference performance is the initial solution to the full degree of freedom OCP (3.1.6) with $N = 1000$, $\underline{\mathbf{u}}_{\mathrm{ref}} := \underline{\mathbf{u}}^*(\utilde{x}_{\mu,0})$, and $\utilde{x}_{\mu,0} = (0.8, 0)^\intercal$. Thus, here the criterion $\bar{J}_{\mathrm{cl}} \in (-\infty\,\%, 0\,\%]$ describes the closed-loop control performance loss due to input move-blocking and the finite horizon length. The normalized median $\bar{t}_{\mathrm{m}}$ includes 100 cold-started solutions to the individual OCPs at time instant $n = 0$ and is given by:

$$\bar{t}_{\mathrm{m}} = \left(1 - \frac{t_{\mathrm{m}}}{t_{\mathrm{m,ref}}}\right) 100\,\%. \tag{7.4.1}$$

The reference median value $t_{\mathrm{m,ref}}$ results from 100 cold-started solutions to the unblocked OCP (3.1.6) with $N = 80$. Thus, the normalized median $\bar{t}_{\mathrm{m}} \in [0\,\%, 100\,\%)$ indicates the execution time improvement resulting from the reduction of optimization variables. The discretization approaches, namely recursive elimination and multiple shooting, are evaluated separately from each other. All optimization variables are initialized to zero and the optimization framework IPOPT [WB06] is used to generate all the evaluation data. In case of multiple shooting, there exist as many shooting intervals $T = M$ as there are blocking intervals. Each shooting interval has the same length as the corresponding blocking interval. The reference median values with recursive elimination and multiple shooting are $t_{\mathrm{m,ref}} = 4.01\,\mathrm{s}$ and $t_{\mathrm{m,ref}} = 560.95\,\mathrm{ms}$, respectively. The first two rows of Table 7.1 relate to the naive MBMPC that does not incorpo-

Table 7.1.: Evaluation of closed-loop control performances and computation times for MBMPC applied to the Van der Pol oscillator. Abbreviations: Recursive Elimination (RE), Multiple Shooting (MS), Move-Blocking (MB).

	MBMPC,	$M = 40$		$M = 16$		$M = 8$		$M = 4$		$M = 2$	
	$N = 80$	$\bar{J}_{\mathrm{cl}}$ [%]	$\bar{t}_{\mathrm{m}}$ [%]	$\bar{J}_{\mathrm{cl}}$ [%]	$\bar{t}_{\mathrm{m}}$ [%]	$\bar{J}_{\mathrm{cl}}$ [%]	$\bar{t}_{\mathrm{m}}$ [%]	$\bar{J}_{\mathrm{cl}}$ [%]	$\bar{t}_{\mathrm{m}}$ [%]	$\bar{J}_{\mathrm{cl}}$ [%]	$\bar{t}_{\mathrm{m}}$ [%]
MB	RE	−0.05	56.61	−0.94	75.06	−6.03	86.36	−24.99	93.71	**−106.38**	**95.88**
	MS	−0.05	12.02	−0.94	34.6	−6.03	44.82	−24.99	52.54	**−106.38**	**52.94**
Buffer	RE	−0.04	56.36	−0.25	75.06	−1.44	86.39	−10.59	93.66	−32.36	95.86
	MS	−0.04	11.92	−0.25	34.27	−1.44	44.5	−10.59	52.18	−32.36	52.33
Offset	RE	0.00	41.15	**−0.01**	**68.83**	−1.18	81.99	−10.7	91.84	**−33.29**	**93.74**
	MS	0.00	4.3	**−0.01**	**36.06**	−1.18	31.23	−10.7	54.94	**−33.29**	**52.33**

rate stabilizing fallback or warm-start control sequences. Since input move-blocking does not destabilize the origin with the chosen benchmark system, the impact of the gradual reduction of complexity on both the closed-loop control performance and the computational effort can be investigated systematically. While a moderate reduction of the degrees of freedom up to $M \geq 8$ leads to an almost negligible degradation of the performance, a significant reduction is observed from $M \leq 4$ on. Hence, the reduction of the computational load decreases at the expense of the closed-loop control performance. Furthermore, MBMPC with recursive elimination benefits significantly more from the reduction of optimization variables than MPC with multiple shooting in terms of runtime performance. In case of recursive elimination, the number of non-zero elements in the Jacobian of the constraints decreases more strongly compared to the multiple shooting case. With $M = 2$, the closed-loop control performance drops to approximately twice the cost level of the unblocked realization with $\bar{J}_{\mathrm{cl}} = -106.38\,\%$. Further, with $M = 2$, recursive elimination outperforms multiple shooting with $t_{\mathrm{m}} = t_{\mathrm{m,ref}}\,(1 - \bar{t}_{\mathrm{m}}/100\,\%) = 4.01\,\mathrm{s}\,(1 - 95.88\,\%/100\,\%) = 165.21\,\mathrm{ms}$ and $t_{\mathrm{m}} = 560.95\,\mathrm{ms}\,(1 - 52.94\,\%/100\,\%) = 263.98\,\mathrm{ms}$, respectively. The third and the fourth two lines of Table 7.1 result from applying the buffer-based approach from Section 7.2. While the computational effort for all evaluated degrees of freedom M remain at a comparable level, the corresponding closed-loop control performances improve significantly. Thus, ensuring stability with MBMPC has the side effect of increasing control performance, although the stability guarantees in this dissertation are not based on optimality. The reason for this improvement stems from the fact that the linearization of the nonlinear system at the origin leads to a relatively large terminal region $\mathrm{lev}_{\pi}F$, where the local controller outperforms MPC with input-move blocking. Finally, the last two rows of Table 7.1 show the evaluation of the offset-based MBMPC according to Section 7.3. Here, the low computational overhead resulting from the additional linear input constraints, as shown in Figure 7.1, is partially observable as for $M = 40$. However, with $M = 2$ both stabilizing move-blocking approaches improve the closed-loop control performance by approximately a factor of three from $\bar{J}_{\mathrm{cl}} = -106.38\,\%$ to $\bar{J}_{\mathrm{cl}} = -33.29\,\%$. As already observed in Figure 7.2, a moderate reduction of the degrees of freedom in control to $M = 16$ reduces the computational load by 68.83 % with recursive elimination and by 36.06 % with multiple shooting. The loss in the closed-loop control performance is negligible with $\bar{J}_{\mathrm{cl}} = -0.01\,\%$. Based on the performance results in Table 7.1 and the fact that the offset-based approach offers a holistic problem formulation in (7.3.2), offset MBMPC emerges as the clear favorite.

7.5. Discussion

Section 2.4 reveals that asymptotic stability with MBMPC is still an open problem (see also Table A.1 in App. A.6). There is no design recommendation for MBMPC in the literature that aims at guaranteeing both recursive feasibility and asymptotic stability for general nonlinear systems. By defining input move-blocking as an explicit source of suboptimality, important closed-loop properties directly follow from the suboptimal MPC framework presented, for example, in [Sco+99; Pan+11; All+17; Raw+20]. The connection of input move-blocking and suboptimal MPC has not been formalized yet in the literature and is therefore novel (except [Mak+22]). Since the proposed extensions address nonlinear systems and arbitrary blocking patterns, the number of degrees of freedom in control and the blocking configuration can be chosen depending on the dedicated application. The numerical example in this chapter shows that input-move blocking is clearly an intuitive approach for controller tuning when computational effort is important besides control performance. By implementing suboptimal offset MBMPC according to Section 7.3, the maximum number of optimization iterations can be simply limited for real-time applications without losing important closed-loop control properties. This early termination feature of conventional suboptimal MPC is also highlighted, for example, in [RR17a]. The evaluation and discussion so far refer to the case that there does not exist an optimal solution at the first closed-loop time instant $n = 0$. Recall that Assumption 7.2.1 only requests an arbitrary but admissible solution. If an optimal solution is available to the unblocked OCP (3.1.6) at time instant $n = 0$, offset MBMPC might be inferior compared to classical suboptimal MPC although more optimization steps can be performed in the same time due to the reduced complexity. While the theoretical derivations require Assumption 7.2.1, for practical applications, however, it is implicitly assumed that the closed-loop system is driven inside the feasible set $\bar{\mathcal{X}}_M$ and that the optimizer can always find a move-blocked but admissible solution at time instant $n = 0$ within the real-time constraints.
Now the question arises, to what extent the results from this chapter can be transferred to SFMPC. In general, both design variations of suboptimal MBMPC apply to SFMPC with smooth optimization. While the first buffer-based realization also applies to SFMPC with exhaustive search, suboptimal offset MBMPC increases the combinatorial complexity since the value range of the scaling parameter λ has to be discretized, too. However, the phase portrait in Figure 7.2 reveals that the single degree of freedom in control significantly reduces the feasible set. Recall that in this chapter the setting $\mathbb{X}_{\mathrm{f}} := \mathrm{lev}_{\pi} F$ applies such that a single degree of freedom in control complicates transferring the nonlinear system (open-loop) from some initial state to the terminal set. The main idea of improving an already stabilizing warm-start is the main foundation of the extended SFMPC formulation in the next chapter.

8 Asymptotic Stabilization with Input Move-Blocking and Finite Control Sets

This chapter aims at both ensuring theoretical closed-loop properties by design and enlarging the region of attraction of SFMPC with stabilizing terminal conditions. Introducing a variable horizon as a first step addresses the demand for stabilizing closed-loop properties. A variable partitioning of a fixed horizon length includes the beneficial stabilizing properties of a variable horizon formulation and increases the flexibility of SFMPC by applying the local control law on the second part of the horizon, even outside the terminal region. The single degree of freedom in control allows to introduce time-varying finite control sets. Parts of this chapter have been published in [Mak+20] and [Mak+21].

8.1. Variable Horizon Formulation

If the horizon length is configured to be an optimization variable with $N \in \mathcal{N} := [N_{\min}, N_{\max}], N_{\min}, N_{\max} \in \mathbb{N}, N_{\max} \geq N_{\min} \geq M$, the naive move-blocking OCP (7.2.3) turns into the following mixed-integer OCP:

$$V_{N^*}(\underset{\sim}{x}_0) := \min_{N \in \mathcal{N}, \underline{\check{\mathbf{u}}} \in \mathbb{U}^M} J_N\big(\underset{\sim}{x}_0, \underset{\sim}{\Theta}_N(\underline{\check{\mathbf{u}}})\big) \quad \text{subject to} \quad \underset{\sim}{\Theta}_N(\underline{\check{\mathbf{u}}}) \in \mathcal{U}_N(\underset{\sim}{x}_0). \tag{8.1.1}$$

To avoid redefining equations, the horizon length as a subscript is treated as an implicit input argument of the cost function $J_N(\cdot)$ and the mapping $\underset{\sim}{\Theta}_N(\cdot)$. Let the optimal solution tuple to OCP (8.1.1), if it exists, be denoted by $\big(N^*, \underline{\check{\mathbf{u}}}^*(\underset{\sim}{x}_0)\big)$. Assume, for the moment, that move-blocking is disabled with $M = N$ such that $\underset{\sim}{\Theta}_N\big(\underline{\check{\mathbf{u}}}^*(\underset{\sim}{x}_0)\big) = \underline{\check{\mathbf{u}}}^*(\underset{\sim}{x}_0) = \underline{\mathbf{u}}^*(\underset{\sim}{x}_0)$ holds. Further assume that the optimizer can simply adjust the horizon length by iterating over the finite interval $\mathcal{N}$ in some outer optimization loop. If OCP (3.1.6) is at least feasible for a single $N \in \mathcal{N}$ for $\underset{\sim}{x}_0 \in \mathcal{X}_N$, then OCP (8.1.1) is also feasible for the same initial state $\underset{\sim}{x}_0$. MPC formulations with a variable horizon mainly originate from the dual-mode control in [MM93]. Here, variable horizon MPC warm-starts the next OCP at time instant $n+1$ with the optimal but shortened control sequence as long as $N^* \geq 2$, which is guaranteed to be admissible in the nominal case. Let $\underset{\sim}{\Omega}_{\text{st}} : \mathbb{U}^N \mapsto \mathbb{U}^{N-1}$ be the operator that shifts and truncates a control sequence by

one step. Since it does not append a control vector at the end of the control sequence, it does not depend on the initial state $\underline{x}_0$ as $\Omega_{\mathrm{sta}}(\cdot)$. With $M = N$ and $N \geq 2$, Bellman's principle of optimality can be used to prove that the implicit control law $\underline{\mu}(\underline{x}_0) := \underline{u}^*(0, \underline{x}_0)$ transfers the closed-loop system (3.3.2) in at most N^* steps into the terminal set $\mathbb{X}_\mathrm{f}$ [MM93, Prop. 1]. For example, if there exists a global optimum at closed-loop time instant n with $\underline{\mathbf{u}}^*(\underline{x}_0) = \big(\underline{u}^*(0, \underline{x}_0), \underline{u}^*(1, \underline{x}_0), ..., \underline{u}^*(N^* - 1, \underline{x}_0)\big)$, the shifted and truncated control sequence $\underline{\mathbf{u}}^*(\underline{x}_0^+) = \Omega_{\mathrm{st}}\big(\underline{\mathbf{u}}^*(\underline{x}_0)\big) = \big(\underline{u}^*(1, \underline{x}_0), \underline{u}^*(2, \underline{x}_0), ..., \underline{u}^*(N^* - 2, \underline{x}_0)\big)$ and the horizon $N^* - 1$ represent the global optimal tuple at time instant $n + 1$. Inside the terminal set $\mathbb{X}_\mathrm{f}$, explicitly switching to the local controller $\underline{\kappa}(\cdot)$ stabilizes the origin for the nonlinear system in the sense of Lyapunov [MM93, Thm. 1, Thm. 2].
In case of active move-blocking with $M < N$, the blocking matrix $\underline{\check{B}} \in \mathbb{R}^{N \times M}$ needs to be adjusted in its dimensions as soon as the horizon length is subject to variation, such that the optimizer can find the shifted and truncated control sequence repeatedly to ensure at least recursive feasibility for N^* steps [SM12, Thm. 16]. The following illustrative evolution of blocking matrices over closed-loop control steps, which is formalized in [SM12; She12] (see also [Cag+07]), would ensure recursive feasibility of OCP (8.1.1) with active move-blocking:

$$\underline{\check{B}} := \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, \begin{matrix} M = 2, \\ N = 4, \\ n = 0. \end{matrix} \rightarrow \underline{\check{B}} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, \begin{matrix} M = 2, \\ N = 3, \\ n = 1. \end{matrix} \rightarrow \underline{\check{B}} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{matrix} M = 2, \\ N = 2, \\ n = 2. \end{matrix}. \tag{8.1.2}$$

Since the blocking pattern changes after adjusting the blocking matrix, the principle of optimality does not hold anymore. When the blocking pattern varies as the closed system evolves, the optimizer might find a better solution such that the previous but shifted solution only represents an upper bound on the costs. The authors in [RH06, Thm. 1] and [SM12, Thm. 17] show finite completion times to pre-defined target regions in case of variable horizon MPC without and with move-blocking, respectively. With the blocking adaptation according to (8.1.2), a finite completion time to the terminal set $\mathbb{X}_\mathrm{f}$ can be derived from the following inequalities ([SM12, Thm. 17], [She12, Sec. 2.4.1]):

$$\begin{aligned} V_{N^*}(\underline{x}_0^+) - V_{N^*}(\underline{x}_0) \leq J_{N^*-1}\Big(\underline{x}_0^+, \Omega_{\mathrm{st}}\big(\Theta_{N^*}(\underline{\check{\mathbf{u}}}^*(\underline{x}_0))\big)\Big) - V_{N^*}(\underline{x}_0) = -\ell\big(\underline{x}_0, \underline{\check{u}}^*(0, \underline{x}_0)\big) \leq \\ \leq -\alpha_\ell(\|\underline{x}_0\|). \end{aligned} \tag{8.1.3}$$

Since the function $\alpha_\ell(\cdot)$ is only zero at the origin (see Asm. 3.3.1), the optimal cost function value decreases continuously after applying the implicit control law $\underline{\mu}(\underline{x}_0) := \underline{\check{u}}^*(0, \underline{x}_0)$. Though the principle of optimality does not hold with input move-blocking, the controller nevertheless transfers the closed-loop system in a finite number of steps into the terminal region $\mathbb{X}_\mathrm{f}$. Then, either manually switching to the local control law $\underline{\kappa}(\cdot)$ as in [MM93] or letting the optimizer search for the optimal unblocked solution stabilizes the origin. The latter case represents conventional MPC with stabilizing terminal conditions. By applying a variable horizon, input move-blocking is disabled automatically as soon as the horizon reduces to $N = M$. This relationship is represented by the last blocking matrix in (8.1.2) and the discussion in the Special Case I in

Section 4.1 ($M = N = 1$). Recall that with a single degree of freedom in control, the blocking matrix is given by $\underline{\check{B}} = \underline{1}_N$ with $M = 1$. Therefore, matching the blocking pattern to the horizon length is trivial. The length of the first dimension of $\underline{\check{B}}$ is simply linked to the chosen horizon length. Before introducing a variable horizon in combination with a single degree of freedom in control, the following closed finite union is introduced:

$$\bar{\mathcal{X}}_{\mathrm{s}} := \bigcup_{i=1}^{N_{\max}} \mathcal{X}_i^{\mathrm{s}}. \tag{8.1.4}$$

Since this chapter is dedicated to the development of an extended formulation of SFMPC with finite control sets, the following mixed-integer OCP directly includes the extended finite control set $\tilde{\mathbb{A}}(\underline{x}_0, n)$:

$$V_{N^*}^{\mathrm{s}}(\underline{x}_0, n) := \min_{N \in \mathcal{N},\, \underline{u}_{\mathrm{s}} \in \tilde{\mathbb{A}}(\underline{x}_0, n)} J_N\big(\underline{x}_0, \underline{\Theta}_N^{\mathrm{s}}(\underline{u}_{\mathrm{s}})\big), \text{ subject to } \underline{\Theta}_N^{\mathrm{s}}(\underline{u}_{\mathrm{s}}) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0). \tag{8.1.5}$$

Let $\big(N^*, \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)\big)$ denote the optimal solution tuple to OCP (8.1.5), which leads to the control sequence $\underline{\Theta}_{N^*}^{\mathrm{s}}(\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0)$. From the variable horizon formulation, the following closed-loop properties can be derived for Single Degree of Freedom Variable Horizon MPC (SFVMPC).

Proposition 8.1.1: *SFMPC with variable horizon and finite control sets*. Suppose Assumptions 3.1.1-3.3.2 and 4.5.1 hold. Then, the implicit control law $\underline{\mu}(\underline{x}_0, n) := \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)$, which is driven by the solutions to the mixed-integer OCP (8.1.5), renders the feasible set $\bar{\mathcal{X}}_{\mathrm{s}}$ positive invariant and the origin asymptotically stable in the sense of Lyapunov for the closed-loop system (4.5.2).

The detailed proof is given in Appendix B.2. However, the proof of Proposition 8.1.1 is straightforward. Assumption 4.5.1 ensures that the very first OCP at time instant $n = 0$ is feasible. This implies that there exists an admissible tuple $(\underline{u}_{\mathrm{s}}^*(\underline{x}_0, 0) = \underline{u}_{\mathrm{s}}^0, N^*)$ for every $\underline{x}_0 \in \bar{\mathcal{X}}_{\mathrm{s}}$. Then, the definition of the finite control set $\mathbb{A}(n)$ in (4.5.4), which embeds the previous closed-loop control vector $\underline{\mu}(\underline{x}_\mu(n-1), n-1)$ into $\mathbb{A}(n)$, ensures recursive feasibility for N^* closed-loop steps by successively reducing the horizon by one step (see Rem. 4.5.1). Equation (8.1.3) ensures, in addition, finite completion times for all $\underline{x}_0 \in \bar{\mathcal{X}}_{\mathrm{s}}$. As soon as the optimal horizon reduces to $N^* = 1$, $\underline{x}_0 \in \mathcal{X}_1^{\mathrm{s}}$ applies. Therefore, the optimizer can fix the horizon to $N = 1$ and emulate conventional and thus stabilizing MPC according to Corollary 4.5.1.

To solve the mixed-inter OCP (8.1.5) exactly, the corresponding optimizer needs to seek the optimal horizon length N^* at every closed-loop time instant. Therefore, in the worst-case, the optimizer needs to evaluate all possible horizon lengths $N \in \mathcal{N}$. The derivation of finite completion in [SM12] (see (8.1.3)) shows that shortening the horizon length by one step, as the closed-loop system evolves by one step, suffices to ensure progress towards the terminal set $\mathbb{X}_{\mathrm{f}}$. The derivations in this section are intended, in particular, to introduce the key stability ingredient required to ensure theoretical closed-loop properties for SFMPC with a horizon length of $N > 1$. Introducing a variable horizon in OCP (8.1.5) somewhat mitigates the negative impact of the extreme input blocking in SFMPC. However, the nonlinear system has to be still transferred (open-loop) to the terminal set $\mathbb{X}_\mathrm{f}$ at constant control. Therefore, it might be the case that the feasible set $\bar{\mathcal{X}}_\mathrm{s}$ is too restrictive for practical control applications. To enlarge the feasible set and to allow suboptimal horizon lengths, the findings from Chapter 7 on suboptimal MBMPC are combined with the idea of a relaxing extreme move-blocking by introducing a variable horizon in the next section.

8.2. Extended Horizon Formulation

This section is mainly motivated by the OCP formulations in [ZA98] and [Mag+01]. In both contributions, the major idea is to divide a moving horizon of fixed length into two sections. While the control variables on the first section are subject to optimization, the control variables on the second part are determined using a local (linear) state controller. Here, the local controller is already applied outside some control invariant terminal set $\mathbb{X}_\mathrm{f}$. The authors of [ZA98] restrict the first section to only one step and saturate the control variables resulting from the application of the local controller on the second section. Closed-loop stability properties are ensured by imposing the stabilizing terminal conditions presented in [CA98]. In contrast, the approach presented in [Mag+01] allows the user to provide an arbitrary length of the first section (control horizon). Closed-loop stability properties mainly follow from the monotonicity property of the optimal cost function with respect to the length of the control horizon (see [Mag+01, Proof of Thm. 2]). If now enforcing input move-blocking on the first part of the horizon, the monotonicity property with respect to the control horizon does not hold anymore. It follows that increasing the length of the control horizon while keeping the total horizon length constant does not generally lead to lower costs. Therefore, the following derivations combine suboptimal MBMPC from Chapter 7 with a variable horizon partition. With only one degree of freedom on the first section, the control sequence of fixed length adopts the following structure:

$$\begin{aligned}\underline{\mathbf{u}}_\mathrm{e}(\underset{\sim}{x}_0, \underline{u}_\mathrm{s}, N_1) := \Big(&\underline{u}(0) = \underline{u}_\mathrm{s}, \underline{u}(1) = \underline{u}_\mathrm{s}, \dots, \underline{u}(N_1 - 1) = \underline{u}_\mathrm{s},\\ &\underline{\tilde{\kappa}}\big(\underset{\sim}{x}(N_1)\big), \underline{\tilde{\kappa}}\Big(\underline{f}\big(\underset{\sim}{x}(N_1), \underline{\tilde{\kappa}}(\underset{\sim}{x}(N_1))\big)\Big), \dots\Big) \in U^N.\end{aligned} \tag{8.2.1}$$

Here, $\underset{\sim}{x}(N_1)$ substitutes $\underline{\varphi}(N_1, \underset{\sim}{x}_0, \underline{\mathbf{u}}_\mathrm{s})$ with some $\underline{\mathbf{u}}_\mathrm{s} \in U^{N_1}$. The local control law $\underline{\tilde{\kappa}}(\cdot)$ is either defined by $\underline{\tilde{\kappa}}(\underset{\sim}{x}_0) := \underline{\Gamma}\big(\underline{\kappa}(\underset{\sim}{x}_0)\big)$ or by $\underline{\tilde{\kappa}}(\underset{\sim}{x}_0) := \underline{\kappa}(\underset{\sim}{x}_0)$ and applied for N_2 time steps following a recursive scheme. As proposed in [ZA98], the resulting control values can be restricted to the admissible control set by applying the saturation function $\underline{\Gamma} : U \mapsto \mathbb{U}$ if $\underset{\sim}{x}_0 \notin \mathbb{X}_\mathrm{f}$. The compact set of all admissible control sequences for a fixed horizon length $N_1 \in [1, N]$ is thus given by:

$$\begin{aligned}\mathcal{U}_N^\mathrm{e}(\underset{\sim}{x}_0, N_1) := \{\underline{\mathbf{u}} \in \mathbb{U}^N \mid\ &\underline{\varphi}(k, \underset{\sim}{x}_0, \underline{\mathbf{u}}) \in \mathbb{X}, \forall\, k = 0, 1, \dots, N-1,\ \underline{\varphi}(N, \underset{\sim}{x}_0, \underline{\mathbf{u}}) \in \mathbb{X}_\mathrm{f},\\ &\underline{u}(k_1) = \underline{u}(0), \forall\, k_1 = 1, \dots, N_1 - 1\} \subset \mathcal{U}_N(\underset{\sim}{x}_0).\end{aligned} \tag{8.2.2}$$

The relationship $\underline{\mathbf{u}}_\mathrm{e}(\underset{\sim}{x}_0, \underline{u}_\mathrm{s}, N_1) \in \mathcal{U}_N^\mathrm{e}(\underset{\sim}{x}_0, N_1)$ implies that there exists a region of attraction of the local controller that might extend beyond the borders of the terminal set $\mathbb{X}_\mathrm{f}$. The definition in (8.2.2) merely requires that the local control law $\underline{\tilde{\kappa}}(\cdot)$ transfers the nonlinear system (3.1.1) into the terminal set $\mathbb{X}_\mathrm{f}$ in a finite number of steps without violating state (and input) constraints. Let $\underline{\Omega}_{\tilde{\kappa}} : X \mapsto \mathbb{U}^N$ be the operator that applies the local control law $\underline{\tilde{\kappa}}(\cdot)$ for N times such that $\underline{\Omega}_{\tilde{\kappa}}(\underline{x}_0) = \big(\underline{\tilde{\kappa}}(\underline{x}_0), \underline{\tilde{\kappa}}\big(\underline{f}(\underline{x}_0, \underline{\tilde{\kappa}}(\underline{x}_0))\big), \ldots\big)$ applies. Then, subsets of the region of attraction of the local controller can be defined as follows:

$$\mathbb{X}_{\tilde{\kappa}}(N) := \{\underline{x} \in \mathbb{X} \,|\, \underline{\Omega}_{\tilde{\kappa}}(\underline{x}) \in \mathcal{U}_N(\underline{x})\}. \tag{8.2.3}$$

Inside the set $\mathbb{X}_{\tilde{\kappa}} \backslash \mathbb{X}_\mathrm{f}$, the local controller does not necessarily have to satisfy the CLF from (3.3.5). With these restrictions and formulations, the closed feasible set for a given first horizon length N_1 is denoted by:

$$\mathcal{X}_N^\mathrm{e}(N_1) := \{\underline{x}_0 \in \mathbb{X} \,|\, \mathcal{U}_N^\mathrm{e}(\underline{x}_0, N_1) \neq \emptyset\}. \tag{8.2.4}$$

Finally, the feasible state space for arbitrary first horizon lengths results from the following finite union:

$$\bar{\mathcal{X}}_N^\mathrm{e} := \bigcup_{i=1}^{N} \mathcal{X}_N^\mathrm{e}(i). \tag{8.2.5}$$

Recall that a finite union of closed sets is closed. The following OCP integrates the proposed variable horizon partition into the suboptimal MPC framework:

$$\begin{aligned} \min_{\underline{u}_\mathrm{s} \in \bar{\mathbb{P}}, N_1 \in \mathbb{N}, N_2 \in \mathbb{N}_0} \quad & J_N\big(\underline{x}_0, \underline{\mathbf{u}}_\mathrm{e}(\underline{x}_0, \underline{u}_\mathrm{s}, N_1)\big), \\ \text{subject to } \underline{\mathbf{u}}_\mathrm{e}(\underline{x}_0, \underline{u}_\mathrm{s}, N_1) \in \mathcal{U}_N^\dagger\big(\underline{x}_0, \underline{\tilde{\mathbf{u}}}(\underline{x}_0)\big) & \text{ and } N_1 + N_2 = N. \end{aligned} \tag{8.2.6}$$

The placeholder set is initially defined by $\bar{\mathbb{P}} := \mathbb{U}$. Let $\big(\underline{u}_\mathrm{s}^\dagger(\underline{x}_0), N_1^\dagger(\underline{x}_0), N_2^\dagger(\underline{x}_0)\big)$ denote a suboptimal solution tuple to OCP (8.2.6) from which the control sequence $\underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_0, \underline{u}_\mathrm{s}^\dagger(\underline{x}_0), N_1^\dagger(\underline{x}_0)\big) \in \mathcal{U}_N^\dagger\big(\underline{x}_0, \underline{\tilde{\mathbf{u}}}(\underline{x}_0)\big)$ can be derived. Here, the term suboptimality implies, similar to suboptimal MPC, that the control sequence $\underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_0, \underline{u}_\mathrm{s}^\dagger(\underline{x}_0), N_1^\dagger(\underline{x}_0)\big)$ is admissible and reaches lower or equal costs compared to the warm-start $\underline{\tilde{\mathbf{u}}}(\underline{x}_0)$. MPC that relies on OCP (8.2.6) is referred to as Single Degree of Freedom Extended Horizon MPC (SFEMPC) in the remainder of this dissertation. Similar to the basic buffer-based move-blocking approach in Section 7.2, the suboptimal MPC formulation in (7.1.2) is enhanced subsequently by the special control parameterization in (8.2.1). Therefore, there is no guarantee that the resulting OCP (8.2.6) is feasible for all $\underline{x}_0 \in \mathcal{X}_N$. The following investigations focus on theoretical closed-loop properties with SFEMPC.

Buffering Stabilizing Warm-Starts

The straightforward approach involves re-establishing a fallback level similar to (7.2.5) for all $\underline{x}_0 \in \mathcal{X}_N$:

$$\underline{\mathbf{u}}^\dagger(\underline{x}_0) := \begin{cases} \underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_0, \underline{u}_\mathrm{s}^\dagger(\underline{x}_0), N_1^\dagger(\underline{x}_0)\big) & \text{if OCP (8.2.6) is feasible for } \underline{x}_0 \in \mathcal{X}_N, \\ \underline{\tilde{\mathbf{u}}}(\underline{x}_0) & \text{otherwise.} \end{cases} \tag{8.2.7}$$

If OCP (8.2.6) is infeasible, SFEMPC simply reuses the stabilizing warm-starts, which are generated manually according to (7.1.5). The buffer-based approach ensures the following closed-loop properties.

Proposition 8.2.1: *SFEMPC with fallback level.* Suppose Assumptions 3.1.1-3.1.3 and 7.2.1 hold. Assume that the suboptimal control sequences are generated based on the fallback level in (8.2.7) and the suboptimal solutions to OCP (8.2.6). Then, the origin is asymptotically stable for the closed-loop system (7.1.7) in the positive invariant set $\mathcal{Z}_N$.

Proof. The proof follows the derivations in the proof of Proposition 7.2.1. Here, only the fallback level (7.2.5) needs to be replaced by the fallback level (8.2.7). □

The fallback level in (8.2.7) allows to use an initial control sequence that is not necessarily subject to input move-blocking with $\tilde{\underline{\mathbf{u}}}(\underline{x}_0) \notin \bigcup_{i=1}^{N} \mathcal{U}_N^{\mathrm{e}}(\underline{x}_0, i)$ (see Asm. 7.2.1). Therefore, stability properties hold on the set $\mathcal{Z}_N$ though $\mathcal{X}_N^{\mathrm{e}} \subseteq \mathcal{X}_N$. However, in contrast to the buffer-based move-blocking approach in Section 7.2, SFEMPC requires the fallback level only until OCP (8.2.6) turns feasible as shown in the next subsection.

Reproducing Stabilizing Warm-Starts

Lemma 8.2.1 shows that if OCP (8.2.6) is feasible for some $\underline{x}_0 \in \mathcal{X}_N$, the variable horizon partitioning only allows improvement of manually generated warm-starts.

Lemma 8.2.1: *Improvement of stabilizing warm-starts.* Suppose there exists a suboptimal solution tuple $\big(\underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0), N_1^{\dagger}(\underline{x}_0), N_2^{\dagger}(\underline{x}_0)\big)$ to OCP (8.2.6) with $\underline{x}_0 \in \mathcal{X}_N$. Then, the following cost descent property holds with $\underline{x}_0^+ = \underline{f}\big(\underline{x}_0, \underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0)\big)$ and $\underline{\Omega}(\cdot)$ from (7.1.5):

$$J_N\Big(\underline{x}_0^+, \underline{\mathbf{u}}_{\mathrm{e}}\big(\underline{x}_0^+, \underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0^+), N_1^{\dagger}(\underline{x}_0^+)\big)\Big) \leq J_N\Big(\underline{x}_0^+, \underline{\Omega}\big(\underline{x}_0, \underline{\mathbf{u}}_{\mathrm{e}}\big(\underline{x}_0, \underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0), N_1^{\dagger}(\underline{x}_0)\big)\big)\Big). \quad (8.2.8)$$

Proof. Let the definition $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0) := \underline{\mathbf{u}}_{\mathrm{e}}\big(\underline{x}_0, \underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0), N_1^{\dagger}(\underline{x}_0)\big)$ hold. Then, the warm-start is given by $\tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) := \underline{\Omega}\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big)$ (see (7.1.5)).
Case I: $N_1^{\dagger}(\underline{x}_0) > 1$, $\underline{x}_0^+ \notin \mathbb{X}_{\mathrm{f}}$. The optimizer reproduces the warm-start $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0^+) = \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) = \underline{\Omega}_{\mathrm{sta}}\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big)$ with the tuple $\big(\underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0^+) = \underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0)$, $N_1^{\dagger}(\underline{x}_0^+) = N_1^{\dagger}(\underline{x}_0) - 1$, $N_2^{\dagger}(\underline{x}_0^+) = N_2^{\dagger}(\underline{x}_0) + 1\big)$.
Case II: $N_1^{\dagger}(\underline{x}_0) > 1$, $\underline{x}_0^+ \in \mathbb{X}_{\mathrm{f}}$. If $J_N\big(\underline{x}_0^+, \underline{\Omega}_{\kappa}(\underline{x}_0^+)\big) > J_N\big(\underline{x}_0^+, \underline{\Omega}_{\mathrm{sta}}\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big)\big)$, the proof follows Case I. Otherwise, the warm-start has to be generated by $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0^+) = \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) = \underline{\Omega}_{\kappa}(\underline{x}_0^+)$. The optimizer invokes the recursive application of the local control law for N steps with $\big(\underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0^+) = \underline{\kappa}(\underline{x}_0^+), N_1^{\dagger}(\underline{x}_0^+) = 1, N_2^{\dagger}(\underline{x}_0^+) = N - 1\big)$.
Case III: $N_1^{\dagger}(\underline{x}_0) = 1$, $\underline{x}_0^+ \notin \mathbb{X}_{\mathrm{f}}$. This case implies that the different warm-starting approaches result in equal costs with $J_N\big(\underline{x}_0^+, \underline{\Omega}_{\kappa}(\underline{x}_0^+)\big) = J_N\big(\underline{x}_0^+, \underline{\Omega}_{\mathrm{sta}}\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big)\big)$. The optimizer can select the tuple $\big(\underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0^+) = \underline{\kappa}(\underline{x}_0^+), N_1^{\dagger}(\underline{x}_0^+) = 1, N_2^{\dagger}(\underline{x}_0^+) = N - 1\big)$ to reproduce the warm-start $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0^+) = \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) = \underline{\Omega}_{\kappa}(\underline{x}_0^+)$.
Case IV: $N_1^{\dagger}(\underline{x}_0) = 1$, $\underline{x}_0^+ \in \mathbb{X}_{\mathrm{f}}$. See Case III.
It follows that at least $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0^+) = \tilde{\underline{\mathbf{u}}}(\underline{x}_0^+) = \underline{\mathbf{u}}_{\mathrm{e}}\big(\underline{x}_0^+, \underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0^+), N_1^{\dagger}(\underline{x}_0^+)\big)$ applies for all cases. Since $J_N\big(\underline{x}_0^+, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0^+)\big) \leq J_N\big(\underline{x}_0, \underline{\mathbf{u}}^{\dagger}(\underline{x}_0)\big)$ holds by design, (8.2.8) is established. □

Note that Lemma 8.2.1 does not explicitly consider the evolution of the closed-loop system (7.1.7). The derivations so far only show that the variable horizon partition formulation in OCP (8.2.6) can reproduce the control sequences that are generated by the warm-starting approaches $\underline{\Omega}_{\kappa}(\cdot)$ and $\underline{\Omega}_{\mathrm{sta}}(\cdot)$. In contrast to suboptimal offset MBMPC in Section 7.3, here the optimizer recovers the manually generated warm-start by adjusting the horizon lengths N_1 and N_2, instead of adjusting an auxiliary parameter λ as in OCP (7.3.2). However, obviously, an initial solution tuple $\big(\underline{u}_\mathrm{s}^\dagger(\underline{x}_0), N_1^\dagger(\underline{x}_0), N_2^\dagger(\underline{x}_0)\big)$ to OCP (8.2.6) can only exist if and only if $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$ and $\underline{\tilde{\mathbf{u}}}(\underline{x}_0) \in \bigcup_{i=1}^N \mathcal{U}_N^\mathrm{e}(\underline{x}_0, i)$. To include the cost descent property from Lemma 8.2.1 into a theoretical closed-loop analysis, the set of extended states $\mathcal{Z}_N$ is restricted to

$$\bar{\mathcal{Z}}_N := \big\{\big(\underline{x}_0, \underline{\tilde{\mathbf{u}}}(\underline{x}_0)\big) \mid \underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e} \text{ and } \underline{\tilde{\mathbf{u}}}(\underline{x}_0) \in \bar{\mathcal{U}}_N(\underline{x}_0)\big\}, \tag{8.2.9}$$

in which the set of admissible warm-start control sequences is now defined by:

$$\bar{\mathcal{U}}_N(\underline{x}_0) := \{\underline{\mathbf{u}} \in \bigcup_{i=1}^{N} \mathcal{U}_N^\mathrm{e}(\underline{x}_0, i) \mid J_N(\underline{x}_0, \underline{\mathbf{u}}) \leq F(\underline{x}_0), \text{ if } \underline{x}_0 \in \mathbb{X}_\mathrm{f}\}. \tag{8.2.10}$$

These restrictions do not impact the theoretical derivations summarized in Section 7.1 since $\bar{\mathcal{U}}_N(\underline{x}_0) \subseteq \tilde{\mathcal{U}}_N(\underline{x}_0)$ and $\mathbb{X}_\mathrm{f} \subseteq \bar{\mathcal{X}}_N^\mathrm{e} \subseteq \mathcal{X}_N$. Thus, $\bar{\mathcal{Z}}_N \subseteq \mathcal{Z}_N$ holds with $\underline{0}_{p+mN} \in \bar{\mathcal{Z}}_N$. From Lemma 8.2.1 it can already be derived that the input set $\bigcup_{i=1}^N \mathcal{U}_N^\mathrm{e}(\underline{x}_0^+, i)$ always contains the control sequences resulting from $\underline{\Omega}_\kappa(\underline{x}_0)$ or $\underline{\Omega}_\mathrm{sta}(\underline{x}_0)$ if $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$. Recall that the Lyapunov inequalities in (7.1.9) hold for all $\big(\underline{x}_0, \underline{\tilde{\mathbf{u}}}(\underline{x}_0)\big)$ in the larger set $\mathcal{Z}_N$ and rely, in particular, on the availability of $\underline{\Omega}_\kappa(\underline{x}_0)$ if $\underline{x}_0 \in \mathbb{X}_\mathrm{f}$ ([All+17, Prop. 10, Prop. 22, Proof Thm. 14]). In contrast to Assumption 7.2.1, the following assumption explicitly ensures that there exists an initial extended state in which the control sequence is subject to extreme input move-blocking.

Assumption 8.2.1: *Admissible initial move-blocked warm-start.* At time instant $n = 0$, there exists an admissible warm-start $\underline{\tilde{\mathbf{u}}}(\underline{x}_0) \in \bar{\mathcal{U}}_N(\underline{x}_0)$ for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$.

This technical assumption is necessary since OCP (8.2.6) already assumes an admissible warm-start at time instant $n = 0$. However, since the optimal solution to OCP (8.2.6) is guaranteed to produce lower or at least equal costs than every available warm-start, the first closed-loop iteration can be solved (offline) to optimality and the optimal solution can then be used to replace the initial warm-start. Finally, taking all previous steps together, the following closed-loop properties can be stated for SFEMPC.

Proposition 8.2.2: *SFEMPC without fallback level.* Suppose Assumptions 3.1.1-3.3.2 and 8.2.1 hold. Let the suboptimal solutions to OCP (8.2.6) with $\bar{\mathbb{P}} := \mathbb{U}$ be defined by $\underline{\mathbf{u}}^\dagger(\underline{x}_0) := \underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_0, \underline{u}_\mathrm{s}^\dagger(\underline{x}_0), N_1^\dagger(\underline{x}_0)\big)$ for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$. Then, the origin is asymptotically stable for the closed-loop system (7.1.7) in the positive invariant set $\bar{\mathcal{Z}}_N$.

Proof. Assumption 8.2.1 provides the first admissible warm-start $\underline{\tilde{\mathbf{u}}}(\underline{x}_0)$ at closed-loop time $n = 0$ for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$. From Lemma 8.2.1 it follows that the optimizer can improve and thus start its optimization routine exactly at the manually generated warm-starts according to (7.1.5) by adjusting the horizon partition. Therefore, in the nominal case, if $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$, it follows that $\underline{x}_0^+ \in \bar{\mathcal{X}}_N^\mathrm{e}$ and $\underline{\tilde{\mathbf{u}}}(\underline{x}_0^+) = \underline{\Omega}\big(\underline{x}_0, \underline{\tilde{\mathbf{u}}}(\underline{x}_0)\big) \in \bar{\mathcal{U}}_N(\underline{x}_0) \subseteq \bigcup_{i=1}^N \mathcal{U}_N^\mathrm{e}(\underline{x}_0, i)$ hold after applying the suboptimal control vector $\underline{u}^\dagger(0, \underline{x}_0)$ for closed-loop control. Consequently, it follows that $\underline{\vartheta}^+ \in \bar{\mathcal{Z}}_N$. Asymptotic stability follows from [All+17, Prop. 13 and Proof of Thm. 14] with $\mathcal{Z}_N := \bar{\mathcal{Z}}_N$ and includes Assumptions 3.1.1-3.3.2 and the stabilizing properties of the manually generated warm-starts. □

Notice that the theoretical closed-loop properties in Proposition 8.2.2 do not depend on the optimal horizon partitioning. Thus, iterating over all possible horizon lengths is no longer necessary in the worst-case. However, mixed-integer programming is nevertheless computationally demanding. The next section proposes a viable strategy for systematically seeking suboptimal horizon lengths for a finite set of control vectors. Recall that the motivation is to dispense with external optimization and solver libraries.

Numerical Realization

The following Algorithm 8.1 represents an efficient search strategy for suboptimal horizon partitions in OCP (8.2.6) with $\bar{\mathbb{P}} := \tilde{\mathbb{A}}(\underline{x}_0, n)$ for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^{\mathrm{e}}$ and all $n \in \mathbb{N}_0$. Let a suboptimal solution tuple to OCP (8.2.6) with $\bar{\mathbb{P}} := \tilde{\mathbb{A}}(\underline{x}_0, n)$ be denoted by $\big(\underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0, n), N_1^{\dagger}(\underline{x}_0, n), N_2^{\dagger}(\underline{x}_0, n)\big)$. On the basis of this solution tuple, a suboptimal control sequence is given by:

$$\underline{\mathbf{u}}^{\dagger}(\underline{x}_0) := \underline{\mathbf{u}}_{\mathrm{e}}\big(\underline{x}_0, \underline{u}_{\mathrm{s}}^{\dagger}(\underline{x}_0, n), N_1^{\dagger}(\underline{x}_0, n)\big), \ \forall\, \underline{x}_0 \in \bar{\mathcal{X}}_N^{\mathrm{e}}, \ \forall\, n \in \mathbb{N}_0. \tag{8.2.11}$$

Recall that in suboptimal MPC according to [Pan+11; All+17], closed-loop stability properties are derived from the worst-case evolution of the difference inclusion (7.1.9), where the time-invariant cost function $J_N(\cdot)$ serves as the Lyapunov function. Since the suboptimal control sequence $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0)$ is not unique anyway, the left side of (8.2.11) dispenses with the explicit dependence on the closed-loop time n, which results from including time-varying finite control sets. The initial state as an input argument, however, links the suboptimal solution $\underline{\mathbf{u}}^{\dagger}(\underline{x}_0)$ to the evolution of the warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0)$. In the suboptimal case, the temporal evolution of the final control set in (4.5.4) is based on the suboptimal control vector selected by the optimizer with $\underline{u}^{\circ}(n) := \underline{u}^{\dagger}(0, \underline{x}_0)$ for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^{\mathrm{e}}$ and all $n \in \mathbb{N}_0$.
Algorithm 8.1 requires three loops to determine a suboptimal solution. The first loop in program line 5 represents a simple exhaustive search, which iterates over the number of elements in the finite control set $\tilde{\mathbb{A}}(\underline{x}_0, n)$. The second loop in program line 9 increases the horizon length N_1 incrementally in line 35 until either the roll-out of variable length violates state constraints (line 13), the local control law $\tilde{\underline{\kappa}}(\cdot)$ becomes admissible for the nonlinear system (lines 26-27), or the prediction reaches the terminal set $\mathbb{X}_{\mathrm{f}}$ at constant control with $N_1 = N$ (line 28). If the local controller neither violates state nor input constraints for $N - N_1$ steps (line 18) and $\underline{\varphi}\big(N, \underline{x}_0, \underline{\mathbf{u}}_{\mathrm{e}}(\underline{x}_0, \underline{u}_{\mathrm{s}}, N_1)\big) \in \mathbb{X}_{\mathrm{f}}$, then this implies that $\underline{\varphi}\big(N_1, \underline{x}_0, \underline{\mathbf{u}}_{\mathrm{e}}(\underline{x}_0, \underline{u}_{\mathrm{s}}, N_1)\big) \in \mathbb{X}_{\tilde{\kappa}}(N - N_1)$. Hence, if the local controller can take over steering the nonlinear system into the terminal set $\mathbb{X}_{\mathrm{f}}$, the algorithm terminates the variable roll-out of the current control candidate on the first part of the horizon (line 33). Otherwise, Algorithm 8.1 terminates the second loop (lines 22 and 24) and increases the first horizon length by one step (line 35). If state constraints violations occur during the first roll-out phase (line 13) or the second population phase (line 22), the proposed algorithm either discards the current control candidate $\underline{u}_{\mathrm{s}}$ or implements a penalty function according to Section 5.3. The theoretical runtime complexity of Algorithm 8.1 is $\mathcal{O}\big(c \cdot N^2\big)$. However, in practice, the runtime complexity can be approximated by $\mathcal{O}\big(c \cdot N \cdot \log(N)\big)$ since the non-admissibility of the local controller for $k_2 \ll N$ is often detectable after a few steps (lines 22 and 24).

Algorithm 8.1.: Procedure to search for a suboptimal solution to OCP (8.2.6)

```
 1: procedure FindSuboptimalSolution($\underline{x}_\mu(n), \tilde{\mathbb{A}}(\underline{x}_\mu(n), n)$)
 2:     $\underline{K}, \underline{P} \leftarrow$ Optional: Solve Riccati/Lyapunov equation online    ▷ See Sec. 3.3
 3:     $\mathcal{S} \leftarrow \emptyset$    ▷ Initialize set of admissible solution tuples
 4:     $\underline{x}_0 \leftarrow \underline{x}_\mu(n)$    ▷ Set initial state
 5:     for all candidates $\underline{u}_\mathrm{s} \in \tilde{\mathbb{A}}(\underline{x}_0, n)$ do    ▷ Parallelizable
 6:         $k_1 \leftarrow 0$
 7:         $\underline{x}(k_1) \leftarrow \underline{x}_0$
 8:         $J_1 \leftarrow 0$
 9:         while $k_1 < N$ do    ▷ **Variable roll-out**: Simulate first section
10:             $\underline{u}(k_1) \leftarrow \underline{u}_\mathrm{s}$
11:             $\underline{x}(k_1+1) \leftarrow \underline{f}(\underline{x}(k_1), \underline{u}(k_1))$    ▷ See (3.1.1)
12:             if $\underline{\mathcal{G}}(\underline{x}(k_1+1)) \not\preceq \underline{0}_r$ then    ▷ Check if $\underline{x}(k_1+1) \notin \mathbb{X}$, see (3.2.2)
13:                 break    ▷ Optional: Penalty function, see Sec. 5.3
14:             else
15:                 $J_1 \leftarrow J_1 + \ell(\underline{x}(k_1), \underline{u}(k_1))$    ▷ See (3.1.3)
16:                 $k_2 \leftarrow k_1 + 1$
17:                 $J_2 \leftarrow 0$
18:             while $k_2 < N$ do    ▷ **Variable horizon partitioning**: Populate second section
19:                 $\underline{u}(k_2) \leftarrow \tilde{\underline{\kappa}}(\underline{x}(k_2))$
20:                 $\underline{x}(k_2+1) \leftarrow \underline{f}(\underline{x}(k_2), \underline{u}(k_2))$    ▷ See (3.1.1)
21:                 if $\underline{\mathcal{G}}(\underline{x}(k_2+1)) \not\preceq \underline{0}$ then    ▷ Check if $\underline{x}(k_2+1) \notin \mathbb{X}$, see (3.2.2)
22:                     break    ▷ Optional: Penalty function, see Sec. 5.3
23:                 else if $\underline{G}_\mathrm{u}\underline{u}(k_2) \not\preceq \underline{h}_\mathrm{u}$ then    ▷ Check if $\underline{u}(k_2) \notin \mathbb{U}$, only if $\tilde{\underline{\kappa}}(\cdot) := \underline{\kappa}(\cdot)$
24:                     break
25:                 else
26:                     $J_2 \leftarrow J_2 + \ell(\underline{x}(k_2), \underline{u}(k_2))$    ▷ See (3.1.3)
27:                     $k_2 \leftarrow k_2 + 1$
28:             if $k_2 = N$ and $\underline{\mathcal{F}}(\underline{x}(N)) \preceq \underline{0}$ then    ▷ Check if $\underline{x}(N) \in \mathbb{X}_\mathrm{f}$, see (3.2.7)
29:                 $J \leftarrow J_1 + J_2 + F(\underline{x}(N))$    ▷ See (3.1.3)
30:                 $N_1 \leftarrow k_1 + 1$
31:                 $N_2 \leftarrow N - (k_1 + 1)$
32:                 $\mathcal{S} \leftarrow \{\mathcal{S}, (J, \underline{u}_\mathrm{s}, N_1, N_2)\}$    ▷ Append admissible tuple
33:                 break    ▷ **Stop variable roll-out**
34:             else
35:                 $k_1 \leftarrow k_1 + 1$
36:     $(J_N(\cdot), \underline{u}_\mathrm{s}^\dagger(\underline{x}_0, n), N_1^\dagger(\underline{x}_0, n), N_2^\dagger(\underline{x}_0, n)) \leftarrow$ Select least-cost tuple from $\mathcal{S}$
37:     $\tilde{\mathbb{A}}\Big(\underline{f}(\underline{x}_0, \underline{u}_\mathrm{s}^\dagger(\underline{x}_0)), n+1\Big) \leftarrow$ Adaptive discretization    ▷ See (4.5.4) and (4.5.6)
38:     return $(\underline{u}_\mathrm{s}^\dagger(\underline{x}_0, n), N_1^\dagger(\underline{x}_0, n), N_2^\dagger(\underline{x}_0, n)), \underline{\mathbf{u}}_\mathrm{e}(\underline{x}_0, \underline{u}_\mathrm{s}^\dagger(\underline{x}_0, n), N_1^\dagger(\underline{x}_0, n)), \tilde{\mathbb{A}}(\underline{x}_0^+, n+1)$
```

Similar to Algorithm 4.1, if all admissible solutions are stored in the memory, the analysis of the space complexity results in $\mathcal{O}(c \cdot N)$. However, if only saving the states $\underline{x}(k)$ and $\underline{x}(k+1)$ in combination with the current best tuple $(\underline{u}_\mathrm{s}, N_1, N_2)$, the space complexity reduces to $\mathcal{O}(1)$. Algorithm 8.1 is highly parallelizable since different threads can evaluate different control candidates (line 5). This algorithm is straightforward to implement, exhibits a deterministic run time limit and addresses industrial applications in particular. If the optimizer can evaluate all control candidates $\underline{u}_\mathrm{s} \in \tilde{\mathbb{A}}(\underline{x}_0, n)$ for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$ and all $n \in \mathbb{N}_0$, then Algorithm 8.1 guarantees to find and evaluate the warm-starts according to (7.1.5). For this purpose the technical Assumption 8.2.1 requires a slight modification.

Assumption 8.2.2: *Admissible initial move-blocked warm-start with minimum first horizon length.* At time instant $n = 0$, there exists an admissible tuple $(\underline{u}_\mathrm{s}^0, N_1^0)$ with $\underline{u}_\mathrm{s}^0 \in \mathbb{A}(0)$ such that $\underline{\mathbf{u}}_\mathrm{e}(\underline{x}_0, \underline{u}_\mathrm{s}^0, N_1^0) \in \bar{\mathcal{U}}_N(\underline{x}_0)$ holds for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$. The horizon length N_1^0 is of minimum length.

Similar to Assumption 4.5.1, Assumption 8.2.2 allows to derive closed-loop properties in the uncountable set $\bar{\mathcal{Z}}_N$, though it does not include finite control sets in its formulation. In addition, Assumption 8.2.2 ensures that the initial solution generated by Algorithm 8.1 produces lower or at least equal costs compared to the initial warm-start $\tilde{\underline{\mathbf{u}}}(\underline{x}_0)$. Practical SFEMPC, which rests upon Algorithm 8.1, exhibits the following closed-loop properties.

Corollary 8.2.1: *SFEMPC with finite control sets and exhaustive search.* Suppose Assumptions 3.1.1-3.3.2 and 8.2.2 hold. Assume that the suboptimal control sequences are generated based on (8.2.11) and the solution tuples $\big(\underline{u}_\mathrm{s}^\dagger(\underline{x}_0, n), N_1^\dagger(\underline{x}_0, n), N_2^\dagger(\underline{x}_0, n)\big)$ generated by Algorithm 8.1 for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$ and all $n \in \mathbb{N}_0$. Then, the origin is asymptotically stable for the closed-loop system (7.1.7) in the positive invariant set $\bar{\mathcal{Z}}_N$.

Proof. Case I: $n = 0$ and $\underline{x}_0 \notin \mathbb{X}_\mathrm{f}$. Algorithm 8.1 terminates the roll-out of the selected control candidate $\underline{u}_\mathrm{s}$ if $\underline{\varphi}\big(N_1, \underline{x}_0, \underline{\mathbf{u}}_\mathrm{e}(\underline{x}_0, \underline{u}_\mathrm{s}, N_1)\big) \in \mathbb{X}_{\tilde{\kappa}}(N - N_1)$. By Assumption 8.2.2, $\underline{u}_\mathrm{s}^0 \in \tilde{\mathbb{A}}(\underline{x}_0, 0)$ and the corresponding horizon length N_1^0 is of minimal length. Therefore, Algorithm 8.1 reproduces exactly the initial warm-start from Assumption 8.2.2.
Case II: $n = 0$ and $\underline{x}_0 \in \mathbb{X}_\mathrm{f}$. Algorithm 8.1 stops the variable roll-out on the first horizon section after one step, providing $N_1^\dagger(\underline{x}_0, 0) = 1$ and $N_2^\dagger(\underline{x}_0, 0) = N - 1$. The finite control set $\tilde{\mathbb{A}}(\underline{x}_0)$ contains $\underline{u}_\mathrm{s} = \underline{\kappa}(\underline{x}_0)$ by (4.5.6) and $\underline{u}_\mathrm{s}^0$ by Assumption 8.2.2. Thus, Algorithm 8.1 exactly reproduces the initial warm-start from Assumption 8.2.2 since $\underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_0, \tilde{\underline{\kappa}}(\underline{x}_0), 1\big) \in \bar{\mathcal{U}}_N(\underline{x}_0)$ and $\underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_0, \underline{u}_\mathrm{s}^0, 1\big) \in \bar{\mathcal{U}}_N(\underline{x}_0)$.
Case III: $n > 0$ and $\underline{x}_0 \notin \mathbb{X}_\mathrm{f}$. By design, the finite control set $\tilde{\mathbb{A}}\big(\underline{x}_\mu(n), n\big)$ contains the previously applied closed-loop control vector $\underline{u}_\mathrm{s}^\dagger\big(\underline{x}_\mu(n-1), n-1\big)$. Algorithm 8.1 evaluates $\underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_\mu(n), \underline{u}_\mathrm{s}^\dagger\big(\underline{x}_\mu(n-1), n-1\big), N_1^\dagger\big(\underline{x}_\mu(n-1), n-1\big) - 1\big) = \underline{\Omega}_\mathrm{sta}\big(\underline{x}_\mu(n-1), \underline{\mathbf{u}}^\dagger\big(\underline{x}_\mu(n-1)\big)\big) \in \bar{\mathcal{U}}_N\big(\underline{x}_\mu(n)\big)$.
Case IV: $n > 0$ and $\underline{x}_0 \in \mathbb{X}_\mathrm{f}$. The proof of this case is analogous to Case II. The optimizer evaluates the control sequence $\underline{\Omega}_\kappa(\underline{x}_0) = \underline{\mathbf{u}}_\mathrm{e}\big(\underline{x}_0, \tilde{\underline{\kappa}}(\underline{x}_0), 1\big) \in \bar{\mathcal{U}}_N(\underline{x}_0)$.
This proof only shows that the optimizer at least reproduce the warm-starts according to (7.1.5) for all $\underline{x}_0 \in \bar{\mathcal{X}}_N^\mathrm{e}$ and all $n \in \mathbb{N}_0$. Therefore, if $\underline{\vartheta} \in \bar{\mathcal{Z}}_N$, it follows that $\underline{\vartheta}^+ \in \bar{\mathcal{Z}}_N$. Asymptotic stability then directly results from [All+17, Prop. 13, Thm. 14]. □

Remark 8.2.1: *Automatic generation of warm-starts by the optimizer.* Recall that the extended state $\underline{\vartheta} = \big(\underline{x}_0, \tilde{\underline{\mathbf{u}}}(\underline{x}_0)\big)$ is a theoretical extension to ensure closed-loop properties if the optimizer cannot find the global optimum in finite time, which usually applies for non-convex optimization problems under real-time constraints [Pan+11; All+17]. Hence, Definition 3.3.1 does not apply for real applications. However, with a conventional (smooth) optimizer, the optimization variables can be initialized with the values of the manually generated warm-start controls sequence. Therefore, the optimizer starts its numerical routine exactly at the already stabilizing warm-start. However, if the conventional optimizer would start its optimization routine always with optimization variables initialized to zero, it is challenging, if not impossible, to estimate an upper bound on the optimization time which the optimizer requires to reproduce the stabilizing warm-start. With finite control sets and exhaustive search, the upper limit for evaluating all control candidates is deterministic. Since Algorithm 8.1 is configured to self-generate, evaluate, and improve stabilizing warm-start sequences in the nominal case, it allows to dispense with the manual generation of warm-starts.

Remark 8.2.2: *Inherent robustness with Algorithm 8.1.* Algorithm 8.1 does not ensure inherent robustness as suboptimal MPC with $\mathbb{X} = X$ and $\mathbb{X}_\mathrm{f} := \mathrm{lev}_\pi F$ with $\pi > 0$. The derived robustness margins in [All+17] rely, in particular, on the appended local control step at the end of the open-loop prediction. Alan et al. [All+17] analyze both in how far the appended local control step drives the prediction further into the interior of the terminal set $\mathrm{lev}_\pi F$ and how large the resulting distance to the boundary of the terminal set is. Hence, the preparation of the warm-start control sequence at time instant n allows to handle small perturbations at the next closed-loop time instant $n+1$. Since Algorithm 8.1 does not buffer the manually generated warm-starts, it cannot reproduce exactly these warm-starts automatically in case of perturbations such as measurement errors and model uncertainty. To guarantee inherent robustness margins like in [All+17], Algorithm 8.1 needs to be, at least, extended such that it compares some buffered warm-start, which was manually generated according to (7.1.5), with its best solution in terms of costs after it terminates the first loop in line 5. However, the analysis of inherent robustness properties is beyond the scope of this dissertation.

8.3. Example Continued: Van der Pol Oscillator

Figure 8.1 again continues on the Van der Pol oscillator and the stability analysis in Section 7.4. The benchmark system is configured as described in Section 7.4. The local controller is defined by $\tilde{\underline{\kappa}}(\underline{x}) := \underline{\Gamma}(\underline{K}\underline{x})$, where $\underline{K}$ involves the solution to the Riccati equation (3.3.19). The left plot shows the feasible sets resulting from different configurations of single degree of freedom OCPs and compares them with the feasible set $\mathcal{X}_N$ resulting from the full degree of freedom OCP in (3.1.6) with $N = 80$. As a reminder, the feasible sets are approximated by evaluating a uniform grid of initial states with 201×201 grid points placed in the constrained state space $\mathbb{X}$. To present almost smooth boundaries, the pure compact hulls of all feasible sets are enlarged in the direction of its convex boundaries while not distorting the polygon area above 5 %. The control samples for the single degree of freedom OCPs (4.1.4), (8.1.5), and (8.2.6) are taken from the finite control set $\mathbb{D} \subset \mathbb{U}$ with $c = 101$. As already mentioned in Section 7.4, the combination of a terminal set constraint with a single degree of freedom in control is too restrictive for practical applications (see small set $\mathcal{X}_N^\mathrm{s}$).

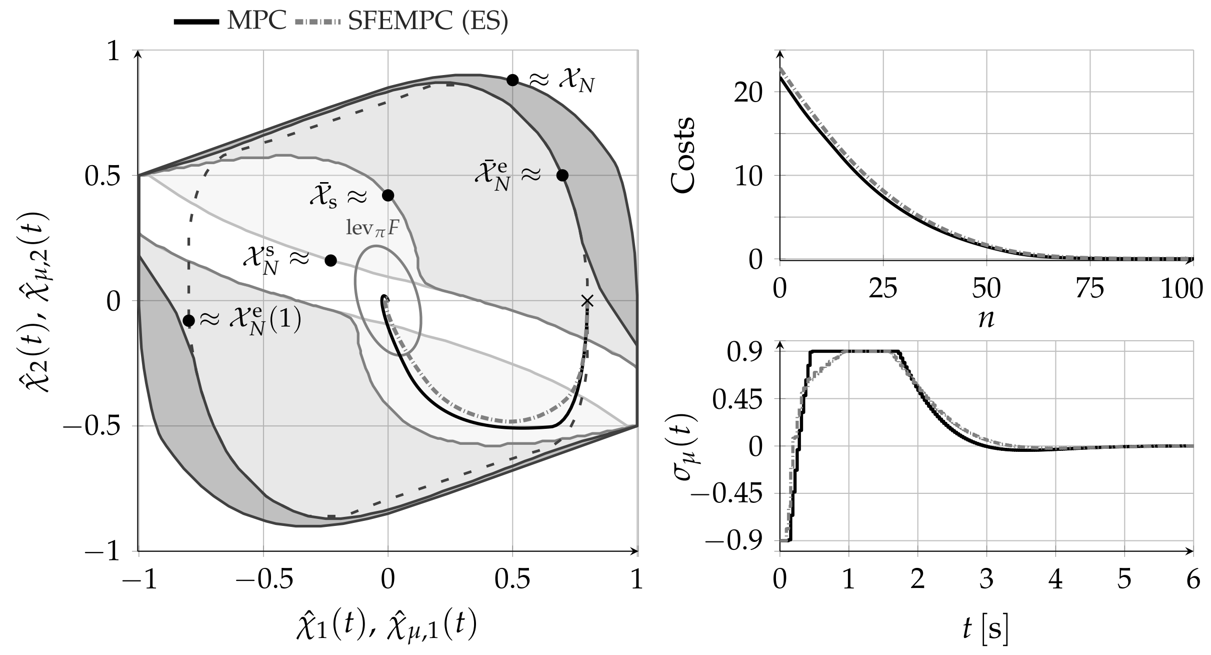


Figure 8.1.: Enlarging the restrictive feasible set $\mathcal{X}^{\mathrm{s}}_N$ (with SFMPC) by relaxing the OCP formulation: $\bar{\mathcal{X}}_{\mathrm{s}}$ (with SFVMPC), $\bar{\mathcal{X}}^{\mathrm{e}}_N$ (with SFEMPC), $\mathcal{X}_N$ (with conventional MPC). Left: Phase-portrait with two illustrative closed-loop state trajectories. Right, top: Descent of costs in the sense of Lyapunov. Right, bottom: Closed-loop control trajectories over time.

Though the variable horizon formulation presented in Section 8.1 guarantees stabilizing closed-loop control properties, the enlargement of the feasible set with $\mathcal{X}^{\mathrm{s}}_N \subset \bar{\mathcal{X}}_{\mathrm{s}}$ and $N_{\max} = 80$ might still be not attractive for practical applications. However, the extended horizon formulation introduced in Section 8.2 results in a feasible set area $\bar{\mathcal{X}}^{\mathrm{e}}_N$ that is at least fairly competitive to the feasible set area of $\mathcal{X}_N$. Recall that the set $\bar{\mathcal{X}}^{\mathrm{e}}_N$ is estimated with Algorithm (8.1) without including smooth derivative-based optimization. The enlargement of the feasible set from $\mathcal{X}^{\mathrm{s}}_N$ to $\bar{\mathcal{X}}^{\mathrm{e}}_N$ mainly stems from the fact that the region of attraction $\mathbb{X}_{\tilde{\kappa}}(N-1)$ covers a large part of the feasible set $\mathcal{X}_N$. For orientation, Figure 8.1 includes the feasible set resulting from OCP (8.2.6) for the case when the first horizon length is fixed to $N_1 = 1$ (see gray dashed polygon). In this case, $\mathbb{X}_{\tilde{\kappa}}(N-1) \subseteq \mathcal{X}^{\mathrm{e}}_N(1)$ applies. The two illustrative closed-loop state trajectories in the left phase portrait of Figure 8.1 all start at $\underline{x}_{\mu,0} = (0.8, 0)^\mathsf{T}$. SFEMPC and conventional MPC show strong similarity in the phase portrait. The bottom time plot on the right confirms this observation, but attests that the significantly higher number of degrees of freedom in control in case of MPC allows better exploitation of the system's control bounds. However, for all control approaches, the top plot of Figure 8.1 shows cost descent properties in the sense of Lyapunov. The following quantitative results enhance the evaluations in Table 7.1. SFEMPC with $M = 1$, $N = 80$, and $c = 21$ reaches a closed-loop performance index of $\bar{J}_{\mathrm{cl}} = -\mathbf{2.18}\,\%$. Thus, for the chosen benchmark system, SFEMPC is comparable to suboptimal MBMPC with $M = 8$ in terms of closed-loop control performance, though it only builds on a single degree of freedom. In the MATLAB simulation environment, the statistical execution time evaluation over 100 optimization runs to OCP (8.2.6) with $c = 21$ yields a normalized median value

of $\bar{t}_\mathrm{m} = \mathbf{91.29}\,\%$ with $t_\mathrm{m} = 48.86\,\mathrm{ms}$. The runtime performance is thus comparable to that of (derivative-based) suboptimal MBMPC with $M = 4$. The exhaustive search strategy and the second population phase in Algorithm 8.1 do not increase the computational effort unreasonably. The statistical evaluation in the C environment results in the values $t_\mathrm{m} = 80.9\,\mu\mathrm{s}$ and $t_{95} = 83.7\,\mu\mathrm{s}$. These values are comparable to those resulting from conventional MPC and the SQP method with a horizon between $N = 6$ and $N = 12$ (see Table 5.1).
The chosen benchmark system and its configuration allow to visualize the enlargement of the feasible set resulting from including the local controller on the second part of the horizon. However, the advantages that arise from choosing $N_1 > 1$ are not yet clearly visible in Figure 8.1. Without tight state box-constraints, Figure 8.1 only reveals that there is a small potential for enlarging the feasible set $\mathcal{X}^\mathrm{e}_N(1)$ to $\bar{\mathcal{X}}^\mathrm{e}_N$. The next example elaborates on the ability of SFEMPC to handle state constraints with $N_1 \geq 1$.

8.4. Example: Simplified Nonlinear Valve Model

This section relates to the application-oriented part of this work in Chapter 6 and aims at placing the theoretical derivations of this chapter in the context of model predictive low-level control. Since nominal offline simulation does not require an observer, offset compensation, and is not subject to real-time constraints, a simplified nonlinear model is used to approximate the real valve behavior. The following benchmark system is represented by a series connection of a PT1 element with the differential equation $0.5\,\dot{y}(t) + y(t) = \sigma(t)$ and the Duffing oscillator with the differential equation $\ddot{\varphi}_\mathrm{v}(t) + 0.2\,\dot{\varphi}_\mathrm{v}(t) - \varphi_\mathrm{v}(t) + \varphi_\mathrm{v}^3(t) = y(t)$ (see, e.g., [Wig03]). The PT1 element is intended to be a simplified representation of the electromagnetic phenomena inside a solenoid. It roughly approximates the dynamic conversion from input voltage to solenoid output force. On the other hand, the Duffing oscillator is supposed to approximate the nonlinear motion of some magnetic armature and piston as a second-order system. The resulting mathematical state space model of third degree with $\underline{\chi}(t) = \big(\chi_1(t) = \varphi_\mathrm{v}(t), \chi_2(t) = \dot{\varphi}_\mathrm{v}(t), \chi_3(t) = y(t)\big)^\mathsf{T}$ is given by:

$$\begin{pmatrix} \dot{\chi}_1(t) \\ \dot{\chi}_2(t) \\ \dot{\chi}_3(t) \end{pmatrix} = \begin{pmatrix} \dot{\varphi}_\mathrm{v}(t) \\ y(t) - 0.2\,\dot{\varphi}_\mathrm{v}(t) + \varphi_\mathrm{v}(t) - \varphi_\mathrm{v}^3(t) \\ -2\,y(t) + 2\,\sigma(t) \end{pmatrix}. \tag{8.4.1}$$

The input of the system $\sigma(t)$ is assumed to be piecewise constant on the fixed time grids $t_{k+1} = t_k + \Delta t_\mathrm{s}, k \in \mathbb{N}^{[0,N-1]}, t_0 = 0\,\mathrm{s}$ (open-loop) and $t_{n+1} = t_n + \Delta t_\mathrm{s}, n \in \mathbb{N}_0, t_0 = 0\,\mathrm{s}$ (closed-loop). Again, discretization is based on the Runge-Kutta method of the fourth order with $\Delta t_\mathrm{s} = 2^{-4}\,\mathrm{s}$. The constraint sets are defined by $\mathbb{X} = \{\underline{x} \in X \,|\, (-1, -v_\mathrm{lim}, -1)^\mathsf{T} \preceq \underline{x} \preceq (1, v_\mathrm{lim}, 1)^\mathsf{T}\}$ with $v_\mathrm{lim} = 0.25$ and $\mathbb{U} = \{u \in U \,|\, |u| \leq 1\}$. The control task is motivated by a common position control task and is therefore translated into a point-to-point state-space control to pre-defined (unstable) steady states $(\underline{x}_\mathrm{f}, \underline{u}_\mathrm{f})$. Here, the coordinate transformations $\underline{\breve{x}}(k) := \underline{x}(k) - \underline{x}_\mathrm{f}$ and $\underline{\breve{u}}(k) := \underline{u}(k) - \underline{u}_\mathrm{f}$ are used along with the cost-functions $\ell\big(\underline{\breve{x}}(k), \underline{\breve{u}}(k)\big) = \|\,\underline{\breve{x}}(k)\|^2_{\underline{\underline{Q}}} + \|\,\underline{\breve{u}}(k)\|^2_{\underline{\underline{R}}}$ and $F\big(\underline{\breve{x}}(N)\big) = \|\,\underline{\breve{x}}(N)\|^2_{\underline{\underline{P}}}$ with $\underline{\underline{Q}} = \mathrm{diag}(1, 0.1, 0.1)$ and $\underline{\underline{R}} = 0.1$. The matrix $\underline{\underline{P}}$

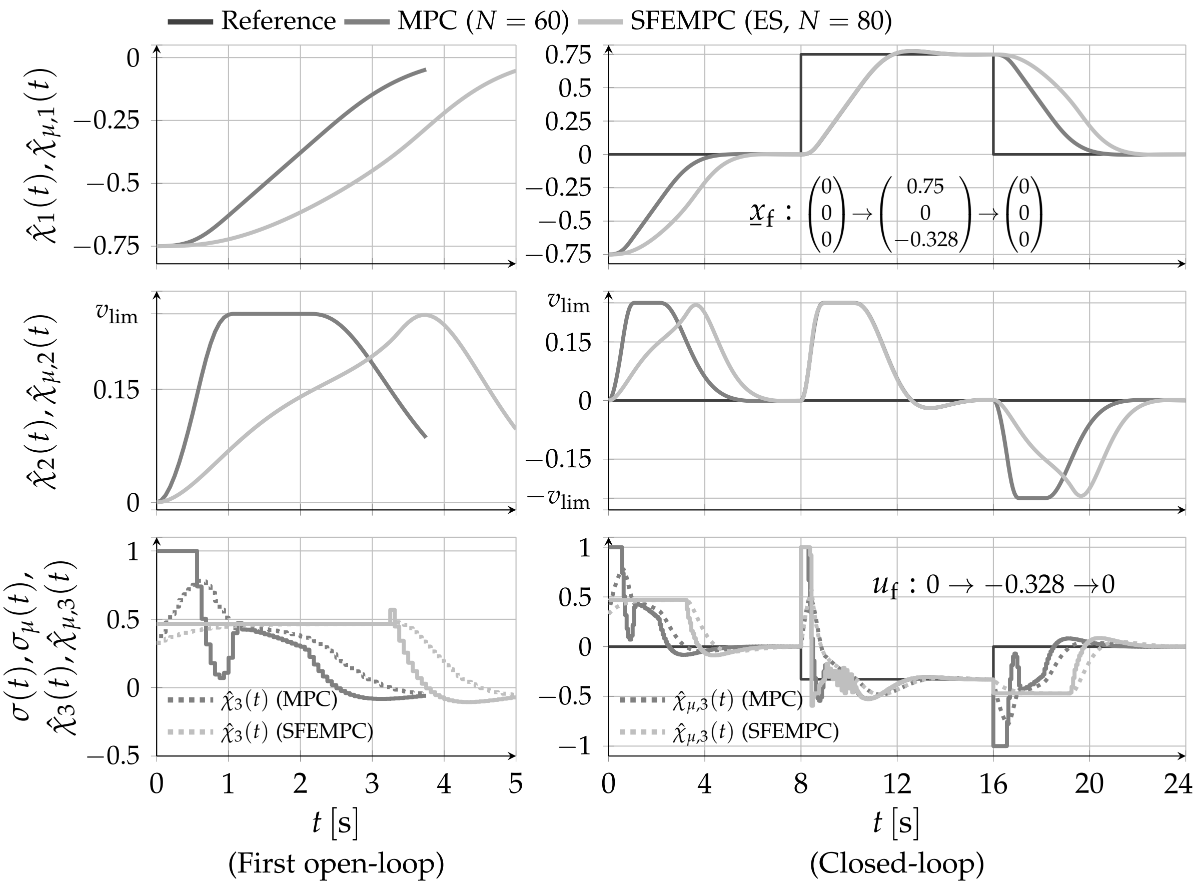


Figure 8.2.: Open- and closed-loop control of the simplified model of a directional control valve. The closed-loop system is initialized by $\underline{x}_{\mu,0} = (-0.75, 0, 0.328)^\intercal$.

is the solution to the Riccati equation (3.3.18) with $\underline{A} = \left(\partial \underline{f}(\underline{x}, \underline{u})/\partial \underline{x}\right)|_{(\underline{x}_\text{f}, \underline{u}_\text{f})}$ and $\underline{B} = \left(\partial \underline{f}(\underline{x}, \underline{u})/\partial \underline{u}\right)|_{(\underline{x}_\text{f}, \underline{u}_\text{f})}$. Linearization at a given steady state and solving the Riccati equation are performed during closed-loop control. Recall, if there exists a solution to the Riccati equation (3.3.18), the linearized system $\underline{\breve{x}}(k+1) = \underline{A}\,\underline{\breve{x}}(k) + \underline{B}\,\underline{\breve{u}}(k)$ is stabilizable at the origin by the local control law $\underline{\kappa}\big(\underline{\breve{x}}(k)\big) = \underline{K}\,\underline{\breve{x}}(k)$, where $\underline{K}$ is defined by (3.3.19). According to Remark 3.3.3, without a proof, the terminal set $\mathbb{X}_\text{f}$ is assumed to have at least an interior $\text{lev}_\pi F \subset \mathbb{X}_\text{f}$ with $\pi = 0.05$ and $\rho = 1.001$ for all steady states under consideration. As discussed in Remark 3.3.4, the time dependence of the steady state $(\underline{x}_\text{f}, \underline{u}_\text{f})$ is omitted. The local control law that is applied to the nonlinear system is further subject to saturation with $\underline{\tilde{\kappa}}\big(\underline{x}(k)\big) := \underline{\Gamma}\big(\underline{u}_\text{f} + \underline{K}\,\underline{\breve{x}}(k)\big)$.

The numerical experiment in Figure 8.2 consists of three individual point-to-point closed-loop control phases in order to emulate conventional position control. On the left side of Figure 8.2, the plots visualize the initial open-loop solution at closed-loop time $t_n = 0\,\text{s}$ for MPC and SFEMPC. To not violate the constraint $v_\text{lim} = 0.25$ and at the same time reaching the terminal set $\text{lev}_\pi F$ with $N = 80$, Algorithm 8.1 with $c = 31$ divides the horizon in $N_1^\dagger(\underline{x}_0, n) = 52$ and $N_2^\dagger(\underline{x}_0, n) = 28$. When starting at $\underline{x}_0 = \underline{x}_{\mu,0} = (-0.75, 0, 0.328)^\intercal$, choosing $N_1 = 1$ is not sufficient since the recursive application of the local control law $\underline{\tilde{\kappa}}(\cdot)$ violates state constraints. Thus, the optimizer selects a moderate system excitation of $\underline{u}_\text{s}^\dagger(\underline{x}_0) \approx 0.47$ on the first part of the predic-

Table 8.1.: Evaluation of closed-loop control performances and computation times for MPC, SFMPC, and SFEMPC applied to the simplified valve model. Abbreviations: Recursive Elimination (RE), Full Discretization (FD), Exhaustive Search (ES).

			MPC		SFMPC (ES)	SFEMPC (ES)
Environment	Solver		FD/RE	$N = 60$	$N = 5, c = 31, \mathbb{X}_{\mathrm{f}} = \mathbb{X}$	$N = 80, c = 31$
MATLAB	IPOPT/ES	$\bar{J}_{\mathrm{cl}}$	FD	0.00 %	$-$**0.31** %	$-$**56.16** %
MATLAB	IPOPT/ES	$\bar{t}_{\mathrm{m}}$	FD	0.00 %	**99.1** %	**62.45** %
C	SQP/ES	t_{m}	RE	693.77 ms	121.3 μs	6.56 ms
C	SQP/ES	t_{95}	RE	698.86 ms	125.75 μs	6.78 ms

tion (variable roll-out) in order to steer the system through the constrained set $\mathbb{X}$ into the subset $\mathbb{X}_{\tilde{\kappa}}\big(N_2^{\dagger}(\underline{x}_0, n)\big)$. Here, the second part of the horizon partition keeps the system inside the constrained state space $\mathbb{X}$ while further binding the last predicted state to the terminal set $\mathrm{lev}_{\pi}F$. Obviously, the solution to the full degree of freedom OCP (3.1.6) in case of MPC outperforms the solution resulting from Algorithm 8.1 since it allows multiple control interventions outside the region of attraction of the local controller. As a logical consequence, MPC only requires a shorter horizon length of $N = 60$ to satisfy the terminal constraint. Thus, the closed-loop control performance with SFEMPC requires a longer time compared to MPC to drive the closed-loop system in the neighborhood of the dedicated steady state. However, during the second point-to-point closed-loop control phase, Algorithm 8.1 returns $N_1^{\dagger}(\underline{x}_0, n) = 1$ and $N_2^{\dagger}(\underline{x}_0, n) = 79$ since at time $t_n = 8\,\mathrm{s}$, $\underline{x}_0 \in \mathbb{X}_{\tilde{\kappa}}(79)$ applies. Here, the closed-loop control performances of MPC and SFEMPC are fairly similar. The third phase proceeds analogously to the first phase.

Table 8.1 summarizes the quantitative analysis of the first closed-loop control phase in Figure 8.2. The first line shows that MPC is almost equal to an optimal full degree of freedom point-to-point solution with $N = 1000$ since $\bar{J}_{\mathrm{cl}} \approx 0\,\%$ after $l = 1000$ closed-loop steps. The basic SFMPC approach with a short horizon of $N = 5$ and with the terminal constraint disabled is similar in terms of closed-loop control performance with $\bar{J}_{\mathrm{cl}} = -0.31\,\%$. Though basic SFMPC shows recursive feasibility for this selected numerical experiment, there do not exist theoretical closed-loop guarantees. It shall be noted that SFMPC is only able to stabilize the origin for the selected benchmark system if including the solution to the Riccati equation $\underline{P}$ in the terminal cost function $F(\cdot)$, though the terminal state vector $\underline{\varphi}\big(N, \underline{x}_0, \underline{\mathbf{u}}_{\mathrm{s}}^*(\underline{x}_0)\big)$ is not inside the terminal set (cf. Ch. 5). In contrast, SFEMPC ensures stabilizing properties in the nominal case by design. However, this closed-loop guarantees can only be obtained at the expense of closed-loop control performance since $\bar{J}_{\mathrm{cl}} = -56.16\,\%$ after $l = 1000$ closed-loop steps. The statistical evaluation of the optimization effort is based on 100 cold-started optimization runs at time $t_n = 0\,\mathrm{s}$ with $t_{\mathrm{m,ref}} = 814.49\,\mathrm{ms}$ (OCP (3.1.6) with full discretization). The runtime performance of SFEMPC ranges between the runtime performance of MPC and SFMPC. Though, the increase of computational effort is not negligible compared to SFMPC. While the results gained in MATLAB are suited for a relative rating of execution time levels of the individual approaches, the

C environment discloses that a smooth optimization framework requires a bunch of specific development steps as rigorous NLP structure exploitation in order to scale in the same ratio as SFMPC and SFEMPC when generating C-code.

8.5. Discussion

This chapter reveals that the key ingredient of stabilizing SFMPC is the variable horizon length on which the control candidate vector is kept constant. The first Section 8.1 introduces the variable horizon formulation and combines it with common stabilizing terminal conditions to provide stabilizing closed-loop guarantees. However, the resulting feasible set from which the nonlinear system can be steered to some pre-defined terminal set suffers from the extreme input move-blocking approach. Enlarging the feasible set is achieved by incorporating the local control law during prediction. In contrast to the formulations in [ZA98] and [Mag+01], in Section 8.2, the time step on the horizon from which the local control law is applied is subject to optimization. Therefore, this extended approach recovers the key ingredient in stabilizing SFMPC. Since in total SFEMPC implements a receding horizon formulation and rests upon common stabilizing terminal conditions, it can be integrated into the suboptimal MPC framework presented in [Pan+11; All+17; Raw+20]. Finally, Algorithm 8.1 addresses the resulting mixed-integer OCP and is designed for searching efficiently for suboptimal solutions. In addition, this tailored exhaustive search algorithm satisfies the fundamental objectives of this dissertation. Namely, its implementation is straightforward and the resulting computational effort is comparatively low. The applicability of SFEMPC depends, in particular, on the size of the region of attraction of the local controller. If the linearized system is only a valid approximation in the immediate vicinity of the selected steady state, it is unlikely that the region of attraction of the local controller exceeds beyond the border of the control invariant terminal set. Here, the configuration of the first section applies and the resulting control approach is too restrictive for practical applications. However, if the local region of attraction is relatively large, SFEMPC is suitable for both guaranteeing stabilizing closed-loop properties and strictly satisfying input and state constraints, at least for the nominal configuration. Though conventional MPC outperforms SFEMPC in terms of open- and closed-loop control performance for the simplified valve model, SFEMPC accomplishes the point-to-point control task while comprising only a few lines of code.
The smaller $\pi > 0$ in $\operatorname{lev}_{\pi} F$, the more likely it is that this level set represents some interior of all emerging terminal sets (see Rem. 3.3.4). However, the smaller the parameter π, the larger the length N_2 and thus the total horizon length N have to be. The increase in N_2 does not increase the computational effort of SFEMPC considerably. Rather, the greatest computational cost of SFEMPC is associated with transferring the open-loop system into a subset of the local controller's domain of attraction. In contrast, with MPC, the optimization time would increase significantly with an increasing number of optimization parameters. Thus, by including a stabilizing local control law for prediction, SFEMPC exhibits similar numerical properties as the finite-tail cost approaches in [Mag+01; KA21].

9 Conclusion and Outlook

Mechatronic subsystems contribute significantly to the overall performance of complex assemblies such as construction and manufacturing machines. However, the transformation of an input signal into a mechanical motion of a mechatronic subsystem is subject to several parasitic and nonlinear effects as, for example, eddy currents and friction. Therefore, low-level control takes on a key role in realizing smooth operation and fulfilling high performance demands. The closed-loop position control of a directional control valve, as an example for a real industrial application, shows that a nonlinear proportional-integral control scheme achieves a high closed-loop control performance at the expense of a high parameter complexity. A high number of free controller parameters, however, complicates the controller design and the subsequent manual adjustment of the system behavior. If there would exist an ideal optimization algorithm that certainly finds the global minimizer of non-convex and non-smooth optimization problems in a negligible amount of time, MPC clearly qualifies for replacing conventional low-level controllers. Since MPC incorporates explicit system knowledge in the form of a dynamic state space model, parameter complexity decreases significantly on the controller side. In addition, usually MPC implements interpretable and weighted objective function terms that enable intuitive controller design. Apart from these two beneficial properties, closed-loop control performance benefits from the recurring solution of an OCP. However, realizing a numerically robust optimization algorithm that satisfies real-time constraints is a major challenge in MPC.

This dissertation focuses on the development of a model predictive low-level control scheme combining the following features: Straightforward implementation without dependence on external optimization libraries, low computational effort, intuitive controller design, and stabilizing closed-loop properties. Extreme input move-blocking, as a key ingredient of the novel control concept for systems with small input and small to mid-sized state dimensions, reduces the degrees of freedom in control to a single degree and establishes the basis for adaptive discretization of the input domain. This discretization procedure is converted into a time evolution of finite control sets, which are then embedded into the different OCP formulations. With the proposed simplified OCP formulations, the simple exhaustive search algorithm qualifies for solving the resulting OCPs and resembles a nearly ideal optimizer.

The basic SFMPC integrates with stabilizing terminal ingredients in the framework of conventional receding horizon MPC. Three special OCP configurations allow to derive stabilizing closed-loop guarantees despite of input move-blocking. Since basic SFMPC reproduces, under mild assumptions, MPC and the well-known LQR, the discretization error of the control domain is analyzed systematically, especially isolated from other effects. This analysis allows to formalize necessary control vectors that have to be included in the time-varying finite control sets to ensure stabilizing closed-loop properties. Since the monotonicity property of the optimal value function does not hold with input move-blocking, the horizon length needs to be chosen by considering the trade-off between constraint compliance and the decreasing closed-loop performance with an increasing horizon length.

Extensive numerical investigations evaluate the closed-loop control performance of basic SFMPC for constrained nonlinear systems, using only approximate infinite horizon costs as a general direction guide, and highlight the beneficial property of handling non-smooth penalty functions to enable robust real-time operation in the presence of short-term state constraint violations. This dissertation presents a step-by-step MPC realization guideline for the real-time closed-loop control of a directional control valve and a servo-motor, including the implementation of offset and computation time compensation techniques. For both mechatronic systems, SFMPC covers a wide operating range, establishes a high control performance, offers real-time operation in the upper kilohertz sampling range, and enables an intuitive controller design. The implementation is based on a few lines of code and dispenses with external optimization libraries.

When great emphasis is given to theoretical closed-loop stability by design, SFMPC with an extended horizon formulation (SFEMPC) can substitute basic SFMPC. This extension combines several important stabilizing ingredients. In preparation to the development of SFEMPC, this dissertation presents a novel formulation of stabilizing MBMPC that integrates into the well-known suboptimal MPC framework. The key idea is to treat move-blocking as a source of suboptimality and to include stabilizing warm-start control sequences as a fallback level either through simple buffering or by introducing auxiliary optimization parameters. This dissertation shows that SFEMPC automatically generates and evaluates stabilizing warm-starts in the nominal case. SFEMPC divides the fixed horizon into two parts, from which only the first part is subject to extreme input move-blocking. The second part of the prediction is completed by recursively applying some stabilizing local control law to the nonlinear system. Since the switching point represents an optimization variable, SFEMPC ensures stabilizing closed-loop properties by following common stability derivations of shifting and truncating the (sub)optimal control sequence of the previous closed-loop control step and appending some local control law step at the end of the sequence. Finally, the basic SFMPC algorithm is enhanced by a variable roll-out procedure of control candidates that searches efficiently for suboptimal solutions to the resulting nonlinear mixed-integer OCP. The resulting SFEMPC algorithm with finite control sets and exhaustive search satisfies all requirements placed on the low-level controllers in this dissertation.

This work offers several OCP formulations for derivative-free model predictive low-level control. The resulting low-level controllers are suitable for replacing existing conventional controllers in cascaded control loops to enhance closed-loop control performance and to enable intuitive controller design. The field of application is not limited to mechatronic systems with electromagnetic actuators, but also extends to a general class of nonlinear systems. However, to minimize the combinatorial complexity, SFMPC and SFEMPC are, in particular, suitable for single-input systems.

Outlook

The promising results of real-time valve control in Chapter 6 are based on an experimental setup optimized for prototyping, where an external amplifier directly drives the solenoid valve (see Fig. 6.3). Since this setup is only suitable for a proof of concept of the new control approach, it is of practical relevance to refine SFMPC for the embedded valve hardware. For this step, a nonlinear model is required that approximates the valve behavior between the input of the current controller $i_{\mathrm{ref}}(t)$ and the measured stroke $y_{\mathrm{m}}(t)$. Future research in this field could focus on the optimization of relevant hardware design parameters, similar to [Mak+18a; Mak+18b; Mak+18c], such that a nonlinear model can achieve high model accuracy during the identification process.
A major challenge of SFMPC, and also of SFEMPC, with finite control sets is the handling of combinatorial complexity in the case of systems with multiple inputs. Recall that the number of control candidates grows exponentially with the input dimensions. The numerical evaluations in this dissertation reveal that the straightforward exhaustive search algorithm is only a suitable optimizer for finite control sets with a low cardinality. To solve problems with a higher combinatorial complexity, alternative search algorithms have to be investigated and developed. Here, it is advisable to analyze the structure and behavior of the system a priori to control to identify control combinations along all input dimensions that are unlikely to be chosen by the optimizer. Then, all control vectors can be sorted into a list according to their selection probability. The first entry in this list could represent the control candidate that generates the stabilizing warm-start. Then, the optimizer can be configured to process the list from top to bottom starting with the control vector that is most likely to result in a low cost function value and terminating as it approaches the maximum run time.
If the degrees of freedom in control of SFMPC are to be increased, note that the combinatorial complexity of MBMPC with finite control sets also increases exponentially with the number of switching points. To further reproduce the switching points of the previous open-loop control solution, the switching points also have to be discretized, as proposed in [Hom+18]. Assuming that a suitable combinatorial search algorithm can be developed in the future, which can handle systematically the combinatorial explosion of optimal control problems with finite control sets for at least a few degrees of freedom in control, the following combination is reasonable. The sampling of switching points as shown in [Hom+18] can be combined with the variable roll-out procedure in Algorithm 8.1 on the last blocked control interval. This combination would ensure recursive feasibility and asymptotic stability of MBMPC with time-varying finite control sets, without the need to manually generate stabilizing warm-starts as in SDNMPC according to [BL17].

A

Supplemental Details on Discretization, Optimization, and Implementation

This appendix chapter provides more details on the discretization of system dynamics and optimal control problems, and summarizes the fundamentals of constrained optimization. In addition, this chapter presents details on the implementation of a proprietary and generic control software framework.

A.1. Discretization of Optimal Control Problems

This section serves as a visual supplement to the explanations on recursive elimination [GP17] and multiple shooting [BP84; Raw+20; GP17] in Section 3.2. Each discretization approach results in an individual structure or sparsity pattern of derivative information matrices, which are required for solving constrained optimization problems. The Jacobian matrix of the constraint functions is used to visualize the resulting structure/sparsity patterns with state and input box-constraints.
Figure A.1 shows the basic principle beyond recursive elimination, using a one-dimensional system with $p = 1$ and $m = 1$. Each time the optimizer provides new values for the optimization vector $\underline{\mathbf{z}} := \underline{\mathbf{u}} = \big(u(0), u(1), u(2), u(3), u(4), u(5)\big)$, an underlying instance solves the recursive dynamics equation (3.1.2) with $x(k) := \underline{\varphi}(k, \underline{x}_0, \underline{\mathbf{z}})$. The dashed lines show the piecewise constant parameterization of the control trajectory and a possible evolution of the state trajectory for the corresponding sampled data system according to Section 3.4. Note that these dashed graphs do not exist in the discrete-time representation since there are no intermediate time points between k and $k+1$. However, assume that the sampled data formulation is defined on the time axis $t \in [kT_\mathrm{s}, (k+1)T_\mathrm{s})$ with $k \in \mathbb{N}^{[0,N-1]}$ and $T_\mathrm{s} = 1\,\mathrm{s}$. The corresponding Jacobian of the constraint functions is denoted by $\underline{D}(\underline{\mathbf{z}})$ and its structure is shown in (A.1.1). This Jacobian matrix has a lower triangular structure for all $\underline{z} \in \mathbb{R}^\omega$ since the variation of a control $u(l)$ at time step $l \in \mathbb{N}^{[0,N-1]}$ only affects the constraints that follow the control in prediction time on the interval $\mathbb{N}^{[l+1,N]}$. This Jacobian matrix excludes lower and upper bounds on the optimization variables. The filled circles represent elements that need to be recomputed each time the optimizer offers new values for the optimization vector $\underline{\mathbf{z}}$, while the other elements represent structural zeros. The example in

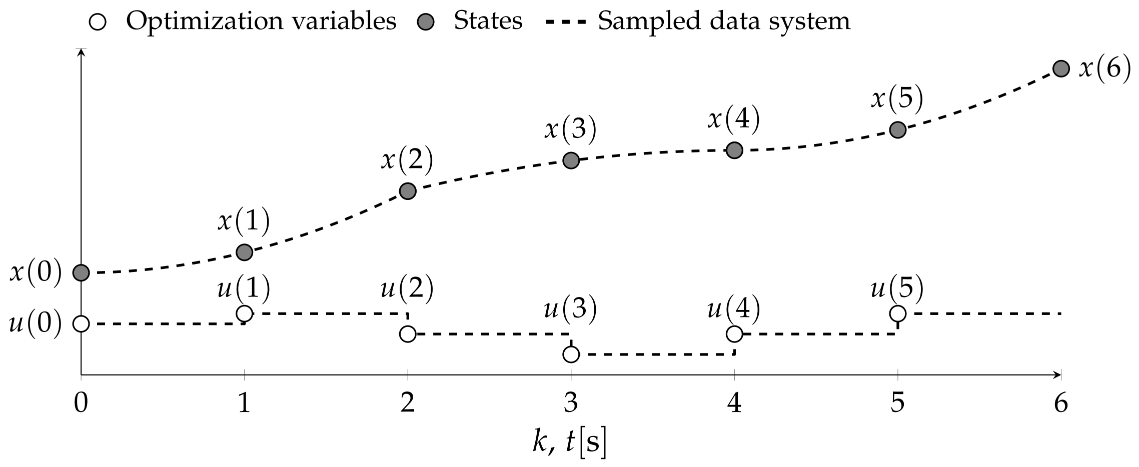


Figure A.1.: Example of recursive elimination for a one-dimensional system with a horizon of $N = 6$. In the discrete-time representation, only the highlighted points exist.

Figure A.1 results in a relative number of structural non-zeros of the Jacobian matrix $\underline{D}(\underline{\mathbf{z}})$ in (A.1.1) of $(49/78) \cdot 100\,\% \approx 62.82\,\%$.

$$\underline{D}(\cdot) = \begin{array}{c} \begin{array}{cccccc} u(0) & u(1) & u(2) & u(3) & u(4) & u(5) \end{array} \\ \left(\begin{array}{cccccc} \bullet & 0 & 0 & 0 & 0 & 0 \\ \bullet & 0 & 0 & 0 & 0 & 0 \\ \bullet & \bullet & 0 & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & \bullet & 0 & 0 \\ \bullet & \bullet & \bullet & \bullet & 0 & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{array}\right) \end{array} \begin{array}{l} \\ x(1) \leq x_{\max} \\ x(1) \geq x_{\min} \\ x(2) \leq x_{\max} \\ x(2) \geq x_{\min} \\ x(3) \leq x_{\max} \\ x(3) \geq x_{\min} \\ x(4) \leq x_{\max} \\ x(4) \geq x_{\min} \\ x(5) \leq x_{\max} \\ x(5) \geq x_{\min} \\ x(6) \leq x_{\max} \\ x(6) \geq x_{\min} \\ x(6) \in \operatorname{lev}_\pi F \end{array} \tag{A.1.1}$$

Figure A.2 illustrates multiple shooting for the case when the optimizer has not yet fulfilled the convergence criterion. The prediction horizon is divided into $T = 3$ individual intervals with the shooting nodes s_0, s_1, and s_2, while the last shooting node s_3 is only introduced for fast compliance with the terminal constraint $s_3 \in \operatorname{lev}_\pi F$. Every shooting interval has the same number of controls with $L_0 = L_1 = L_2 = 2$. Hence, it follows that $I(0) = 0$, $I(1) = 2$, $I(2) = 4$, and $I(3) = 6$. The optimization vector is defined by $\underline{\mathbf{z}} := \big(s_0, s_1, s_2, s_3, u(0), u(1), u(2), u(3), u(4), u(5)\big)$. In Figure A.2, the continuity constraints, $s_1 = x(2)$, $s_2 = x(4)$, and $s_3 = x(6)$, are not yet satisfied.
If rearranging the optimization vector, the Jacobian matrix of the constraint functions $\underline{D}(\underline{\mathbf{z}})$ in (A.1.2) exhibits a banded sparsity structure for all $\underline{\mathbf{z}} \in \mathbb{R}^\omega$. The example in Figure A.2 results in a relative level of structural non-zero elements of the Jacobian matrix in (A.1.2) of $(44/170) \cdot 100\,\% \approx 25.89\,\%$. Though the Jacobian matrix

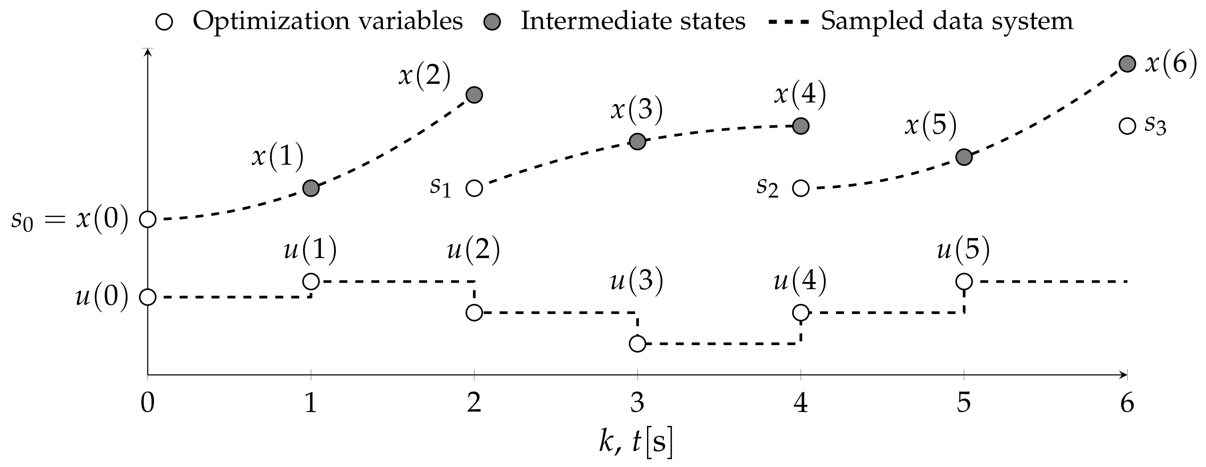


Figure A.2.: Example of multiple-shooting for a one-dimensional system with a horizon of $N = 6$ and $T = 3$ shooting intervals. This figure illustrates the non-converged case, which implies that the (continuity) constraints are not yet satisfied by the optimizer.

$\underline{D}(\underline{\mathbf{z}}) \in \mathbb{R}^{17\times 10}$ is larger with multiple shooting compared to recursive elimination with $\underline{D}(\underline{\mathbf{z}}) \in \mathbb{R}^{13\times 6}$, the number of structural non-zero elements, namely 44 compared to 49, is similar to each other. Recall that full discretization would further substitute the intermediate states by introducing further shooting nodes.

$$\underline{D}(\cdot) = \begin{array}{c} \begin{array}{cccccccccc} s_0 & u(0) & u(1) & s_1 & u(2) & u(3) & s_2 & u(4) & u(5) & s_3 \end{array} \\ \left(\begin{array}{cccccccccc}
\bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bullet & \bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bullet & \bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bullet & \bullet & \bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bullet & \bullet & \bullet & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bullet & \bullet & \bullet & \bullet & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \bullet & \bullet & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \bullet & \bullet & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \bullet & \bullet & \bullet & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \bullet & \bullet & \bullet & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \bullet & \bullet & \bullet & \bullet & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \bullet & \bullet & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \bullet & \bullet & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \bullet & \bullet & \bullet & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \bullet & \bullet & \bullet & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \bullet & \bullet & \bullet & \bullet \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bullet
\end{array}\right) \end{array}
\begin{array}{c} \\ s_0 = x_0 \\ x(1) \leq x_{\max} \\ x(1) \geq x_{\min} \\ x(2) \leq x_{\max} \\ x(2) \geq x_{\min} \\ s_1 = x(2) \\ x(3) \leq x_{\max} \\ x(3) \geq x_{\min} \\ x(4) \leq x_{\max} \\ x(4) \geq x_{\min} \\ s_2 = x(4) \\ x(5) \leq x_{\max} \\ x(5) \geq x_{\min} \\ x(6) \leq x_{\max} \\ x(6) \geq x_{\min} \\ s_3 = x(6) \\ s_3 \in \mathrm{lev}_\pi F \end{array} \tag{A.1.2}$$

Different instances of an optimization framework can exploit structure and sparsity information and therefore improve optimization runtime performance. These instances include, inter alia, sparse finite differences, condensing, sparse linear algebra, and sparse (linear) solvers.

For the sake of completeness, the NLP formulation for full discretization is presented below. Here, the optimization vector is defined by $\underline{\mathbf{z}} := \big(\underline{x}(0), \underline{x}(1), ..., \underline{x}(N), \underline{u}(0), \underline{u}(1), ..., \underline{u}(N-1)\big) \in \mathbb{R}^{p(N+1)+mN}$. Since with full discretization there are no intermediate states on the individual shooting intervals, NLP (3.2.9) simplifies to (similar to [GP17, Sec. 12.1]):

$$\min_{\underline{\mathbf{z}} \in \mathbb{R}^{p(N+1)+mN}} \psi(\underline{\mathbf{z}}) := \min_{\underline{\mathbf{z}} \in \mathbb{R}^{p(N+1)+mN}} \sum_{k=0}^{N-1} \ell\big(\underline{x}(k), \underline{u}(k)\big) + F\big(\underline{x}(N)\big) \quad \text{(A.1.3)}$$

subject to

$$\begin{aligned}
\underline{x}(0) &= \underline{x}_0, && \text{(initialization)}\\
\underline{G}_\mathrm{u}\underline{u}(k) &\preceq \underline{h}_\mathrm{u},\ \forall k \in \mathbb{N}^{[0,N-1]}, && (\underline{u}(k) \in \mathbb{U})\\
\underline{\mathcal{G}}\big(\underline{x}(k)\big) &\preceq \underline{0},\ \forall k \in \mathbb{N}^{[0,N-1]}, && (\underline{x}(k) \in \mathbb{X})\\
\underline{\varphi}\big(1, \underline{x}(k), \underline{u}(k)\big) - \underline{x}(k+1) &= \underline{0},\ \forall k \in \mathbb{N}^{[0,N-1]}, && \text{(continuity)}\\
\underline{\mathcal{F}}\big(\underline{x}(N)\big) &\preceq \underline{0}. && (\underline{x}(N) \in \mathbb{X}_\mathrm{f})
\end{aligned}$$

A.2. Fundamentals of Constrained Optimization

This section summarizes only the basics of constrained optimization to support the content and statements of this dissertation. The summary is based on [NW06; WB06]. Consider the NLP from (3.2.1), which is given again as:

$$\min_{\underline{z} \in \mathbb{R}^\omega} \psi(\underline{z}) \ \text{ subject to } \begin{cases} h_i(\underline{z}) = 0,\ \forall i \in \mathcal{E} \subset \mathbb{N}_0, \\ g_i(\underline{z}) \leq 0,\ \forall i \in \mathcal{I} \subset \mathbb{N}_0. \end{cases} \quad \text{(A.2.1)}$$

Assume that $\psi : \mathbb{R}^\omega \mapsto \mathbb{R}_0^+$, $h_i, g_i : \mathbb{R}^\omega \mapsto \mathbb{R}$ represent continuously differentiable cost and constraint functions, respectively, with $\omega \in \mathbb{N}$.

Remark A.2.1: *General requirements for smooth optimization.* The theoretical derivations in the main part of the dissertation only require continuous cost and constraint functions. Assumptions 3.1.1 and 3.1.2, in particular, ensure that the individual OCPs have feasible solutions in compact sets (see, e.g., [Raw+20, Prop. 2.4]). Note that an ideal (global) optimizer does not have to necessarily determine derivative information. However, derivative-based numerical optimization has proven to be a suitable approach for high-dimensional optimization problems where optimization time is also an important criterion. Therefore, this appendix section demands that the cost and constraint functions are not only continuous but also continuously differentiable.

A minimizer $\underline{z}^*$ is a point in the search space at which there is no optimization direction along which the cost function can be further minimized without violating any constraints. Hence, the gradient of the cost function $\nabla\psi(\underline{z}^*)$ has to be colinear with the sum of the gradients of all active constraints. Let $\underline{\tilde{\lambda}}$ and $\underline{\tilde{\mu}}$ denote the vectors of Lagrange multipliers with the components $\tilde{\lambda}_i$ with $i \in \mathcal{E}$ and $\tilde{\mu}_i$ with $i \in \mathcal{I}$, respectively. The set of indices of all active inequality constraints at point $\underline{z}$ is denoted by $\mathcal{A}(\underline{z}) \subseteq \mathcal{I}$.

The Lagrange function is defined by:

$$\mathcal{L}(\underline{z}, \underline{\tilde{\lambda}}, \underline{\tilde{\mu}}) = \psi(\underline{z}) + \sum_{i \in \mathcal{E}} \tilde{\lambda}_i h_i(\underline{z}) + \sum_{i \in \mathcal{I}} \tilde{\mu}_i g_i(\underline{z}). \tag{A.2.2}$$

The gradient of the Lagrange function with respect to the primal variables $\underline{z}$ at point $(\underline{z}, \underline{\tilde{\lambda}}, \underline{\tilde{\mu}})$ results in:

$$\nabla_{\underline{z}} \mathcal{L}(\underline{z}, \underline{\tilde{\lambda}}, \underline{\tilde{\mu}}) = \nabla \psi(\underline{z}) + \sum_{i \in \mathcal{E}} \tilde{\lambda}_i \nabla h_i(\underline{z}) + \sum_{i \in \mathcal{I}} \tilde{\mu}_i \nabla g_i(\underline{z}). \tag{A.2.3}$$

The Karush-Kuhn-Tucker (KKT) conditions represent the first-order necessary conditions for optimization problems with equality and inequality constraints. The KKT conditions for a given tuple $(\underline{z}^*, \underline{\tilde{\lambda}}^*, \underline{\tilde{\mu}}^*)$ are defined by (e.g., [NW06, Thm. 12.1]):

$$\begin{aligned} \nabla_{\underline{z}} \mathcal{L}(\underline{z}^*, \underline{\tilde{\lambda}}^*, \underline{\tilde{\mu}}^*) &= \underline{0}, \\ h_i(\underline{z}^*) &= 0, \quad \forall\, i \in \mathcal{E} \subset \mathbb{N}_0, \\ g_i(\underline{z}^*) &\leq 0, \quad \forall\, i \in \mathcal{I} \subset \mathbb{N}_0, \\ \tilde{\mu}_i &\geq 0, \quad \forall\, i \in \mathcal{I} \subset \mathbb{N}_0, \\ \tilde{\mu}_i g_i(\underline{z}^*) &= 0, \quad \forall\, i \in \mathcal{I} \subset \mathbb{N}_0. \end{aligned} \tag{A.2.4}$$

If an inequality constraint is active with $g_i(\underline{z}^*) = 0$ and $i \in \mathcal{A}(\underline{z}^*)$, then $\tilde{\mu}_i \geq 0$ applies $\big(\tilde{\mu}_i > 0$ for strict complementarity [NW06, Def. 12.5]$\big)$. If an inequality constraint is inactive with $g_i(\underline{z}) < 0$, then $\tilde{\mu}_i = 0$ applies. In both cases, the complementary condition $\tilde{\mu}_i g_i(\underline{z}) = 0$ holds for all $i \in \mathcal{I}$. If a tuple $(\underline{z}^*, \underline{\tilde{\lambda}}^*, \underline{\tilde{\mu}}^*)$ satisfies the KKT conditions in (A.2.4) and the set of all active constraint gradients $\{\nabla h_i(\underline{z}^*), \forall i \in \mathcal{E}$, and $\nabla g_i(\underline{z}^*), \forall i \in \mathcal{A}(\underline{z}^*)\}$ is linearly independent, $\underline{z}^*$ represents a minimizer of NLP (A.2.1) (e.g., [NW06, Thm. 12.1]). The latter condition is known as the linear independent constraint qualification, which ensures that the linearization of the constraints at point $\underline{z}^*$ is a valid geometric representation of the feasible set in some close neighborhood of $\underline{z}^*$ (e.g., [NW06, Def. 12.4]). For second-order sufficient conditions refer, for example, to [NW06, Sec. 12.5]. Recall that if the optimization problem is convex (convex function defined on a convex set), there exist only one minimizer $(\underline{z}^*, \underline{\tilde{\lambda}}^*, \underline{\tilde{\mu}}^*)$ that satisfies the KKT conditions. This minimizer then also represents the global minimum. Moreover, if the constraints further satisfy the Slater condition, then the feasible set is non-empty, and this minimizer is guaranteed to exist [BV04].

Newton-KKT System

Assume for the moment that NLP (A.2.1) is subject only to equality constraints. Let $\underline{D}_\mathrm{h}(\underline{z}) := \mathrm{D}\, \underline{h}(\underline{z}) = \big(\nabla h_1(\underline{z}), \nabla h_2(\underline{z}), ..., \nabla h_{\max(\mathcal{E})}(\underline{z})\big)^\intercal$ be the Jacobian matrix of the equality constraint functions evaluated at point $\underline{z}$. The KKT conditions in (A.2.4) in vector representation are given by [NW06, Eq. (18.3)]:

$$\underline{h}_{\mathrm{KKT}}(\underline{z}, \underline{\tilde{\lambda}}) = \begin{pmatrix} \nabla \psi(\underline{z}) + \underline{D}_\mathrm{h}^\intercal(\underline{z}) \underline{\tilde{\lambda}} \\ \underline{h}(\underline{z}) \end{pmatrix} = \underline{0}. \tag{A.2.5}$$

The Jacobian matrix of (A.2.5) with respect to $\underline{z}$ and $\tilde{\underline{\lambda}}$ is given by [NW06, Eq. (18.4)]:

$$\mathrm{D}\,\underline{h}_{\mathrm{KKT}}(\underline{z}, \tilde{\underline{\lambda}}) = \begin{pmatrix} \nabla^2_{\underline{z}\underline{z}}\mathcal{L}(\underline{z}, \tilde{\underline{\lambda}}) & \underline{D}^{\mathsf{T}}_{\mathrm{h}}(\underline{z}) \\ \underline{D}_{\mathrm{h}}(\underline{z}) & \underline{0} \end{pmatrix}. \tag{A.2.6}$$

Here, $\nabla^2_{\underline{z}\underline{z}}\mathcal{L}(\underline{z}, \tilde{\underline{\lambda}})$ denotes the Hessian of the Lagrange function $\mathcal{L}(\cdot)$ with respect to $\underline{z}$. Finally, the Newton-KKT system at optimization step $l \in \mathbb{N}_0$ follows as [NW06, Eq. (18.6)]:

$$\begin{pmatrix} \nabla^2_{\underline{z}\underline{z}}\mathcal{L}(\underline{z}_l, \tilde{\underline{\lambda}}_l) & \underline{D}^{\mathsf{T}}_{\mathrm{h}}(\underline{z}_l) \\ \underline{D}_{\mathrm{h}}(\underline{z}_l) & \underline{0} \end{pmatrix} \begin{pmatrix} \Delta\underline{z} \\ \Delta\tilde{\underline{\lambda}} \end{pmatrix} = \begin{pmatrix} -\nabla\psi(\underline{z}_l) - \underline{D}^{\mathsf{T}}_{\mathrm{h}}(\underline{z}_l)\tilde{\underline{\lambda}}_l \\ -\underline{h}(\underline{z}_l) \end{pmatrix}. \tag{A.2.7}$$

If $\mathrm{D}\,\underline{h}_{\mathrm{KKT}}(\underline{z}_l, \tilde{\underline{\lambda}}_l)$ is non-singular and thus invertible, the Newton iteration can be realized as follows [NW06, Eq. (18.5)]:

$$\begin{pmatrix} \underline{z}_{l+1} \\ \tilde{\underline{\lambda}}_{l+1} \end{pmatrix} = \begin{pmatrix} \underline{z}_l \\ \tilde{\underline{\lambda}}_l \end{pmatrix} + \bar{\alpha}_l \begin{pmatrix} \Delta\underline{z} \\ \Delta\tilde{\underline{\lambda}} \end{pmatrix}. \tag{A.2.8}$$

The step widths $\bar{\alpha}_l \in (0, 1]$ for all $l \in \mathbb{N}_0$ can, for example, be determined by a backtracking line search approach as proposed in [NW06, Alg. 18.3]. The Newton iteration scheme in (A.2.8) represents a strategy to search iteratively for the roots of (A.2.5), which represent local optima of NLP (A.2.1) in the absence of inequality constraints.

The Newton-KKT system in (A.2.7) already indicates which derivative matrices are required for constrained optimization. In general, a Newton-type solver for conventional optimal control requires methods that determine, inter alia, the gradient of the objective function, the Jacobian of the constraint functions, and the Hessian of the Lagrange function. These vectors and matrices can either be determined numerically or, if possible, analytically.

Sequential-Quadratic-Programming

With SQP, the iterative solution process of the Newton-KKT system can be transformed into a sequential solution of quadratic programs (e.g., [NW06, Sec. 18.1]). Consider the following QP that includes the linearized representations of the inequality and equality constraints at a point $\underline{z}_l$ as follows [NW06, Eq. (18.11)]:

$$\begin{aligned} \min_{\Delta\underline{z} \in \mathbb{R}^{\omega}} \quad & \frac{1}{2}\Delta\underline{z}^{\mathsf{T}}\, \nabla^2_{\underline{z}\underline{z}}\mathcal{L}(\underline{z}_l, \tilde{\underline{\lambda}}_l, \tilde{\underline{\mu}}_l)\, \Delta\underline{z} + \nabla\psi^{\mathsf{T}}(\underline{z}_l)\Delta\underline{z} + \psi(\underline{z}_l) \\ \text{subject to} \quad & \nabla h_i^{\mathsf{T}}(\underline{z}_l)\Delta\underline{z} + h_i(\underline{z}_l) = 0, \quad \forall i \in \mathcal{E}, \\ & \nabla g_i^{\mathsf{T}}(\underline{z}_l)\Delta\underline{z} + g_i(\underline{z}_l) \leq 0, \quad \forall i \in \mathcal{I}. \end{aligned} \tag{A.2.9}$$

The solution to this QP is denoted by the tuple $(\Delta\underline{z}^*, \Delta\tilde{\underline{\lambda}}^*, \Delta\tilde{\underline{\mu}}^*)$ and can be used to generate a new iterate $(\underline{z}_{l+1}, \tilde{\underline{\lambda}}_{l+1}, \tilde{\underline{\mu}}_{l+1})^{\mathsf{T}}$ similar to (A.2.8) by applying a suitable line search strategy. If the underlying QP solver represents an active-set solver, such as the KWIK algorithm [SB94] in *mpcActiveSetSolver*$(\cdot)$ from MATLAB, the determined active set of QP (A.2.9) at iteration l is used to warm-start the active set of the NLP (A.2.1) at iteration $l + 1$. If this estimate of the active set is sufficient, the SQP algorithm exhibits the same convergence properties as the direct solution to the equality constrained Newton-KKT system in (A.2.7) and (A.2.8) [NW06, Sec. 18.1].

Interior Point Optimization for Nonlinear Programming

For interior point approaches, NLPs with inequality constraints are usually first transformed into NLPs with only equality constraints by introducing slack variables $\underline{\tilde{s}} = (\tilde{s}_1, \tilde{s}_2, ..., \tilde{s}_{\max(\mathcal{I})})^\intercal \in \mathbb{R}^{\max(\mathcal{I})}$ as follows (e.g., [NW06, Sec. 19.1]):

$$\min_{\underline{z} \in \mathbb{R}^{\omega},\ \underline{\tilde{s}} \in \mathbb{R}^{\max(\mathcal{I})}} \psi(\underline{z}) \ \text{ subject to } \begin{cases} h_i(\underline{z}) = 0, \ \forall\, i \in \mathcal{E} \subset \mathbb{N}_0, \\ g_i(\underline{z}) + \tilde{s}_i = 0, \ \forall\, i \in \mathcal{I} \subset \mathbb{N}_0, \\ \tilde{s}_i \geq 0, \ \forall\, i \in \mathcal{I} \subset \mathbb{N}_0. \end{cases} \tag{A.2.10}$$

For the sake of illustration, a more restrictive formulation of NLP (A.2.10) is given by (e.g., [WB06, Eq. (2)]):

$$\min_{\underline{z} \in \mathbb{R}^{\tilde{\omega}}} \psi(\underline{z}) \ \text{ subject to } \begin{cases} h_i(\underline{z}) = 0, \ \forall\, i \in \tilde{\mathcal{E}} \subset \mathbb{N}_0, \\ z_i \geq 0, \ \forall\, i \in \mathbb{N}^{[1,\tilde{\omega}]} \subset \mathbb{N}_0. \end{cases} \tag{A.2.11}$$

Here, the relations $\tilde{\omega} = \omega + \max(\mathcal{I})$ and $\mathcal{E} \subset \tilde{\mathcal{E}}$ hold. The elements of $\underline{z}$ are denoted by z_i for all $i \in \mathbb{N}^{[1,\tilde{\omega}]}$. Finally by introducing a logarithmic barrier penalty function with some $\bar{\beta} > 0$, the lower bounds in NLP (A.2.11) can be omitted (e.g., [WB06; NW06]):

$$\min_{\underline{z} \in \mathbb{R}^{\tilde{\omega}}} \left(\psi(\underline{z}) - \bar{\beta} \sum_{i=1}^{\tilde{\omega}} \log(z_i) \right) \ \text{ subject to } \ h_i(\underline{z}) = 0, \ \forall\, i \in \tilde{\mathcal{E}} \subset \mathbb{N}_0. \tag{A.2.12}$$

The barrier function term $-\log(z_i)$ tends to infinity as z_i tends to zero. This property implicitly ensures that $z_i \geq 0$ holds for all $i \in \mathbb{N}^{[1,\tilde{\omega}]}$. Let $\underline{D}_{\mathrm{h}}(\underline{z}) := \mathrm{D}\, \underline{h}(\underline{z}) = \left(\nabla h_1(\underline{z}), \nabla h_2(\underline{z}), ..., \nabla h_{\max(\tilde{\mathcal{E}})}(\underline{z})\right)^\intercal$ be the Jacobian matrix of the equality constraint functions evaluated at point $\underline{z}$. Now, the so-called system of primal-dual equations is given by (e.g., [WB06, Eq. (4)]):

$$\underline{h}_{\mathrm{pd}}(\underline{z}, \underline{\tilde{\lambda}}, \underline{\tilde{\mu}}) := \begin{pmatrix} \nabla\psi(\underline{z}) + \underline{D}_{\mathrm{h}}^\intercal(\underline{z})\underline{\tilde{\lambda}} - \underline{\tilde{\mu}} \\ \underline{h}(\underline{z}) \\ \mathrm{diag}(\underline{z})\, \mathrm{diag}\left(\underline{\tilde{\mu}}\right)\underline{1}_{\tilde{\omega}} - \bar{\beta}\, \underline{1}_{\tilde{\omega}} \end{pmatrix} = \underline{0}. \tag{A.2.13}$$

Here, $\underline{\tilde{\mu}}$ represents the vector of Lagrange multipliers corresponding to the lower bounds in NLP (A.2.11). Notice that with $\bar{\beta} = 0$, (A.2.13) represents the KKT conditions from (A.2.4) in vector form. The Jacobian matrix of (A.2.13) with respect to the optimization variables $\underline{z}$ and the Lagrange multipliers $\underline{\tilde{\lambda}}$ and $\underline{\tilde{\mu}}$ is given by:

$$\mathrm{D}\, \underline{h}_{\mathrm{pd}}(\underline{z}, \underline{\tilde{\lambda}}, \underline{\tilde{\mu}}) = \begin{pmatrix} \nabla^2_{\underline{z}\underline{z}}\mathcal{L}(\underline{z}, \underline{\tilde{\lambda}}, \underline{\tilde{\mu}}) & \underline{D}_{\mathrm{h}}^\intercal(\underline{z}) & \underline{I}_{\tilde{\omega}} \\ \underline{D}_{\mathrm{h}}(\underline{z}) & \underline{0} & \underline{0} \\ \mathrm{diag}\left(\underline{\tilde{\mu}}\right) & \underline{0} & \mathrm{diag}(\underline{z}) \end{pmatrix}. \tag{A.2.14}$$

To solve the barrier problem (A.2.13) for a given $\bar{\beta}_l > 0$ and optimization step $l \in \mathbb{N}_0$, again, the Newton method can be applied as follows (e.g., [WB06, Eq. (9)]):

$$\begin{pmatrix} \nabla^2_{\underline{z}\underline{z}}\mathcal{L}(\underline{z}_l, \underline{\tilde{\lambda}}_l, \underline{\tilde{\mu}}_l) & \underline{D}_{\mathrm{h}}^\intercal(\underline{z}_l) & -\underline{I}_{\tilde{\omega}} \\ \underline{D}_{\mathrm{h}} & \underline{0} & \underline{0} \\ \mathrm{diag}\left(\underline{\tilde{\mu}}_l\right) & \underline{0} & \mathrm{diag}(\underline{z}_l) \end{pmatrix} \begin{pmatrix} \Delta\underline{z} \\ \Delta\underline{\tilde{\lambda}} \\ \Delta\underline{\tilde{\mu}} \end{pmatrix} = - \begin{pmatrix} \nabla\psi(\underline{z}_l) + \underline{D}_{\mathrm{h}}^\intercal(\underline{z}_l)\underline{\tilde{\lambda}}_l - \underline{\tilde{\mu}}_l \\ \underline{h}(\underline{z}_l) \\ \mathrm{diag}(\underline{z}_l)\mathrm{diag}\left(\underline{\tilde{\mu}}_l\right)\underline{1}_{\tilde{\omega}} - \bar{\beta}_l\, \underline{1}_{\tilde{\omega}} \end{pmatrix} \tag{A.2.15}$$

After solving the linear system of equations in (A.2.15) for the unknown search directions $\Delta \underline{z}$, $\Delta \underline{\tilde{\lambda}}$, and $\Delta \underline{\tilde{\mu}}$, the next iterate $(\underline{z}_{l+1}, \tilde{\lambda}_{l+1}, \underline{\tilde{\mu}}_{l+1})^\mathsf{T}$ can be again derived from the step in (A.2.8) by applying some suitable line search strategy. Besides the step size $\bar{\alpha}_l$, the barrier weighting parameter $\bar{\beta}_l$ has to be adjusted iteratively such that $\bar{\beta}_l \to 0$ as the number of iterations l increases. Starting with $\bar{\beta}_0 > 0$ enforces (implicit) compliance with the lower bounds in (A.2.11), while converging to $\bar{\beta}_l = 0$ allows the iterates to converge to some KKT point of NLP (A.2.11) (e.g., [NW06; WB06]).

A.3. Proprietary Software Framework

For this dissertation, a generic software framework has been written in MATLAB to evaluate the developed control approaches (SFMPC, SFVMPC, SFEMPC, suboptimal MBMPC) and compare them systematically with the state-of-the-art in conventional MPC. Key objectives of the software development are the straightforward configuration of the benchmark problems, reproducible and automated generation of results, and the following extraction of these results. Therefore, the framework is fully based on a modular class hierarchy with the following base classes: Numerics (for sampled data systems), dynamics, constraints, objective, NLP, solver, controller, observer, buffer, control task. The software framework is designed for a discrete-time formulation, but the dynamics step method can invoke the solution to an underlying continuous-time system in the sense of a sampled data formulation. The sequential and simultaneous OCP discretization approaches (e.g., recursive elimination and multiple shooting) inherit from the base NLP class and provide an algorithmic definition of their structure/sparsity. This structure is then exploited in the application of sparse central finite differences and sparse solvers. Central finite differences use a step width of 10^{-8} to calculate derivatives. Different evaluation tasks, such as statistical runtime evaluations, closed-loop control, and estimation of feasible regions, inherit from the base control task class. For instantiating specific class objects, the user can call a graphical user interface (build with MATLAB App Designer) and select from a collection of objects. Finally, a parser/printer then composes the benchmark setup automatically and generates a callable function.

Another major objective of the software development is to create full compatibility for automatic C/C++ code generation using Matlab Coder or Simulink Coder. This requirement enables the acquisition of realistic execution time measurements and is an important prerequisite for real-time operation. A key feature of the software framework is its seamless integration into real-time environments such as Simulink Real-Time since most of the instantiated class objects do not depend on external libraries. The numerical procedure for solving the algebraic discrete-time Riccati equation (3.3.18) is adopted from [RK20] and rewritten in MATLAB, such that it can be used for the interpreted execution mode and the automatic code generation. The numerical solution to the Riccati equation (3.3.18) is based on the papers [BD93; BS72; Sim16].

Since the calculation of a Hessian matrix based on finite differences is time-consuming and sensitive to the chosen numerical step width, all NLP solvers used in this work each implement a tailored form of the BFGS scheme for iterative approximation of the Hessian of the Lagrange function $\underline{H}_l := \nabla^2_{\underline{z}\underline{z}}\mathcal{L}(\underline{z}_l, \tilde{\underline{\lambda}}_l, \tilde{\underline{\mu}}_l)$. The Hessian approximation rests upon the first-order derivative information $\underline{y}_l := \nabla_{\underline{z}}\mathcal{L}(\underline{z}_{l+1}, \tilde{\underline{\lambda}}_{l+1}, \tilde{\underline{\mu}}_{l+1}) - \mathcal{L}(\underline{z}_l, \tilde{\underline{\lambda}}_{l+1}, \tilde{\underline{\mu}}_{l+1})$ and the difference vector $\tilde{\underline{z}}_l := \underline{z}_{l+1} - \underline{z}_l$. The basic BFGS update rule for a Hessian matrix at the next iteration $\underline{H}_{l+1}$ is given by (e.g., [NW06, Eq. (18.16)]):

$$\underline{H}_{l+1} = \underline{H}_l - \frac{\underline{H}_l \tilde{\underline{z}}_l \tilde{\underline{z}}_l^\intercal \underline{H}_l}{\tilde{\underline{z}}_l^\intercal \underline{H}_l \tilde{\underline{z}}_l} + \frac{\underline{y}_l \underline{y}_l^\intercal}{\underline{y}_l^\intercal \tilde{\underline{z}}_l}. \tag{A.3.1}$$

The initial Hessian matrix $\underline{H}_0$ might follow, for example, from finite differences.

Proprietary SQP Implementation

As already mentioned in Chapter 5, the software implementation comprises a fully self-written SQP algorithm adopted from [Rös19, App. E.2.2.]. This SQP algorithm is based on the backtracking line search algorithm in [NW06, Alg. 18.3]. To ensure progress of the iterates $\underline{z}_{l+1}$ to some KKT point, thus reducing costs while minimizing constraint violations, an ℓ_1 exact penalty function serves as merit function (e.g., [Mor+12]), which is minimized during line-search to estimate a proper step width $\bar{\alpha}_l$. To include the external QP solver OSQP [Ste+20] into the code generation pipeline, which is a library free C implementation, the software framework implements a customized MATLAB/C interface. Here, the MATLAB implementation allocates the memory for the QP solver and avoids local copies of QP matrices. Note that the used QP solver from MATLAB, *mpcActiveSetSolver*$(\cdot)$, also supports automatic code generation. The implemented SQP algorithm approximates the Hessian of the Lagrange function by the damped BFGS approach presented in [NW06, Proc. 18.2]. This damped update rule modifies the third term in (A.3.1) such that the next iterate of the Hessian $\underline{H}_{l+1}$ is guaranteed to be positive definite. This modification is important since the underlying QP solvers, namely OSQP and *mpcActiveSetSolver*$(\cdot)$, assume positive (semi)definite Hessian matrices. However, not taking a full BFGS update step as shown in (A.3.1) might cause a poor convergence behavior (see, e.g., [NW06]). The initial Hessian matrix $\underline{H}_0$ is set to a scaled identity matrix with proper dimensions. The relative tolerance of the SQP is set to 10^{-3}, which is satisfied when all constraint violations and all elements of the KKT condition $\nabla_{\underline{z}}\mathcal{L}(\underline{z}_l, \tilde{\underline{\lambda}}_l, \tilde{\underline{\mu}}_l) = \underline{0}$ are below this tolerance (thus evaluating the infinity norm). For further parameters, refer to [Rös19, App. E.2.2.]. The *mpcActiveSetSolver*$(\cdot)$ is configured to use the default parameter values. The solver OSQP is also configured to use the default parameter values, except for the following modifications: Absolute tolerance $eps\_abs = 10^{-5}$, relative tolerance $eps\_rel = 10^{-5}$, primal infeasibility tolerance $eps\_prim\_inf = 10^{-4}$, dual infeasibility tolerance $eps\_dual\_inf = 10^{-4}$.

IPOPT

The developed software framework embeds the general purpose IPOPT solver [WB06] in the form of a pre-compiled MATLAB executable provided by [Ber20]. Therefore, the integration of IPOPT via existing binaries does not support code generation and is intended for the interpreted mode only. The IPOPT framework is configured to use the sparse linear solver MUMPS [Ame+01] and the limited memory BFGS update strategy [WB06]. The solver IPOPT is also configured to use the default parameter values, except for the following modifications: Update strategy for barrier parameter *mu_strategy* is set to *adaptive*, relative tolerance $tol = 10^{-3}$, maximal number of iterations $max\_iter = 200$. All resulting NLP formulations and the corresponding methods for determining derivative information pass IPOPT's first-order derivative test, which is based on internal finite differences, with a tolerance of at least 10^{-4}.

Benchmark Hardware

All offline simulations and statistical execution time measurements were performed on a conventional PC with the following software and hardware setup:

- CPU Intel Core i5-9500T (max. 3.7 GHz),
- Windows 10, 64-Bit-Version,
- Microsoft Visual Studio C++ 2015,
- MATLAB R2020b.

A.4. Discretization of System Dynamics

Assume that there exist an arbitrary vector $\underline{y} = (y_0, y_1, ..., y_N)^\intercal \in \mathbb{R}^N$ and the reference vector $\underline{y}_{\text{ref}} = (y_{\text{ref},0}, y_{\text{ref},1}, ..., y_{\text{ref},N})^\intercal \in \mathbb{R}^N$. Then, the normalized mean squared error (NRMSE) in this dissertation is given by:

$$NRMSE = \sqrt{\frac{\sum_{k=0}^{N}(y_k - y_{\text{ref},k})^2}{\sum_{k=0}^{N}\left(y_{\text{ref},k} - \frac{1}{N+1}\sum_{l=0}^{N} y_{\text{ref},l}\right)^2}}\,100\,\%. \qquad (A.4.1)$$

To capture the characteristic properties of the selected continuous-time benchmark systems, the motivation is to determine an accurate solution that is sufficiently close to the exact solution to the IVP in (3.4.3). To find a proper sampling time Δt_{s} and iterative solution method, this dissertation adopts the following procedure (see [Wor12, Sec. 1.3]). First, the input signal $\underline{\sigma}(t)$ is configured to be constant over the time interval $t \in [0, t_N]$ (e.g., unit step). The reference trajectory is determined with the fourth-order Runge-Kutta method and a small sampling time of $\Delta t_{\text{s}} = 2^{-12}\,\text{s}$. Assume that t_N is an integer multiple of the sampling time Δt_{s}. Then, for a less or equal accurate iterative solution method, the sampling time is defined by $\Delta t_i := 2^{-12+i}\,\text{s}$ with $i \in \mathbb{N}^{[0,12]}$. Therefore, all resulting sampling times Δt_i are an integer multiple of the base sampling time Δt_{s}, and the horizon length t_N is still an integer multiple of the chosen sampling time Δt_i. Let $\underline{\hat{\chi}}(t_k) = \left(\hat{\chi}_1(t_k), \hat{\chi}_2(t_k), ..., \hat{\chi}_p(t_k)\right)^\intercal := \underline{\phi}_\Sigma\left(t_k, \underline{x}_0, \underline{\sigma}(t)\right)$ denote the solution resulting from the current solution method with $t_{k+1} = t_k + \Delta t_i$ and $k \in \mathbb{N}_0$. Let $\chi_{\text{ref},j}(t_l)$ denote the j-th state generated with the comparative method on the time grid $t_{l+1} = t_l + 2^{-12}\,\mathrm{s}$ with $l \in \mathbb{N}_0$. The NRMSE is used to evaluate the numerical performance of the chosen configuration as follows:

$$e_{j,i} := \sqrt{\frac{\sum_{k=0}^{t_N/\Delta t_i} \left(\hat{\chi}_j(k\Delta t_i) - \chi_{\mathrm{ref},j}(k\,\Delta t_i)\right)^2}{\sum_{k=0}^{t_N/\Delta t_i} \left(\chi_{\mathrm{ref},j}(k\,\Delta t_i) - \frac{1}{t_N/\Delta t_i+1}\sum_{k=0}^{t_N/\Delta t_i} \chi_{\mathrm{ref},j}(k\,\Delta t_i)\right)^2}}\,100\,\%. \qquad \text{(A.4.2)}$$

Now, the variable i is increased from $i = 0$ to $i = 12$ as long as $\max\{e_{1,i}, e_{2,i}, ..., e_{p,i}\}$ $\leq e_{\max}$. The largest step size that satisfies the additional constraint with, for example, $e_{\max} = 0.005\,\%$ is defined to be the sampling time for the selected solution method. Since the sampled data formulation allows oversampling with $T_{\mathrm{s}} = \gamma\Delta t_i$, the number of numerical solution steps $\gamma \in \mathbb{N}$ per interval then defines the zero-order hold interval length and thus consequently the number of optimization parameters. Depending on the computing power, γ is chosen as small as possible, in particular, equal to one. A detailed analysis on choosing sampling times in the context of sampled data systems is provided in [NT04] (see also [Ack85]). The work in [Wor12, Sec. 1.3] analyzes numerically the detrimental effect of increasing zero-order hold interval lengths on the closed-loop control performance in the context of MPC.

A.5. Polynomial Input Domain Discretization

The exponential discretization proposed in Section 4.5 for generating time-varying finite control sets $\mathbb{A}(n)$ represents a fairly straightforward discretization approach. However, the formulation of the finite control sets in this dissertation also allows for other discretization approaches as long as the previously applied control vector is always contained in the current finite control set (see Rem. 4.5.1). The following input domain discretization is based on two composite polynomial functions and is motivated by the polynomial discretization presented in [Kel17]. In contrast to [Kel17], the resulting samples are guaranteed to be part of the input constraint set $\mathbb{U}$, with no post-processing. A similar description has been published in [Mak+18d]. The polynomial mapping first requires the following definitions:

$$u_{\mathrm{l}}(i) := u_{\min} + (i-1)\,\frac{u_{\max} - u_{\min}}{c-1},\ i \in \mathbb{N}^{[1,c]},\ c \in \mathbb{O}_{\geq 3},\ u_{\mathrm{h}} := \frac{u_{\min} + u_{\max}}{2}. \qquad \text{(A.5.1)}$$

Similar to (4.5.3), the polynomial mapping is defined by:

$$\eta(i, c, u, u_{\max}, u_{\min}) := \begin{cases} \tilde{p}_1\, u_{\mathrm{l}}^{\iota}(i) + \tilde{p}_2\, u_{\mathrm{l}}(i) + \tilde{p}_3 & \text{if } u_{\mathrm{l}}(i) \geq u_{\mathrm{h}}, \\ \tilde{p}_4\, u_{\mathrm{l}}^{\iota}(i) + \tilde{p}_5\, u_{\mathrm{l}}(i) + \tilde{p}_6 & \text{otherwise,} \end{cases} \qquad \text{with } \iota \in \mathbb{N}_{\geq 2}. \qquad \text{(A.5.2)}$$

The variable u_{h} represents the middle point of the compact and feasible control interval $[u_{\min}, u_{\max}]$ with $u_{\mathrm{l}}\big((c-1)/2+1\big) = u_{\mathrm{h}}$. A polynomial function is designed to the left and right of this value, respectively. The coefficients $\tilde{p}_1 - \tilde{p}_6$ represent the solution to the following linear system of equations in (A.5.3) (see also [Kel17]).

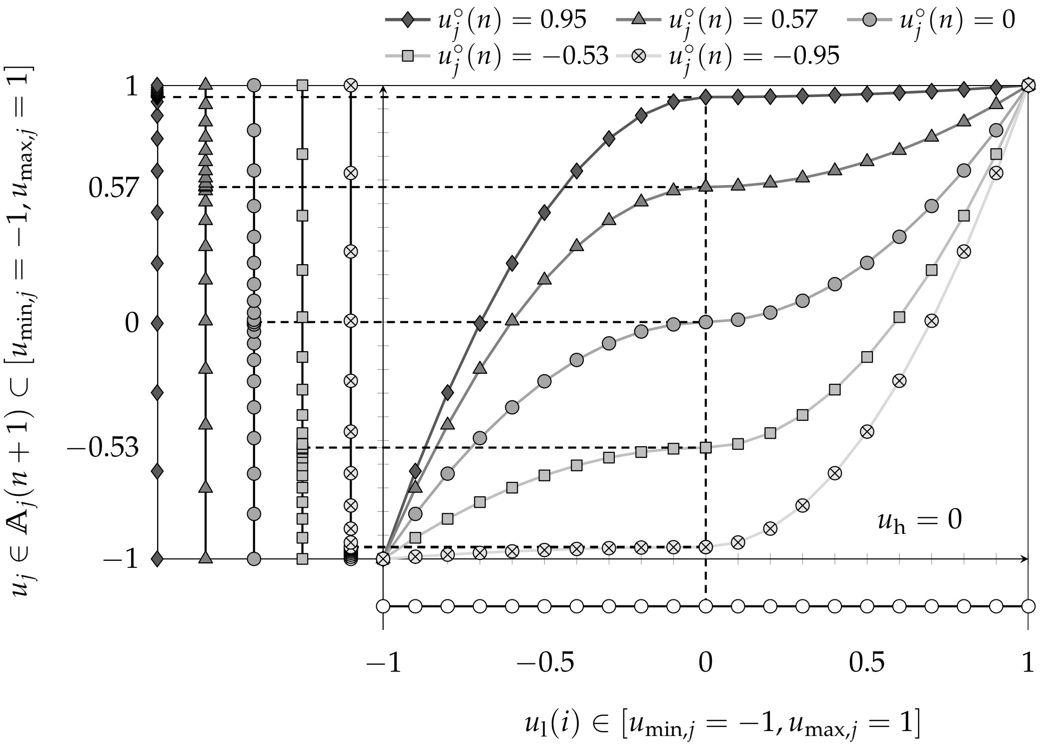


Figure A.3.: Input domain discretization based on piecewise polynomial functions. The coefficients of the individual polynomial functions depend, in particular, on the control $u_j^\circ(n)$. Left side: Location of control samples $u_j \in \mathbb{A}_j(n+1)$ on the interval $[u_{\mathrm{min},j}, u_{\mathrm{max},j}]$ after applying polynomial discretization with $u_{\mathrm{max},j} = 1$, $u_{\mathrm{min},j} = -1$, $\iota = 2$, and $c_j = 21$.

$$\begin{bmatrix} u_\mathrm{max}^\iota & u_\mathrm{max} & 1 & 0 & 0 & 0 \\ u_\mathrm{h}^\iota & u_\mathrm{h} & 1 & 0 & 0 & 0 \\ \iota u_\mathrm{h}^{(\iota-1)} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_\mathrm{min}^\iota & u_\mathrm{min} & 1 \\ 0 & 0 & 0 & u_\mathrm{h}^\iota & u_\mathrm{h} & 1 \\ 0 & 0 & 0 & \iota u_\mathrm{h}^{(\iota-1)} & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p}_1 \\ \tilde{p}_2 \\ \tilde{p}_3 \\ \tilde{p}_4 \\ \tilde{p}_5 \\ \tilde{p}_6 \end{bmatrix} = \begin{bmatrix} u_\mathrm{max} \\ u \\ 0 \\ u_\mathrm{min} \\ u \\ 0 \end{bmatrix} \tag{A.5.3}$$

The dependencies of the coefficients $\tilde{p}_1 - \tilde{p}_6$ on the control u are omitted here for simplicity. Figure A.3 shows several polynomial curves for different values of $u_j^\circ(n) \in \mathbb{U}$. This example is based on the configuration $u_{\mathrm{max},j} = 1$, $u_{\mathrm{min},j} = -1$, $\iota = 2$, and $c_j = 21$. The left side of Figure A.3 visualizes the resulting distribution on the interval $[-1, 1]$, which results from projecting the sampled control values on the y-axis. It should be noted that the time required for solving the linear system of equations (A.5.3) is negligible. The first and the fourth line of system (A.5.3) ensure that all samples are elements of $\mathbb{U}$ as long as $i \leq c_j$ holds. The second and the fifth line of system (A.5.3) ensure that $u_j^\circ(n) \in \mathbb{A}_j(n+1)$. By the third and sixth line of system (A.5.3), the polynomial functions are enforced to have zero slopes in the point $\big(u_\mathrm{h}, u_j^\circ(n)\big)$. This property ensures a saddle point in $\big(u_\mathrm{h}, u_j^\circ(n)\big)$ and thus a fine granularity of input sampling in the neighborhood of $u_j^\circ(n)$. Consequently, during closed-loop control, the finite control sets

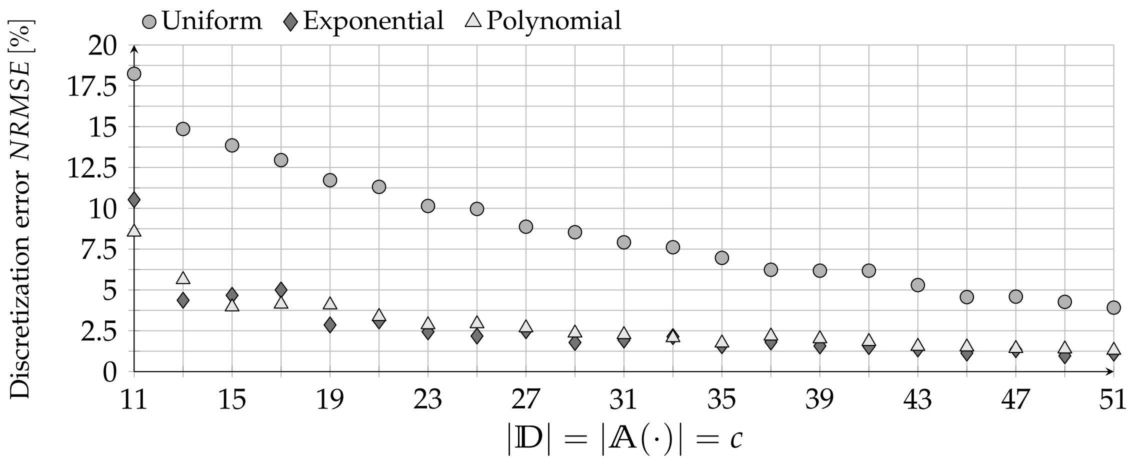


Figure A.4.: Extension of the left side of Figure 4.4 by the polynomial input domain discretization approach.

evolve as follows with, for example, $\underline{u}^{\circ}(n) = \big(u_1^{\circ}(n), u_2^{\circ}(n), ..., u_m^{\circ}(n)\big) := \underline{\mu}(\underline{x}_{\mu}(n), n)$:

$$\mathbb{A}_j(n+1) := \bigcup_{i=1}^{c_j} \Big\{\eta\big(i, c_j, u_j^{\circ}(n), u_{\max,j}, u_{\min,j}\big)\Big\},\ \forall j \in \mathbb{N}^{[1,m]},\ \forall n \in \mathbb{N}_0. \qquad \text{(A.5.4)}$$

Recall that (A.5.4) defines the finite control set for the j-th input dimension for the next closed-loop time instant $n+1$. This polynomial discretization follows the motivation presented in [Kel17] and enables a dense discretization in the vicinity of the previously applied control input if $\underline{u}^{\circ}(n) := \underline{\mu}(\underline{x}_{\mu}(n), n)$.
Finally, Figure A.4 extends the evaluation of the discretization error in Figure 4.4 by the polynomial discretization approach. This analysis in Figure A.4 reveals that both adaptive discretization approaches for the input domain, namely the exponential and the polynomial discretization, lead to similar discretization errors over cardinality c. However, implementing the exponential discretization is associated with less realization effort.

A.6. Further Details on Input Move-Blocking

Table A.1 lists and classifies key references on online MBMPC with a receding horizon that include discussions on stabilizing closed-loop properties. The last column in Table A.1 represents a subjective ranking of the realization effort required to implement each approach from scratch. A similar tabular summary has been published in [Mak+22]. The literature review reveals that the proposed move-blocking approaches in Section 7.2 and Section 7.3 close the existing gap and ensure both recursive feasibility and asymptotic stability for MBMPC for general nonlinear systems and with arbitrary move-blocking patterns.

Table A.1.: Literature review on input move-blocked online MPC [Mak+22]. NTI: Nonlinear time-invariant, LTI: Linear time-invariant, RTI: Real-time iteration, $+$: Statement applies, $-$: Statement does not apply, $\circ$: No detailed statement is made by the authors.

Reference	Benchmark system	Applicability to NTI systems	Recursive feasibility	Asymptotic stability	Optimal ($*$)/ Suboptimal (†)	Sparsity/Structure exploitation	Realization effort[5]
[Cag+07]	LTI	$-$	$+$	$+$	$*$	$\circ$	●
[GI07; Gon+09]	LTI	$+$	$+$	$-$	$*$	$\circ$	●●[6]
[Lon+11]	LTI	$+^1$	$+$	$+$	$*$	$\circ$	●●
[SM15][7]	LTI	$+$	$+$	$\circ^2$	$*$	$-$	●
[Che+20]	NTI	$+$	$-$	$-$	† (RTI)	$+$	●●●
[GR20]	NTI	$+$	$+^3$	$+^3$	† (RTI)	$+$	●●●
[Son+21]	LTI	$-$	$+$	$+$	$*$	$\circ$	●
[Mak+22]	NTI	$+$	$+$	$+$	†[4]	$+$	●

[1] Requires a sampled data formulation and a uniform move-blocking pattern;
[2] Proof of a valid Lyapunov function is not provided; [3] Derived from a terminal equality constraint;
[4] Formulation also includes optimal solution; [5] Subjective assessment; [6] This rating considers LTI systems;
[7] The evaluation in this row excludes optimal blocking (Sec. 4 in [SM15]).

B Supplemental Stability and Experimental Results

This appendix chapter expands on closed-loop stability analysis, provides details on system identification, and shows additional results of real-time closed-loop control.

B.1. One-Step Horizon and Time-Varying Finite Control Sets

The following Corollary B.1.1 replicates Corollary 4.5.1 for a concise presentation. The proof of Corollary B.1.1 adapts the major steps that are required to proof Theorem 19 and Theorem 2.39 in [Raw+20] for MPC with uncountable control sets (see Sec. 3.3 for summary) to SFMPC with time-varying finite control sets.

Corollary B.1.1: *SFMPC with finite control sets emulates conventional MPC.* Suppose Assumptions 3.1.1-3.1.3, 3.3.1-3.3.2, and 4.5.1 hold. Define $\mathbb{P}(\underline{x}_0, n) := \tilde{\mathbb{A}}(\underline{x}_0, n)$ and the control law $\underline{\mu}(\underline{x}_0, n) := \underline{u}_\mathrm{s}^*(\underline{x}_0, n)$ for all $\underline{x}_0 \in \mathcal{X}_1^\mathrm{s}$, all $n \in \mathbb{N}_0$. Then, the optimal cost function $V_1^\mathrm{s}(\cdot)$ in (4.5.1) represents a time-varying Lyapunov function in the set $\mathcal{X}_1^\mathrm{s}$ for the closed-loop system (4.5.2) with $\alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}_\infty$ for all $\underline{x}_0 \in \mathcal{X}_1^\mathrm{s}$, all $n \in \mathbb{N}_0$:

$$\alpha_1(\|\underline{x}_0\|) \leq V_1^\mathrm{s}(\underline{x}_0, n) \leq \alpha_2(\|\underline{x}_0\|),\ V_1^\mathrm{s}\Big(\underline{f}\big(\underline{x}_0, \underline{\mu}(\underline{x}_0, n)\big), n+1\Big) \leq V_1^\mathrm{s}(\underline{x}_0, n) - \alpha_1(\|\underline{x}_0\|). \tag{B.1.1}$$

Therefore, the origin is asymptotically stable in the positive invariant set $\mathcal{X}_1^\mathrm{s}$ for the closed-loop system (4.5.2).

Proof. Recall that all ingredients of OCP (4.5.1), excluding the time-varying finite control set $\tilde{\mathbb{A}}(\underline{x}_0, n)$, are time-invariant. Since Assumptions 3.1.1 and 3.1.2 ensure that all cost functions and the system dynamics are continuous, $J_1(\cdot)$ is also continuous [Raw+20, Prop. 2.4 (a)]. Assumption 3.1.3 ensures that $\mathbb{U}$ is compact such that the bounds $\max \mathbb{U}$ and $\min \mathbb{U}$ exist. These boundary vectors are required to design the finite control set $\mathbb{A}(n)$ for all $n \in \mathbb{N}_0$. Evaluating a finite subset $\tilde{\mathbb{A}}(\underline{x}_0, n) \subset \mathcal{U}_1^\mathrm{s}(\underline{x}_0)$ based on the continuous function $J_1(\cdot)$, including the admissible sample $\underline{u}_\mathrm{s}^0$ from Assumption 4.5.1, results in a list of function values for all $\underline{x}_0 \in \mathcal{X}_1^\mathrm{s}$ and all $n \in \mathbb{N}_0$ from which the optimum can be determined. It follows that OCP (4.5.1) with $\mathbb{P}(\underline{x}_0, n) := \tilde{\mathbb{A}}(\underline{x}_0, n)$ is feasible for all $\underline{x}_0 \in \mathcal{X}_1^\mathrm{s}$ and all $n \in \mathbb{N}_0$. If $\underline{x}_0 \in \mathcal{X}_1^\mathrm{s} \backslash \mathbb{X}_\mathrm{f}$, Assumption 4.5.1

ensures that there exists at least an admissible control sample $\underline{u}_{\mathrm{s}}^{0}$ that can steer the nonlinear system into $\mathbb{X}_{\mathrm{f}}$ in one step. By the definition of $\tilde{\mathbb{A}}(\underline{x}_0, n)$ in (4.5.6), the optimizer can always resort to the included local control law $\underline{u}_{\mathrm{s}}^{*}(\underline{x}_0, n) = \underline{\kappa}(\underline{x}_0)$ at each time instant $n \in \mathbb{N}_0$ if $\underline{x}_0 \in \mathbb{X}_{\mathrm{f}}$. By additionally including Assumption 3.3.1, it follows that $\alpha_{\ell}(\|\underline{x}_0\|) \le V_1^{\mathrm{s}}(\underline{x}_0, n) \le F(\underline{x}_0) \le \alpha_{\mathrm{f}}(\|\underline{x}_0\|)$ holds for all $\underline{x}_0 \in \mathbb{X}_{\mathrm{f}}$ and all $n \in \mathbb{N}_0$ (see monotonicity property in [Raw+20, Prop. 2.18]). By Assumption 3.1.3, the terminal set $\mathbb{X}_{\mathrm{f}}$ has an interior. Based on the previous statements, $V_1^{\mathrm{s}}(\cdot)$ is continuous at the origin (see [Raw+20, Prop. 2.38 (a)]). Since $\tilde{\mathbb{A}}(\underline{x}_0, n) \subset \mathcal{U}_1^{\mathrm{s}}(\underline{x}_0)$, the cost function $V_1^{\mathrm{s}}(\cdot)$ is uniformly bounded for all $n \in \mathbb{N}_0$. With these properties, Proposition 2.38 in [Raw+20], which includes Proposition 14 from [RR17b], states that there exists an upper bound $V_1^{\mathrm{s}}(\underline{x}_0, n) \le \alpha_2(\|\underline{x}_0\|)$ for all $\underline{x}_0 \in \mathcal{X}_1^{\mathrm{s}}$ and all $n \in \mathbb{N}_0$. Finally, by Assumption 3.3.2 and the definition of $\tilde{\mathbb{A}}(\underline{x}_0, n)$ in (4.5.6), it follows that $V_1^{\mathrm{s}}(\underline{x}_0^{+}, n+1) \le F(\underline{x}_0^{+}) = V_1^{\mathrm{s}}(\underline{x}_0, n) - \ell\big(\underline{x}_0, \underline{u}_{\mathrm{s}}^{*}(\underline{x}_0, n)\big) \le V_1^{\mathrm{s}}(\underline{x}_0, n) - \alpha_{\ell}(\|\underline{x}_0\|)$ holds for all $\underline{x}_0 \in \mathcal{X}_1^{\mathrm{s}}$ and all $n \in \mathbb{N}_0$. Thus, the proof for the existence of the time-varying Lyapunov function in (B.1.1) is established. By Theorem 2.39 in [Raw+20] (time-variant version of Thm. 3.3.2), the origin is asymptotically stable for the closed-loop system (4.5.2). The set $\mathcal{X}_1^{\mathrm{s}}$ is positive invariant since the optimizer can always resort to the local control law $\underline{\kappa}(\cdot)$ if the initial state $\underline{x}_0$ is a member of the control invariant terminal set $\mathbb{X}_{\mathrm{f}}$. $\square$

B.2. Variable Horizon and Time-Varying Finite Control Sets

The following Proposition B.2.1 replicates Proposition 8.1.1 for a concise presentation. The proof of Proposition B.2.1 is inspired by Theorem 19 and Theorem 2.39 in [Raw+20], finite completion times with variable horizon MBMPC from [SM12], and the derivations of the dual-mode control in [MM93].

Proposition B.2.1: *SFMPC with variable horizon and finite control sets.* Suppose Assumptions 3.1.1-3.3.2 and 4.5.1 hold. Then, the implicit control law $\underline{\mu}(\underline{x}_0, n) := \underline{u}_{\mathrm{s}}^{*}(\underline{x}_0, n)$, which is driven by the solutions to the mixed-integer OCP (8.1.5), renders the feasible set $\bar{\mathcal{X}}_{\mathrm{s}}$ positive invariant and the origin asymptotically stable in the sense of Lyapunov for the closed-loop system (4.5.2).

Proof. Assumptions 3.1.1 and 3.1.2 ensure that all cost functions and the system dynamics are continuous. Therefore, $J_N(\cdot)$ is continuous for a given horizon length N [Raw+20, Prop. 2.4 (a)]. By adding Assumption 3.1.3, it follows that $\mathcal{U}_N(\underline{x}_0) \subset \mathbb{U}^N$ is compact [Raw+20, Prop. 2.4 (b)]. Thus, the set $\mathcal{U}_N^{\mathrm{s}}(\underline{x}_0) \subset \mathbb{U}^N$ is also compact for a given horizon length N (see Sec. 4.1). Since, by Assumption 3.1.3, the set $\mathbb{U}$ is compact, the bounds $\max \mathbb{U}$ and $\min \mathbb{U}$ exist that are required to design the finite control set $\mathbb{A}(n) \subset \tilde{\mathbb{A}}(\underline{x}_0, n)$ for all $n \in \mathbb{N}_0$. Further, Assumption 4.5.1 ensures that there exists an initial and admissible tuple $(\underline{u}_{\mathrm{s}}^{0}, N)$ for every $\underline{x}_0 \in \bar{\mathcal{X}}_{\mathrm{s}}$ with $\underline{u}_{\mathrm{s}}^{0} \in \tilde{\mathbb{A}}(\underline{x}_0, 0)$. Therefore, iterating over a finite number of horizon lengths $N \in \mathcal{N} := [1, N_{\max}]$ and control candidates $\tilde{\mathbb{A}}(\underline{x}_0, 0)$ and evaluating the continuous cost function $J_N(\cdot)$ results in a list of function values from which the global optimum $(\underline{u}_{\mathrm{s}}^{*}(\underline{x}_0, 0), N^{*})$ can be determined at time instant $n = 0$. Recall that every admissible control candidate $\underline{u}_{\mathrm{s}} \in \tilde{\mathbb{A}}(\underline{x}_0, n)$

results in $\Theta_N^{\mathrm{s}}(\underline{u}_{\mathrm{s}}) \in \mathcal{U}_N^{\mathrm{s}}(\underline{x}_0)$ for all $\underline{x}_0 \in \mathcal{X}_N^{\mathrm{s}}$ and all $n \in \mathbb{N}_0$.
Case I: $N^* > 1$. As noted in Remark 4.5.1, the adaptive input domain discretization in (4.5.4) always embeds the previous control sample $\underline{\mu}(\underline{x}_\mu(n-1), n-1)$ into the current finite control set $\mathbb{A}(n)$ and therefore also into $\tilde{\mathbb{A}}(\underline{x}_\mu(n), n)$ since $\mathbb{A}(n) \subset \tilde{\mathbb{A}}(\underline{x}_\mu(n), n)$. The optimizer can at least resort to the tuple $\big(\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n), N^*-1\big)$ at time instant $n+1$. In the nominal case, this property ensures recursive feasibility for at least N^* closed-loop steps and progress towards the terminal set $\mathbb{X}_{\mathrm{f}}$ with $V_{N^*}^{\mathrm{s}}(\underline{x}_0^+, n+1) - V_{N^*}^{\mathrm{s}}(\underline{x}_0, n) \leq J_{N^*-1}\big(\underline{x}_0^+, \Omega_{\mathrm{st}}\big(\Theta_{N^*-1}^{\mathrm{s}}\big(\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)\big)\big)\big) - V_{N^*}^{\mathrm{s}}(\underline{x}_0, n) \leq \ell\big(\underline{x}_0, \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)\big)$. By Assumption 3.3.1, the stage cost function $\ell(\cdot)$ is only zero at the origin. This Case I applies recursively until the optimizer returns $N^* = 1$.
Case II: $N^* = 1$. Here, $\underline{x}_0 \in \mathcal{X}_1^{\mathrm{s}}$ applies. If $\underline{x}_0 \notin \mathbb{X}_{\mathrm{f}}$, the optimal control vector $\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n) \in \tilde{\mathbb{A}}(\underline{x}_0, n)$, which transfers the nonlinear system in one step into $\mathbb{X}_{\mathrm{f}}$, follows from the previous closed-loop step with $\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n) = \underline{u}_{\mathrm{s}}^*\big(\underline{x}_\mu(n-1), n-1\big)$ (see Rem. 4.5.1), by Assumption 4.5.1 with $\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n) = \underline{u}_{\mathrm{s}}^0$, or is some other admissible control candidate $\underline{u}_{\mathrm{s}}$ with $\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n) = \underline{u}_{\mathrm{s}} \in \tilde{\mathbb{A}}(\underline{x}_0, n)$. Based on the definition of the finite set $\tilde{\mathbb{A}}(\underline{x}_0, n)$ in (4.5.6), the optimizer can resort to $\underline{u}_{\mathrm{s}}^*(\underline{x}_0, n) = \underline{\kappa}(\underline{x}_0)$ for all $n \in \mathbb{N}_0$ if $\underline{x}_0 \in \mathbb{X}_{\mathrm{f}}$. By Assumption 3.3.2, the local control law $\underline{\kappa}(\cdot)$ asymptotically stabilizes the origin for the nonlinear closed-loop system (4.5.2) with $\underline{\mu}(\underline{x}_0, n) := \underline{\kappa}(\underline{x}_0)$ for all $\underline{x}_0 \in \mathbb{X}_{\mathrm{f}}$ and all $n \in \mathbb{N}_0$. $\square$

Note that this proof does not rely on an explicit Lyapunov function candidate. This proof only attests that the closed-loop system (4.5.2) is transferable to the terminal set $\mathbb{X}_{\mathrm{f}}$ by applying the implicit control law $\underline{\mu}(\underline{x}_0, n) := \underline{u}_{\mathrm{s}}^*(\underline{x}_0, n)$ for all $\underline{x}_0 \in \bar{\mathcal{X}}_{\mathrm{s}}$ and all $n \in \mathbb{N}_0$. Inside the terminal set $\mathbb{X}_{\mathrm{f}}$, the implicit control law replicates the local controller $\underline{\kappa}(\cdot)$ and thus stabilizes the origin by Assumption 3.3.2.

B.3. Experimental Stability Analysis

This section shows a possible procedure for approximating the region of attractions of MPC and SFMPC for the Van der Pol Oscillator in Section 5.2. This experimental stability analysis represents an alternative approach when theoretical closed-stability results cannot be derived. The proposed approach involves the following assumption (cf. [GP17, Def. 2.15]).

Assumption B.3.1: *Local stability in some neighborhood of the origin.* There exists a positive invariant terminal region $\mathcal{T} := \mathcal{X}_N \cap \mathcal{X}_N^{\mathrm{s}} \cap \mathcal{B}_\delta$ for some small $\delta > 0$ such that $\underline{f}(\underline{x}, \underline{\mu}(\underline{x})) \in \mathcal{T}$ if $\underline{x} \in \mathcal{T}$.

This assumption only implies that the application of the implicit control law keeps the system in some close neighborhood of the origin. Here, the parameter $\delta > 0$ is subject to a heuristic estimation. The stabilizable set includes all initial states $\underline{x}_{\mu,0}$ from which the closed-loop system (3.3.2) can be driven to the terminal set $\mathcal{T}$ in at most $l \in \mathbb{N}$ steps (see [KM00a, Def. 2.7]):

$$\begin{aligned} \mathcal{S}_l := \{ \underline{x} \in \mathcal{X}_N \,|\, \exists i \leq l \text{ such that } & \underline{\varphi}_\mu(n, \underline{x}) \in \mathcal{X}_N, \forall\, n \in \mathbb{N}^{[1,i-1]}, \\ & \underline{\varphi}_\mu(n, \underline{x}) \in \mathcal{T}, \forall\, n \in \mathbb{N}^{[i,l]} \}. \end{aligned} \tag{B.3.1}$$

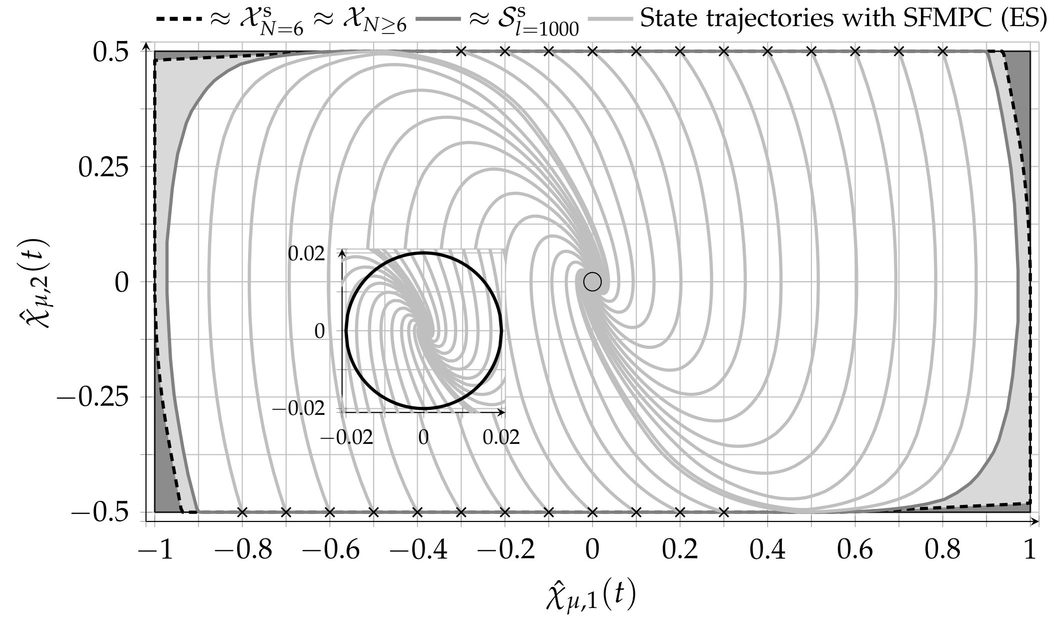


Figure B.1.: Feasible and l-step stabilizable set of SFMPC applied to the Van der Pol oscillator. The graphs inside the stabilizable set $\mathcal{S}_{l=1000}$ (white area) show the closed-loop state trajectories with SFMPC for different initial state vectors marked by a cross. For this system, the single degree of freedom in control has only a minor adverse impact on the size of the stabilizable set (light gray area). Subplot: Terminal ball $\mathcal{T}$.

Consequently, the stabilizable set for SFMPC is defined by:

$$\mathcal{S}^{s}_{l} := \{\underline{x} \in \mathcal{X}^{s}_{N} \mid \exists i \leq l \text{ such that } \underline{\varphi}_{\mu}(n, \underline{x}) \in \mathcal{X}^{s}_{N}, \forall n \in \mathbb{N}^{[1,i-1]}, \\ \underline{\varphi}_{\mu}(n, \underline{x}) \in \mathcal{T}, \forall n \in \mathbb{N}^{[i,l]}\}. \tag{B.3.2}$$

Note that with $\mathbb{X}_{\mathrm{f}} = \mathbb{X}$, the relations $\mathcal{S}_l \subseteq \mathcal{X}_N$ and $\mathcal{S}^{s}_{l} \subseteq \mathcal{X}^{s}_{N} \subseteq \mathcal{X}_N$ hold for all $l \in \mathbb{N}$. The origin is only practically asymptotically stable in $\mathcal{S}_\infty$ or $\mathcal{S}^{s}_{\infty}$ (maximal stabilizing sets [KM00a]) for the closed-loop system (3.3.2) since the terminal region $\mathcal{T}$ has an interior in which the origin is not assumed to be necessarily attractive. Practical asymptotic stability turns into asymptotic stability if the terminal region is defined as $\mathcal{T} := \underline{0}_p$ [GP17, Def. 2.15]. Notice that with $\mathcal{T} := \underline{0}_p$, the number of steps might tend to $l \to \infty$ such that a numerical estimation of $\mathcal{S}^{s}_{l}$ would not terminate in finite time. Figure B.1 shows the results of the experimental stability analysis of the origin. Along each dimension of the constrained state space 501 uniformly distributed samples are placed, resulting in a finite grid with 501×501 grid points. Each initial state vector is verified whether it is a member of some set of interest. Convex polytopes finally enclose all verified members and thus approximate the originally uncountable individual sets. If OCP (4.1.4) with $\bar{\mathbb{P}} := \mathbb{U}$ (or also $\bar{\mathbb{P}} := \mathbb{D}$) and $N = 6$ is feasible for some grid point $\underline{x}$, this initial state vector $\underline{x}$ is a member of $\mathcal{X}^{s}_{N=6}$. If the closed-loop system can be transferred to the terminal ball $\mathcal{T}$ in at most $l = 1000$ closed-loop steps starting at $\underline{x}$, then $\underline{x}$ is a member of the stabilizable set $\mathcal{S}^{s}_{l=1000}$. Notice that the subplot in Figure B.1 shows the terminal ball $\mathcal{T}$ with $\delta = 0.02$. After entering the terminal ball $\mathcal{T}$, the state trajectories remain inside for all subsequent time steps. The black dashed convex polytope approximates the feasible set $\mathcal{X}^{\mathrm{s}}_{N=6}$, and it is only slightly smaller than the state constraint set $\mathbb{X}$. The dark gray shaded area visualizes the difference between the latter two sets. For the initial state vectors inside the dark gray shaded area, applying the maximum or minimal input over $N = 6$ steps is not strict enough to prevent the closed-loop system from violating the state box-constraints. Hence, $\mathcal{X}^{\mathrm{s}}_{N=6} = \mathcal{X}_{N\geq 6}$ applies to this benchmark system. The area that is shaded by a light gray color contains the initial states from which the closed-loop system (3.3.2) does not reach the terminal set $\mathcal{T}$ in $l = 1000$ steps. The evolutions of the example trajectories over $l = 1000$ steps, which are based on the solutions to OCP (4.5.1) with $\mathbb{P}(\underline{x}_0, n) := \mathbb{A}(n)$, however, reveal that this light gray area represents the initial states for which the closed-loop control is not recursively feasible. Consequently, the light gray shaded area represents the maximal theoretical reduction of the stabilizable set $\mathcal{S}_l$ resulting from the extreme input move-blocking in case of SFMPC, described by $\mathcal{S}^{\mathrm{s}}_l$. Recall that $\mathcal{S}_l \subseteq \mathcal{X}_N$ applies by design. The subplot in Figure B.1 shows that with $l = 1000$ the stabilizable set $\mathcal{S}^{\mathrm{s}}_{l=1000}$ is a close approximation of the region of attraction of the origin for the closed-loop system (3.3.2). The exemplary state trajectories not only end in the target region, but remain in the region for many steps and even converge towards the origin. Thus, this systematic experimental analysis reveals an extensive region of attraction with SFMPC and provides an alternative method for checking stability properties when theoretical derivations are not available. However, the combinatorial complexity of initial states grows exponentially with the state dimension p, thus restricting this approach to small-sized systems.

B.4. Experimental System Identification

This section presents the identification signals and the resulting model performances. In addition, information about data acquisition during identification is provided.

Directional Control Valve

The challenge in identifying the dynamics of the directional control valve consists in the design of a proper input stimulus. The objective is to design the input voltage signal $u_{\mathrm{a}}(t)$ in such a way that it does not drive the piston into the mechanical stops while stimulating a wide characteristic valve behavior. A linear dynamic and continuous-time model is chosen to approximate the nonlinear valve dynamics on average. The estimation of the model parameters is based on smooth least-squares optimization (see the identification framework *tfest*$(\cdot)$ in MATLAB). Here, a quadratic scalar cost function evaluates the difference between the measured and simulated position signals $y_{\mathrm{m}}(t_n)$ and $y(t_n)$, respectively, on a uniform time grid $t_{n+1} = t_n + \Delta t_{\mathrm{s}}$ with $n \in \mathbb{N}_0$. Before starting the identification procedure of the linear dynamics, the mean values of the individual signals are removed. To account for the fast valve dynamics, the real-time target machine (see Fig. 6.3) logs the identification signals at a sampling rate of $f_{\mathrm{s}} = 1\,\mathrm{MHz}$, which corresponds to a sampling time of $\Delta t_{\mathrm{s}} = 1\,\mu\mathrm{s}$. Afterward, the logged data is sampled down offline by an integer multiple of 20 to a sampling

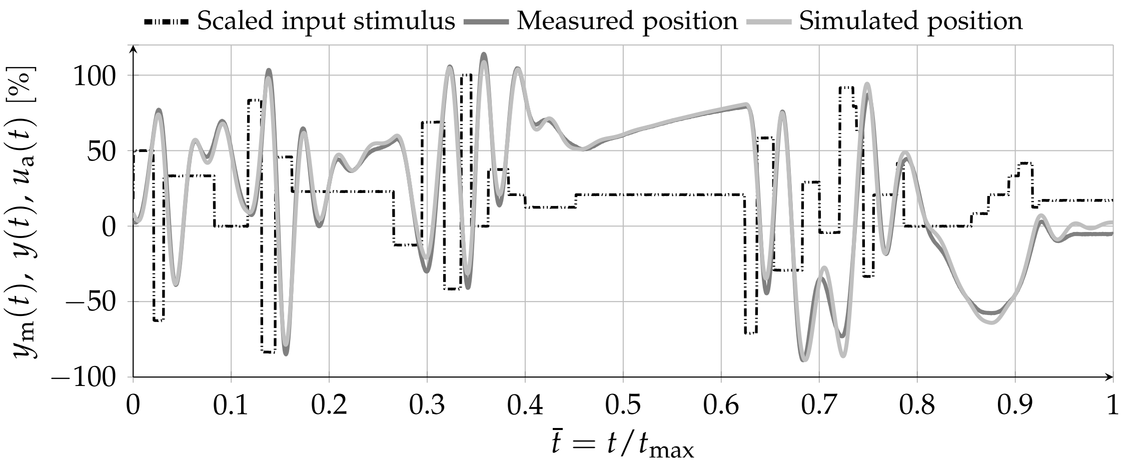


Figure B.2.: Valve identification signals and model performance. The densely dash-dotted signal represents the solenoid input signal $u_a(t)$ multiplied by 100 %/24 V.

rate of $f_s = 50\,\text{kHz}$, which corresponds to a sampling time of $\Delta t_s = 20\,\mu\text{s}$. Since the data accusation is done in open-loop, the piston and the armature are in most of the experimental time in motion. Figure B.2 highlights model uncertainty especially at the end of the experiment for $\bar{t} > 0.9$ when nonlinear friction forces occur in the small velocity range. The model performance reaches a fitness value of $NRMSE = 9.69\,\%$ during identification. For the purpose of validation, the identified model is stimulated once with a different ARBS as the input stimulus. Figure B.3 attests a validation performance of $NRMSE = 12.43\,\%$ with respect to the measured position signal $y_m(t)$.

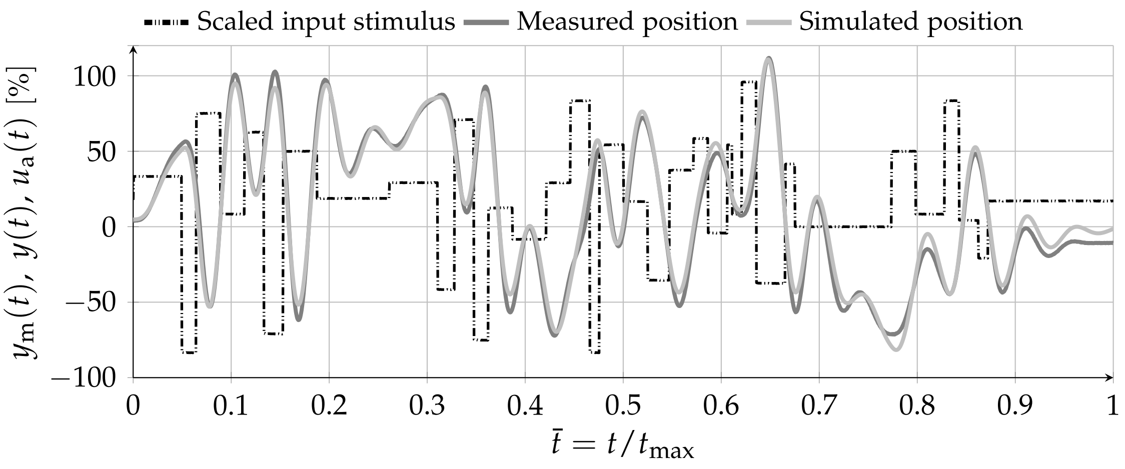


Figure B.3.: Valve validation signals and model performance. The densely dash-dotted signal represents the solenoid input signal $u_a(t)$ multiplied by 100 %/24 V.

Industrial Plant Emulator

The identification input stimulus of the Industrial Plant Emulator also represents an ARBS for motor 1. The estimation of the model parameters is based on global optimization (see pattern search in MATLAB). Here, the objective function represents the weighted sum of the NRMSE with respect to the position, evaluating the difference

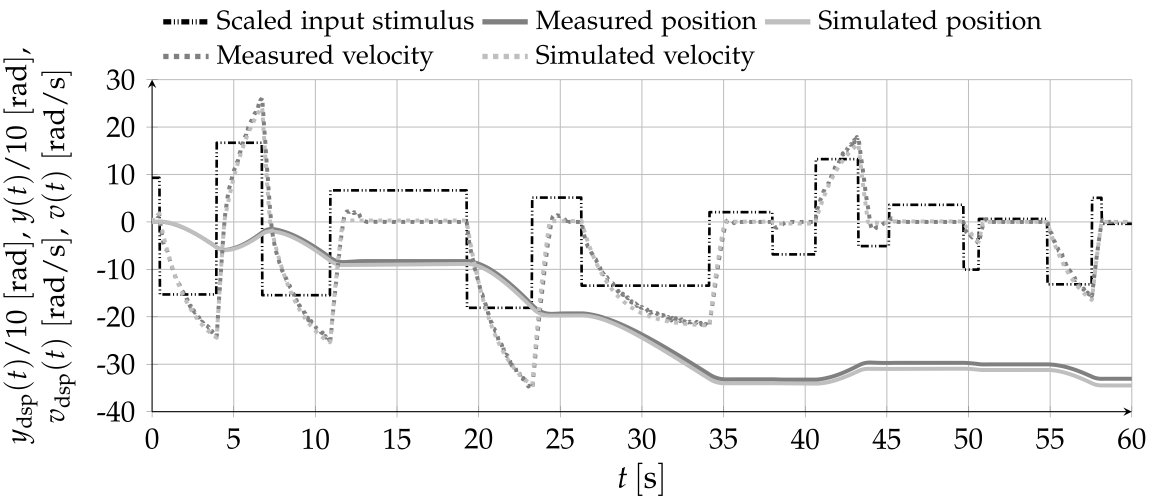


Figure B.4.: Servo motor identification signals and model performance. The black solid line represents the input stimulus $i_{\mathrm{ref},1}(t)$ multiplied by 20/A.

between the measured and simulated position $y_{\mathrm{dsp}}(t_n)$ and $y(t_n)$, and the NRMSE with respect to the velocity, evaluating the difference between the measured and simulated velocity $v_{\mathrm{dsp}}(t_n)$ and $v(t_n) = \dot{y}(t_n)$. Data logging operates at a fixed sampling time of $\Delta t_{\mathrm{s}} = 0.01\,\mathrm{s}$ with $t_{n+1} = t_n + \Delta t_{\mathrm{s}}$ and $n \in \mathbb{N}_0$. The simulation is based on the fourth-order Runge-Kutta method with a sampling time of $\Delta t_{\mathrm{s}} = 0.01\,\mathrm{s}$. The identified parameter values of model (6.2.2) are $p_1 = 13.92, p_2 = 5.28, p_3 = 0.55$, and $p_4 = 38.92$. For the identification stimulus in Figure B.4, the simulated position and velocity signals reach fitness values of $NRMSE = 7.45\,\%$ and $NRMSE = 7.48\,\%$, respectively. For the validation stimulus in Figure B.5, the simulated position and velocity signals reach fitness values of $NRMSE = 13.42\,\%$ and $NRMSE = 10.55\,\%$, respectively.

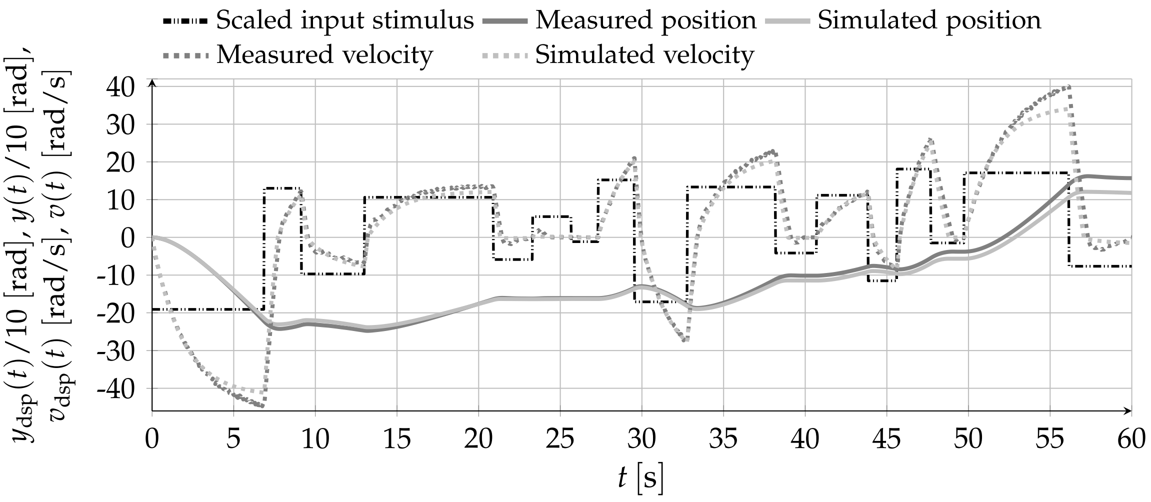


Figure B.5.: Servo motor validation signals and model performance. The black solid line represents the input stimulus $i_{\mathrm{ref},1}(t)$ multiplied by 20/A.

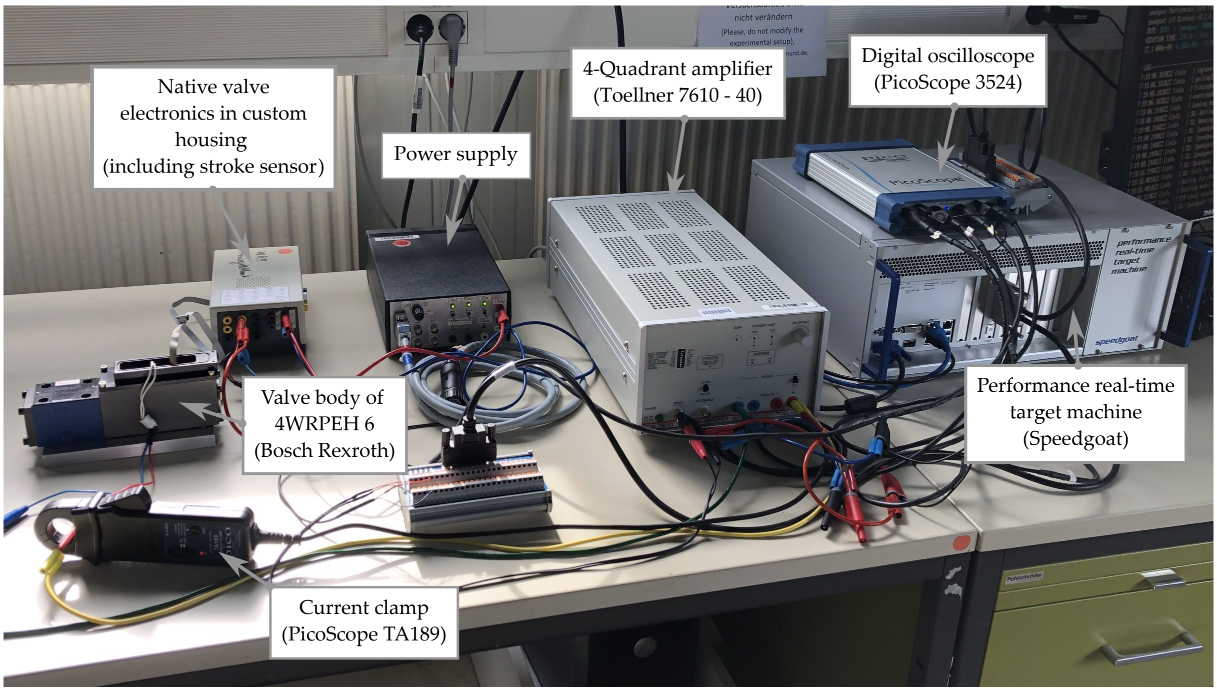

Native valve electronics in custom housing (including stroke sensor)
Power supply
4-Quadrant amplifier (Toellner 7610 - 40)
Digital oscilloscope (PicoScope 3524)
Valve body of 4WRPEH 6 (Bosch Rexroth)
Performance real-time target machine (Speedgoat)
Current clamp (PicoScope TA189)

Figure B.6.: Picture of the valve test bench for prototyping. See also Figure 6.3.

B.5. Valve Test Bench for Prototyping

Figure B.6 shows a picture of the experimental setup for real-time valve control. For a detailed description of the setup, refer to Section 6.1.1 and Figure 6.3. The digital oscilloscope and current clamp are used to track additional signals such as the solenoid current to prevent the valve from electrical overload.

B.6. Additional Results of Closed-Loop Valve Control

This section verifies the high closed-loop valve control performance in Section 6.1.4 using another reference profile. Figure B.7 (see end of chapter) shows high closed-loop reference tracking performance with an inactive velocity limit for both SFMPC and LMPC. The position signal $y_{\mathrm{ls}}(t)$ shows a short rise time and almost no overshooting. The compensation technique from Section 6.1.2 steers the piston in some close neighborhood of the reference position and thus compensates effectively for model-mismatch. Both control approaches offer a low computational effort, however, the exhaustive search algorithm in SFMPC exhibits a more deterministic runtime characteristic that never violates real-time constraints ($t_{\mathrm{run}}/\Delta t_{\mathrm{s}} \leq 1$). Here, the closed-loop real-time control operates at a fixed sampling time of $\Delta t_{\mathrm{s}} = 100\,\mu\mathrm{s}$ ($f_{\mathrm{s}} = 10\,\mathrm{kHz}$). Figure B.8 (see end of chapter) verifies that the softened OCP formulation in Section 5.3 leads to a smooth closed-loop control though SFMPC temporarily violates the velocity limit $v_{\mathrm{lim}} = 2000\,\%\,s^{-1}$ in the presence of model mismatch and disturbances. Since LMPC does not include robustification strategies in this dissertation, the closed-loop performance deteriorates significantly and the optimization time fluctuates. In contrast to the softened SFMPC, nominal LMPC violates real-time constraints ($t_{\mathrm{run}}/\Delta t_{\mathrm{s}} > 1$).

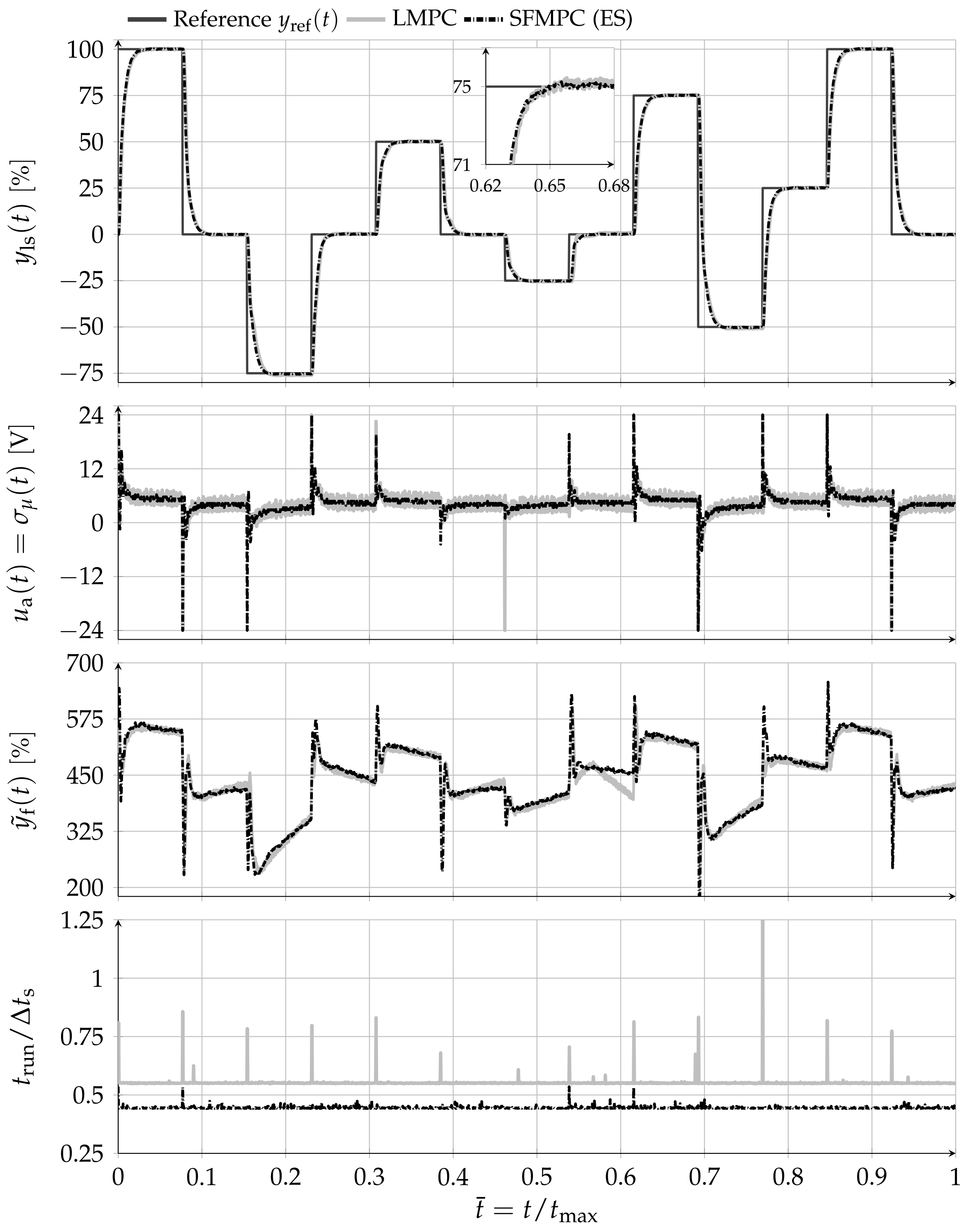


Figure B.7.: Valve closed-loop real-time control for an additional reference profile: Constant reference tracking with an **inactive** stroke **velocity limit**. First: Stroke time performance. Second: Solenoid input voltage. Third: Reference adaptation due to offset compensation. Bottom: Normalized execution time.

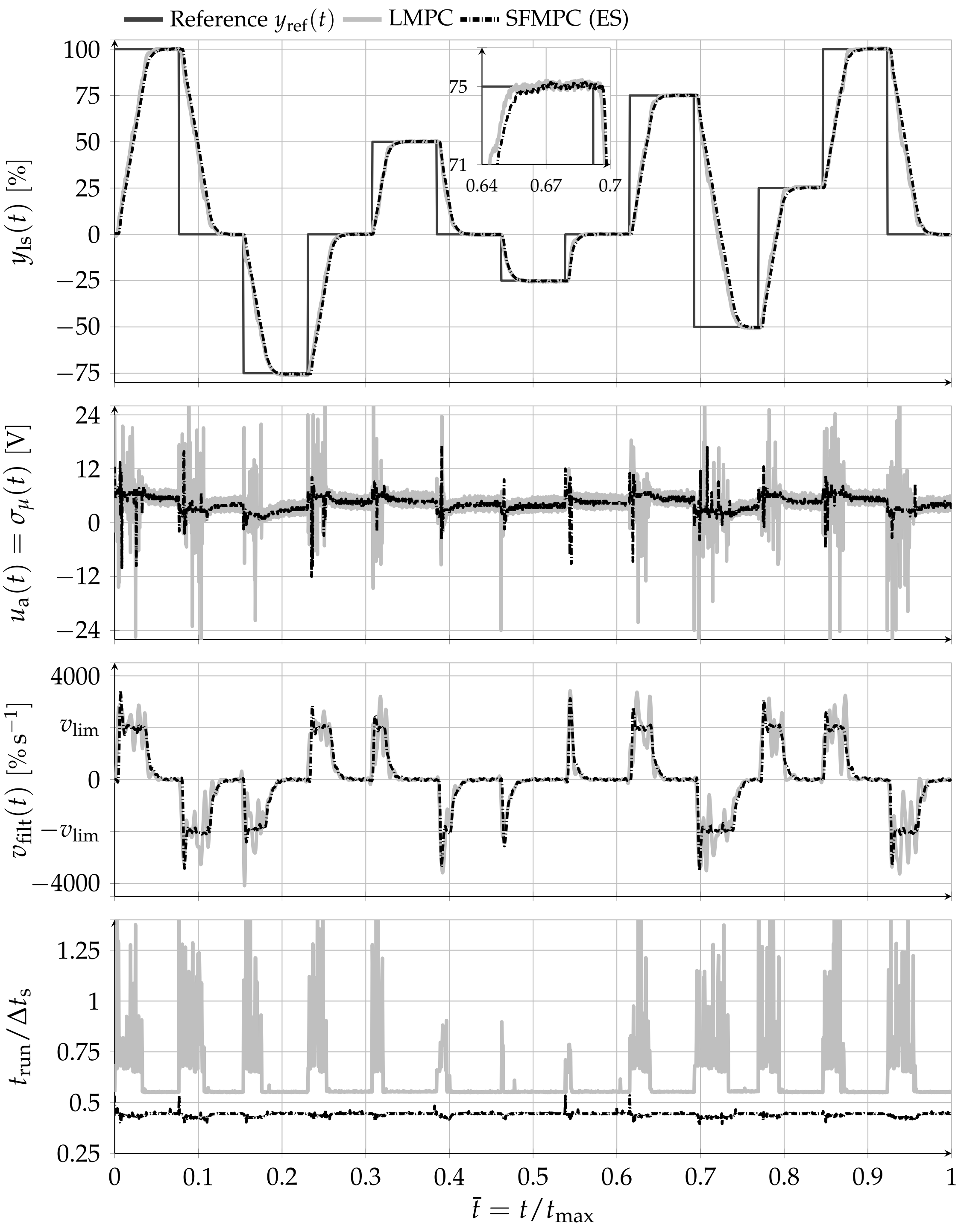


Figure B.8.: Valve closed-loop real-time control for an additional reference profile: Constant reference tracking with an **active** stroke **velocity limit**. First: Stroke time performance. Second: Solenoid input voltage. Third: Reference adaptation due to offset compensation. Bottom: Normalized task execution time.

B.7. Model-Based System Optimization

Model-based system optimization represents another important application for gradient-free MPC and is highly relevant for the virtual and automated development of prototypes of mechatronic systems. For example, the overall performance of a directional control valve depends on the smooth interaction of different internal components during closed-loop control. A model-based system development enables the analysis and optimization of physical interactions at an early stage of design. In general, the earlier the solenoid is matched to the valve mechanics, for example, the more the controller can contribute to the overall closed-loop control performance of the valve, rather than compensating for mechanical and electromagnetic shortcomings. The model-based system optimization framework proposed in [Mak+18a] enables systematic evaluation of the closed-loop control performances of virtual valve prototypes. However, since the closed-loop control evaluation in [Mak+18a] is integrated with global optimization of the valve hardware, the system behavior changes with the design parameters. Therefore, to decouple the robustness properties of the controller from the hardware optimization result, the controller parameters must be adjusted every time the global optimizer provides a new hardware configuration. At this point, it is convenient to include the controller parameters into the hardware optimization loop. However, with the native vale controller, the parameter complexity increases significantly and thus the required optimization time. The contribution in [Mak+18c] addresses this optimization complexity by introducing MPC for the system optimization of a directional control valve. The main idea is to include the model, on which the system optimization is based, as the internal prediction model. Hence, variations of the hardware design parameters are directly translated into the controller's domain. The free parameters of the controller are again the weights of the individual cost terms, which can be included in the hardware optimization loop either directly or via a tailored two-stage approach presented in [Mak+18b]. Since the base evaluation model includes non-smooth components such as characteristic force curves, a derivative-free MPC scheme qualifies for virtual closed-loop control.

Bibliography

[AB09] A. Alessio and A. Bemporad. “A Survey on Explicit Model Predictive Control”. In: *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Ed. by L. Magni, D. M. Raimondo, and F. Allgöwer. 1st ed. Lecture Notes in Control and Information Sciences. Berlin, Heidelberg: Springer-Verlag, 2009. Chap. 6, pp. 345–369.

[AB95] M. Alamir and G. Bornard. “Stability of a Truncated Infinite Constrained Receding Horizon Scheme: the General Discrete Nonlinear Case”. In: *Automatica* 31.9 (1995), pp. 1353–1356.

[Ack85] J. Ackermann. *Sampled-Data Control Systems: Analysis and Synthesis, Robust System Design*. Berlin, Heidelberg: Springer, 1985.

[All+16] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings. *On the inherent robustness of optimal and suboptimal MPC*. Tech. rep. Madison, WI 53706, USA: Deptartment of Chemical and Biological Engineering, University of Wisconsin-Madison, 2016.

[All+17] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings. “On the inherent robustness of optimal and suboptimal nonlinear MPC”. In: *Systems & Control Letters* 106 (2017), pp. 68–78.

[AM89] B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall International, 1989.

[Ame+01] P. Amestoy, I. S. Duff, J. Koster, and J.-Y. L’Excellent. “A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling”. In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 15–41.

[And+18] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. “CasADi: a software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2018), pp. 1–36.

[AQ11] R. P. Aguilera and D. E. Quevedo. “On the stability of MPC with a Finite Input Alphabet”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 7975–7980.

[AQ13] R. P. Aguilera and D. E. Quevedo. “Stability Analysis of Quadratic MPC With a Discrete Input Alphabet”. In: *IEEE Transactions on Automatic Control* 58.12 (2013), pp. 3190–3196.

[BC12] G. M. Bone and X. Chen. “Position Control of Hybrid Pneumatic-Electric Actuators”. In: *American Control Conference (ACC)*. 2012, pp. 1793–1799.

[BD93] Z. Bai and J. W. Demmel. “On Swapping Diagonal Blocks in Real Schur Form”. In: *Linear Algebra and its Applications* 186 (1993), pp. 75–95.

[Bem+00] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos. “The Explicit Solution of Model Predictive Control via Multiparametric Quadratic Programming”. In: *American Control Conference (ACC)*. 2000, pp. 872–876.

[Ber20] E. Bertolazzi. *IPOPT Interface for MATLAB*. `https://github.com/ebertolazzi/mexIPOPT`. Used binaries from release tag 1.1.3. 2020.

[Bet98] J. T. Betts. “Survey of Numerical Methods for Trajectory Optimization”. In: *Journal of Guidance, Control, and Dynamics* 21.2 (1998), pp. 193–207.

[Bin+01] T. Binder, L. Blank, H. G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. P. Schlöder, and O. von Stryk. “Introduction to Model Based Optimization of Chemical Processes on Moving Horizons”. In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke, and J. Rambau. 1st ed. Berlin, Heidelberg: Springer, 2001. Chap. 3, pp. 295–339.

[BL17] R. Bobiti and M. Lazar. “Towards Parallelizable Sampling–based Nonlinear Model Predictive Control”. In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 13176–13181.

[Blo97] E. D. Bloch. *A First Course in Geometric Topology and Differential Geometry*. Modern Birkhäuser Classics. Boston: Birkhäuser, 1997.

[Bob17] R. V. Bobiti. “Sampling–driven stability domains computation and predictive control of constrained nonlinear systems”. Dissertation. Eindhoven University of Technology, 2017.

[Boc+14] A. Boccia, L. Grüne, and K. Worthmann. “Stability and feasibility of state constrained MPC without stabilizing terminal constraints”. In: *Systems & Control Letters* 72 (2014), pp. 14–21.

[Bor+17] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge: Cambridge University Press, 2017.

[Bos10] Bosch Rexroth AG. *4/4-way servo solenoid directional control valves, directly operated, with electrical position feedback and on-board electronics (OBE)*. RD 29035/10.10. Replaces: 05.10. 2010.

[BP84] H. Bock and K. Plitt. “A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems”. In: *IFAC Proceedings Volumes* 17.2 (1984). 9th IFAC World Congress, pp. 1603–1608.

[BS72] R. H. Bartels and G. W. Stewart. “Solution of the Matrix Equation AX + XB = C”. In: *Communications of the ACM* 15.9 (1972), pp. 820–826.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

[Byr+08] R. H. Byrd, J. Nocedal, and R. A. Waltz. "Steering exact penalty methods for nonlinear programming". In: *Optimization Methods & Software* 23.2 (2008), pp. 197–213.

[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+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.

[Che+00] W.-H. Chen, J. O'Reilly, and D. Ballance. "Model predictive control of nonlinear systems: Computational burden and stability". In: *IEE Proceedings - Control Theory and Applications* 147.4 (2000), pp. 387–394.

[Che+20] Y. Chen, N. Scarabottolo, M. Bruschetta, and A. Beghi. "Efficient move blocking strategy for multiple shooting-based non-linear model predictive control". In: *IET Control Theory and Applications* 14.2 (2020), pp. 343–351.

[CR80] C. R. Cutler and B. L. Ramaker. "Dynamic matrix control - a computer control algorithm". In: *Joint Automatic Control Conference*. 1980.

[DB94] N. M. C. De Oliveira and L. T. Biegler. "Constraint Handing and Stability Properties of Model-Predictive Control". In: *AIChE Journal* 40.7 (1994), pp. 1138–1155.

[DeN+98] G. De Nicolao, L. Magni, and R. Scattolini. "Stabilizing Receding-Horizon Control of Nonlinear Time-Varying Systems". In: *IEEE Transactions on Automatic Control* 43.7 (1998), pp. 1030–1036.

[Die+02] M. Diehl, H. 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+05a] M. Diehl, R. Findeisen, H. Bock, F. Allgöwer, and J. Schlöder. "Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control". In: *IEE Proceedings - Control Theory and Applications* 152.3 (2005), pp. 296–308.

[Die+05b] 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.

[DM21] R. Dyrska and M. Mönnigmann. "Accelerating Nonlinear Model Predictive Control by Constraint Removal". In: *IFAC-PapersOnLine* 54.6 (2021). 7th IFAC Conference on Nonlinear Model Predictive Control (NMPC), pp. 278–283.

[Dun+08] D. D. Dunlap, E. G. Collins Jr., and C. V. Caldwell. "Sampling Based Model Predictive Control with Application to Autonomous Vehicle Guidance". In: *Florida Conference on Recent Advances in Robotics*. 2008.

[Dun+10] D. D. Dunlap, C. V. Caldwell, and E. G. Collins Jr. "Nonlinear Model Predictive Control using sampling and goal-directed optimization". In: *IEEE International Conference on Control Applications (CCA)*. 2010, pp. 1349–1356.

[ECPa] ECP. *Fact Sheet: Model 220 Industrial Plant Emulator*. `http://www.ecpsystems.com/controls_emulator.htm`. Visited on 2021-11-23.

[ECPb] ECP. *Manual: Model 220 Industrial Plant Emulator*. `http://users.encs.concordia.ca/home/r/realtime/elec372/docs/industrialEmulator.pdf`. Visited on 2021-11-23.

[Eng+19] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen. "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: *Optimization and Engineering* 20.3 (2019), pp. 769–809.

[Ewa+03] R. Ewald, J. Hutter, D. Kretz, F. Liedhegener, W. Schenkel, A. Schmitt, and M. Reik. *Proportional- und Servoventiltechnik: Der Hydraulik Trainer, Band 2*. 2nd ed. Vol. 2. English version exists. Würzburg: Bosch Rexroth AG, Drive & Control Academy, 2003.

[FA04] R. Findeisen and F. Allgöwer. "Computational Delay in Nonlinear Model Predictive Control". In: *IFAC Proceedings Volumes* 37.1 (2004). 7th International Symposium on Advanced Control of Chemical Processes (ADCHEM), pp. 427–432.

[Far+20] H. Farooqi, L. Fagiano, P. Colaneri, and D. Barlini. "Shrinking horizon parametrized predictive control with application to energy-efficient train operation". In: *Automatica* 112 (2020), p. 108635.

[Fee+98] B. Feeny, A. Guran, N. Hinrichs, and K. Popp. "A Historical Review on Dry Friction and Stick-Slip Phenomena". In: *Applied Mechanics Review* 51.5 (1998), pp. 321–341.

[Fer+17] H. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. Jerez, G. Stathopoulos, and C. Jones. "Embedded Optimization Methods for Industrial Automatic Control". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 13194–13209.

[Fia83] F. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Amsterdam, Boston: Academic Press, 1983.

[Fin+03] R. Findeisen, L. Imsland, F. Allgower, and B. Foss. "Stability Conditions for Observer Based Output Feedback Stabilization with Nonlinear Model Predictive Control". In: *IEEE International Conference on Decision and Control (CDC)*. 2003, pp. 1425–1430.

[Fin+18] R. Findeisen, K. Graichen, and M. Mönnigmann. "Eingebettete Optimierung in der Regelungstechnik – Grundlagen und Herausforderungen". In: *at - Automatisierungstechnik* 66.11 (2018). In German, pp. 877–902.

[Fle00] R. Fletcher. *Practical Methods of Optimization*. 2nd ed. Note: 1st ed. from 1987. New York: John Wiley & Sons, 2000.

[Fon01] F. A. Fontes. "A General Framework to Design Stabilizing Nonlinear Model Predictive Controllers". In: *Systems & Control Letters* 42.2 (2001), pp. 127–143.

[Fox+97] D. Fox, W. Burgard, and S. Thrun. "The Dynamic Window Approach to Collision Avoidance". In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.

[Fri+16] G. Frison, D. Kouzoupis, J. B. Jorgensen, 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.

[GA14] G. Goebel and F. Allgöwer. "State dependent parametrizations for nonlinear MPC". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 1005–1010.

[Gar+89] C. E. García, D. M. Prett, and M. Morari. "Model predictive control: Theory and practice - A survey". In: *Automatica* 25.3 (1989), pp. 335–348.

[GI07] R. Gondhalekar and J. Imura. "Strong Feasibility in Input-Move-Blocking Model Predictive Control". In: *IFAC Proceedings Volumes* 40.12 (2007), pp. 816–821.

[Gis14] P. Giselsson. "Improved Fast Dual Gradient Methods for Embedded Model Predictive Control". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 2303–2309.

[GK10] K. Graichen and A. Kugi. "Stability and Incremental Improvement of Suboptimal MPC Without Terminal Constraints". In: *IEEE Transactions on Automatic Control* 55.11 (2010), pp. 2576–2580.

[Gon+09] R. Gondhalekar, J. Imura, and K. Kashima. "Controlled invariant feasibility – A general approach to enforcing strong feasibility in MPC applied to move-blocking". In: *Automatica* 45.12 (2009), pp. 2869–2875.

[Goo+10] G. C. Goodwin, D. Q. Mayne, K.-Y. Chen, C. Coates, G. Mirzaeva, and D. E. Quevedo. "An introduction to the control of switching electronic systems". In: *Annual Reviews in Control* 34.2 (2010), pp. 209–220.

[GP17] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. 2nd ed. Communications and Control Engineering. Note: 1st ed. from 2011. Cham: Springer, 2017.

[GQ14] T. Geyer and D. E. Quevedo. “Multistep Finite Control Set Model Predictive Control for Power Electronics”. In: *IEEE Transactions on Power Electronics* 29.12 (2014), pp. 6836–6846.

[GR08] L. Grune and A. Rantzer. “On the Infinite Horizon Performance of Receding Horizon Controllers”. In: *IEEE Transactions on Automatic Control* 53.9 (2008), pp. 2100–2111.

[GR20] O. J. Gonzalez Villarreal and A. Rossiter. “A Shifting Strategy for Efficient Block-based Nonlinear Model Predictive Control Using Real-Time Iterations”. In: *IET Control Theory & Applications* 14.6 (2020), pp. 865–877.

[Gri+04] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel. “Examples when nonlinear model predictive control is nonrobust”. In: *Automatica* 40.10 (2004), pp. 1729–1738.

[Gri+05] G. Grimm, M. Messina, S. Tuna, and A. Teel. “Model Predictive Control: For Want of a Local Control Lyapunov function, All is Not Lost”. In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 546–558.

[Gri+07] G. Grimm, M. Messina, S. Tuna, and A. Teel. “Nominally Robust Model Predictive Control With State Constraints”. In: *IEEE Transactions on Automatic Control* 52.10 (2007), pp. 1856–1870.

[Grü09] L. Grüne. “Analysis and Design of Unconstrained Nonlinear MPC Schemes for Finite and Infinite Dimensional Systems”. In: *SIAM Journal on Control and Optimization* 48.2 (2009), pp. 1206–1228.

[Grü12] L. Grüne. “NMPC without terminal constraints”. In: *IFAC Proceedings Volumes* 45.17 (2012). 4th IFAC Conference on Nonlinear Model Predictive Control, pp. 1–13.

[Hal09] J. K. Hale. *Ordinary Differential Equations*. Dover ed. Note: 1st ed. from 1969, Wiley, New York. Mineola, New York: Dover Publications, 2009.

[Ham95] S. P. Hampson. “Nonlinear Model Predictive Control of a Hydraulic Actuator: Theory and Implementation”. PhD thesis. University of Canterbury, 1995.

[Har+96] F. Harashima, M. Tomizuka, and T. Fukuda. “Mechatronics - "What Is It, Why, and How?" An editorial”. In: *IEEE/ASME Transactions on Mechatronics* 1.1 (1996), pp. 1–4.

[Has+17] M. S. Hasan, A. E. Hafni, and R. Kennel. “Position Control of an Electromagnetic Actuator Using Model Predictive Control”. In: *IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*. 2017, pp. 37–41.

[HK17] P. Hyatt and M. D. Killpack. “Real-Time Evolutionary Model Predictive Control Using a Graphics Processing Unit”. In: *IEEE International Conference on Humanoid Robotics (Humanoids)*. 2017, pp. 569–576.

[Hol+20] J. Holaza, J. Oravec, M. Kvasnica, R. Dyrska, M. Mönnigmann, and M. Fikar. “Accelerating Explicit Model Predictive Control by Constraint Sorting”. In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 11356–11361.

[Hom+18] A. Homann, M. Buss, M. Keller, K.-H. Glander, and T. Bertram. “Multi Stage Model Predictive Trajectory Set Approach for Collision Avoidance”. In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 945–950.

[Hom+19] A. Homann, C. Lienke, M. Keller, M. Buss, M. Mohamed, and T. Bertram. “Sampling-based Trajectory Planning and Control for a Collision Avoidance System”. In: *IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 2956–2962.

[Hou+10] 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 (2010), pp. 298–312.

[Jer+11] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides. “A Condensed and Sparse QP Formulation for Predictive Control”. In: *Conference on Decision and Control and European Control Conference (CDC)*. IEEE, 2011, pp. 5217–5222.

[JH05] A. Jadbabaie and J. Hauser. “On the Stability of Receding Horizon Control With a General Terminal Cost”. In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 674–678.

[JK12] M. Jelali and A. Kroll. *Hydraulic Servo-systems: Modelling, Identification and Control*. 1st ed. London: Springer, 2012.

[Joo+11] A. Joos, M. Müller, D. Baumgärtner, W. Fichter, and F. Allgöwer. “Nonlinear Predictive Control Based on Time-Domain Simulation for Automatic Landing”. In: *AIAA Guidance, Navigation, and Control Conference*. 2011.

[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.

[KA21] J. Köhler and F. Allgöwer. “Stability and performance in MPC using a finite-tail cost”. In: *IFAC-PapersOnLine* 54.6 (2021). 7th IFAC Conference on Nonlinear Model Predictive Control (NMPC), pp. 166–171.

[Kal+12] E. Kallenbach, R. Eick, P. Quendt, T. Ströhla, K. Feindt, M. Kallenbach, and O. Radler. *Elektromagnete: Grundlagen, Berechnung, Entwurf und Anwendung*. 4th ed. In German. Wiesbaden: Vieweg+Teubner Verlag, 2012.

[Kel+15] M. Keller, C. Hass, A. Seewald, and T. Bertram. "A Model Predictive Approach to Emergency Maneuvers in Critical Traffic Situations". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2015, pp. 369–374.

[Kel17] M. Keller. "Trajektorienplanung zur Kollisionsvermeidung im Straßenverkehr". In German, published in: Fortschritt-Berichte VDI: Reihe 12, Verkehrstechnik/Fahrzeugtechnik, VDI-Verlag, Düsseldorf. Dissertation. TU Dortmund University, 2017.

[Ker00] E. C. Kerrigan. "Robust Constraint Satisfaction: Invariant Sets and Predictive Control". PhD thesis. University of Cambridge, 2000.

[KG14] B. Kapernick and K. Graichen. "The Gradient Based Nonlinear Model Predictive Control Software GRAMPC". In: *European Control Conference (ECC)*. 2014, pp. 1170–1175.

[KG20] P. Karamanakos and T. Geyer. "Guidelines for the Design of Finite Control Set Model Predictive Controllers". In: *IEEE Transactions on Power Electronics* 35.7 (2020), pp. 7434–7450.

[KG88] S. S. Keerthi and E. G. Gilbert. "Optimal Infinite-Horizon Feedback Laws for a General Class of Constrained Discrete-Time Systems: Stability and Moving-Horizon Approximations". In: *Journal of Optimization Theory and Applications* 57.2 (1988), pp. 265–293.

[KM00a] E. Kerrigan and J. Maciejowski. "Invariant Sets for Constrained Nonlinear Discrete-time Systems with Application to Feasibility in Model Predictive Control". In: *IEEE Conference on Decision and Control (CDC)*. 2000, pp. 4951–4956.

[KM00b] E. C. Kerrigan and J. M. Maciejowski. "Soft constraints and exact penalty functions in model predictive control". In: *UKACC International Conference on Control*. 2000.

[Köh+20] J. Köhler, M. A. Müller, and F. Allgöwer. "A Nonlinear Model Predictive Control Framework Using Reference Generic Terminal Ingredients". In: *IEEE Transactions on Automatic Control* 65.8 (2020), pp. 3576–3583.

[Kou+15a] D. Kouzoupis, H. J. Ferreau, H. Peyrl, 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-PapersOnLine* 48.23 (2015). 5th IFAC Conference on Nonlinear Model Predictive Control (NMPC), pp. 26–31.

[Kra85] D. Kraft. "On Converting Optimal Control Problems into Nonlinear Programming Problems". In: *Computational Mathematical Programming*. Ed. by K. Schittkowski. 1st ed. Vol. 15. Nato ASI Series, Series F:

Computer and System Sciences. Berlin, Heidelberg: Springer-Verlag, 1985. Chap. 9, pp. 261–280.

[Kre+07] J. Krettek, D. Schauten, F. Hoffmann, and T. Bertram. "Evolutionary Hardware-in-the-Loop Optimization of a Controller for Cascaded Hydraulic Valves". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2007, pp. 1–6.

[Kre+09] J. Krettek, J. Braun, F. Hoffmann, T. Bertram, T. Ewald, H.-G. Schubert, and H. Lausch. "Interactive Evolutionary Multiobjective Optimization for Hydraulic Valve Controller Parameters". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2009, pp. 816–821.

[Kri+15] C. Krimpmann, G. Schoppel, I. Glowatzky, and T. Bertram. "Performance Evaluation of Nonlinear Surfaces for Sliding Mode Control of a Hydraulic Valve". In: *IEEE Conference on Control Applications (CCA)*. 2015, pp. 822–827.

[Kri+16b] C. Krimpmann, G. Schoppel, I. Glowatzky, and T. Bertram. "Lyapunov-Based Self-Tuning of Sliding Surfaces – Methodology And Application to Hydraulic Valves". In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. 2016, pp. 457–462.

[Kri18] C. Krimpmann. "Sliding Mode Control of Mechatronic Systems: From robust control towards self-tuning control with experimental investigations on hydraulic valves". Published in: Reihe Regelungstechnik, Dissertationsverlag Dr. Hut, München. Dissertation. TU Dortmund University, 2018.

[Kuf+14] D. K. M. Kufoalor, S. Richter, L. Imsland, T. A. Johansen, M. Morari, and G. O. Eikrem. "Embedded Model Predictive Control on a PLC Using a Primal-Dual First-Order Method for a Subsea Separation Process". In: *Mediterranean Conference on Control and Automation (MED)*. 2014, pp. 368–373.

[Kuf+15] D. K. M. Kufoalor, B. J. T. Binder, H. J. Ferreau, L. Imsland, T. A. Johansen, 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. "g^2o: A general Framework for Graph Optimization". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011.

[Lau90] H. Lausch. *Digitale Regelung hydraulischer Antriebe mittels pulsbreitenmoduliert angesteuerter Proportionalventile*. Vol. 213. Fortschritt-Berichte VDI: Reihe 8, Meß-, Steuerungs- und Regelungstechnik. In German. Düsseldorf: VDI-Verlag, 1990.

[LaV06] S. M. LaValle. *Planning Algorithms*. 1st ed. Cambridge: Cambridge University Press, 2006.

[Lee+95] J. H. Lee, Y. Chikkula, Z. Yu, and J. C. Kantor. "Improving computational efficiency of model predictive control algorithm using wavelet transformation". In: *International Journal of Control* 61.4 (1995), pp. 859–883.

[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". In: *Computers & Chemical Engineering* 27.2 (2003), pp. 157–166.

[Lim+06] D. Limon, T. Alamo, F. Salas, and E. Camacho. "On the Stability of Constrained MPC Without Terminal Constraint". In: *IEEE Transactions on Automatic Control* 51.5 (2006), pp. 832–836.

[Lim+18] D. Limon, A. Ferramosca, I. Alvarado, and T. Alamo. "Nonlinear MPC for Tracking Piece-Wise Constant Reference Signals". In: *IEEE Transactions on Automatic Control* 63.11 (2018), pp. 3735–3750.

[LM67] E. B. Lee and L. Markus. *Foundations of optimal control theory*. 1st ed. SIAM series in applied mathematics. New York: Wiley, 1967.

[Lon+11] S. Longo, E. C. Kerrigan, K. V. Ling, and G. A. Constantinides. "Parallel Move Blocking Model Predictive Control". In: *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. 2011, pp. 1239–1244.

[Lun16] J. Lunze. *Regelungstechnik 2 - Mehrgrössensysteme, Digitale Regelung*. 9th ed. In German. Berlin, Heidelberg: Springer Vieweg, 2016.

[Mac02] J. M. Maciejowski. *Predictive Control with Constraints*. London: Pearson Education Limited, Prentice Hall, 2002.

[Mae+09] U. Maeder, F. Borrelli, and M. Morari. "Linear offset-free Model Predictive Control". In: *Automatica* 45.10 (2009), pp. 2214–2222.

[Mag+01] L. Magni, G. De Nicolao, L. Magnani, and R. Scattolini. "A stabilizing model-based predictive control algorithm for nonlinear systems". In: *Automatica* 37.9 (2001), pp. 1351–1362.

[Mak+15a] A. Makarow, J. Braun, C. Krimpmann, T. Bertram, G. Schoppel, and I. Glowatzky. "Regelungstechnische Modellierung eines hydraulischen Wegeventils [Control engineering modeling of a hydraulic directional control valve]". In: *VDI/VDE Mechatroniktagung*. Paper in German, abstract review only. 2015, pp. 203–208.

[Mak+15c] A. Makarow, J. Braun, C. Krimpmann, T. Bertram, G. Schoppel, and I. Glowatzky. "System Model of a Directional Control Valve for Control Applications". In: *The Fourteenth Scandinavian International Conference on Fluid Power (SICFP)*. 2015, pp. 542–553.

[Mak+17] 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.

[Mak+18a] A. Makarow, J. Braun, M. Keller, T. Bertram, G. Schoppel, and I. Glowatzky. "A Holistic Approach to the System Optimization of a Proportional Valve". In: *International Fluid Power Conference (IFK)*. Abstract review only. 2018.

[Mak+18b] A. Makarow, J. Braun, C. Rösmann, and T. Bertram. "Cascaded Evolutionary Multi-Objective System Optimization for a Proportional Directional Control Valve". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2018, pp. 1408–1413.

[Mak+18c] 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.

[Mak+18d] 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.

[Mak+18e] A. Makarow, C. Rösmann, M. Keller, and T. Bertram. "Vergleich der modellprädiktiven Trajektorienscharregelung mit konventionellen Regelungsverfahren [Comparison of model predictive trajectory set control with conventional control approaches]". In: *International Federation for the Promotion of Mechanism and Machine Science D-A-CH Conference (IFToMM D-A-CH)*. Short paper in German. 2018.

[Mak+20] A. Makarow, C. Rösmann, and T. Bertram. "Single Degree of Freedom Model Predictive Control with Variable Horizon". In: *American Control Conference (ACC)*. 2020, pp. 2419–2425.

[Mak+21] A. Makarow, C. Rösmann, and T. Bertram. "Suboptimale Modellprädiktive Regelung mit einem Freiheitsgrad für unterlagerte Regelkreise [Model predictive control with single degree of freedom for low-level control]". In: *at - Automatisierungstechnik* 69.7 (2021). Special Issue: GMA FA 1.40. In German, pp. 597–607.

[Mak+22] A. Makarow, C. Rösmann, and T. Bertram. "Suboptimal nonlinear model predictive control with input move-blocking". In: *International Journal of Control* (2022), pp. 1–10. Preprint from 2021: arXiv:2109.12355 [eess.SY].

[May+00] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6 (2000), pp. 789–814.

[May13] D. Mayne. "An apologia for stabilising terminal conditions in model predictive control". In: *International Journal of Control* 86.11 (2013), pp. 2090–2095.

[MB02] K. R. Muske and T. A. Badgwell. "Disturbance modeling for offset-free linear model predictive control". In: *Journal of Process Control* 12.5 (2002), pp. 617–632.

[MK11] M. Mönnigmann and M. Kastsian. "Fast explicit MPC with multiway trees". In: *IFAC Proceedings Volumes* 44.1 (2011). 18th IFAC World Congress, pp. 1356–1361.

[ML99] M. Morari and J. H. Lee. "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667–682.

[MM12] M. Morari and U. Maeder. "Nonlinear offset-free model predictive control". In: *Automatica* 48.9 (2012), pp. 2059–2067.

[MM90] D. Q. Mayne and H. Michalska. "Receding horizon control of nonlinear systems". In: *IEEE Transactions on Automatic Control* 35.7 (1990), pp. 814–824.

[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.

[Mor+12] J. L. Morales, J. Nocedal, and Y. Wu. "A Sequential Quadratic Programming Algorithm with an Additional Equality Constrained Phase". In: *Journal of Numerical Analysis* 32.2 (2012), pp. 553–579.

[Nic+01] B. Nicolaus, H. Kiendl, W. Blumendeller, and U. Schwane. "Evolutionary Optimization of an Industrial Hydraulic Valve with the help of a Fuzzy Performance-Index". In: *IEEE International Conference on Fuzzy Systems*. 2001, pp. 139–142.

[NK15] I. Necoara and S. Kvamme. "DuQuad: A toolbox for solving convex quadratic programs using dual (augmented) first order algorithms". In: *IEEE Conference on Decision and Control (CDC)*. 2015, pp. 2043–2048.

[Noc80] J. Nocedal. "Updating Quasi-Newton Matrices With Limited Storage". In: *Mathematics of Computation* 35.151 (1980), pp. 773–782.

[NT04] D. Nesic 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. Vol. 12. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006.

[Ott04] B. Ottersbach. "Dezentrale Strategieelemente für Evolutionsstrategien und Anwendung zum Reglerentwurf". In German. Dissertation. TU Dortmund University, 2004.

[OW14] C.-J. Ong and Z. Wang. "Reducing variables in Model Predictive Control of linear system with disturbances using singular value decomposition". In: *Systems & Control Letters* 71 (2014), pp. 62–68.

[Pan+11] G. Pannocchia, J. B. Rawlings, and S. J. Wright. "Conditions under which suboptimal nonlinear MPC is inherently robust". In: *Systems & Control Letters* 60.9 (2011), pp. 747–755.

[Pan15] G. Pannocchia. "Offset-free tracking MPC: A tutorial review and comparison of different formulations". In: *European Control Conference (ECC)*. 2015, pp. 527–532.

[Per+88] A. L. Peressini, F. E. Sullivan, and J. J. Uhl. *The Mathematics of Nonlinear Programming*. 1st ed. Note: Used softcover reprint from 2012. New York: Springer, 1988.

[Pic+03] B. Picasso, S. Pancanti, A. Bemporad, and A. Bicchi. "Receding-Horizon Control of LTI Systems with Quantized Inputs". In: *IFAC Proceedings Volumes* 36.6 (2003). IFAC Conference on Analysis and Design of Hybrid Systems, pp. 259–264.

[Pol26] 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.

[PR03] G. Pannocchia and J. B. Rawlings. "Disturbance Models for Offset-Free Model-Predictive Control". In: *AIChE Journal* 49.2 (2003), pp. 426–437.

[PZ95] T. Parisini and R. Zoppoli. "A Receding-horizon Regulator for Nonlinear Systems and a Neural Approximation". In: *Automatica* 31.10 (1995), pp. 1443–1451.

[QB03] S. Qin and T. A. Badgwell. "A survey of industrial model predictive control technology". In: *Control Engineering Practice* 11.7 (2003), pp. 733–764.

[Rat+18] K. M. M. Rathai, M. Alamir, O. Sename, and R. Tang. "A Parameterized NMPC Scheme for Embedded Control of Semi-active Suspension System". In: *IFAC-PapersOnLine* 51.20 (2018). 6th IFAC Conference on Nonlinear Model Predictive Control (NMPC), pp. 301–306.

[Rat+19] K. M. M. Rathai, O. Sename, and M. Alamir. "GPU-Based Parameterized NMPC Scheme for Control of Half Car Vehicle With Semi-Active Suspension System". In: *IEEE Control Systems Letters* 3.3 (2019), pp. 631–636.

[Raw+20] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd ed. Note: 1st ed. from 2009. Santa Barbara: Nob Hill Publishing, 2020.

[RH06] A. Richards and J. P. How. "Robust variable horizon model predictive control for vehicle maneuvering". In: *International Journal of Robust and Nonlinear Control* 16.7 (2006), pp. 333–351.

[Ric+77] J. Richalet, A. Rault, J. Testud, and J. Papon. "Model algorithmic control of industrial processes". In: *IFAC Proceedings Volumes* 10.16 (1977), pp. 103–120.

[Ric+78] J. Richalet, A. Rault, J. Testud, and J. Papon. "Model predictive heuristic control". In: *Automatica* 14.5 (1978), pp. 413–428.

[Ric85] N. L. Ricker. "Use of Quadratic Programming for Constrained Internal Model Control". In: *Industrial Engineering Chemistry Process Design and Development* 24.4 (1985), pp. 925–936.

[RK20] C. Rösmann and M. Krämer. *Control-Box RST*. `https://github.com/rst-tu-dortmund/control_box_rst`. Visited on 2021-12-17. 2020.

[RM93] J. Rawlings and K. Muske. "The Stability of Constrained Receding Horizon Control". In: *IEEE Transactions on Automatic Control* 38.10 (1993), pp. 1512–1516.

[Rod+13] J. Rodriguez, M. P. Kazmierkowski, J. R. Espinoza, P. Zanchetta, H. Abu-Rub, H. A. Young, and C. A. Rojas. "State of the Art of Finite Control Set Model Predictive Control in Power Electronics". In: *IEEE Transactions on Industrial Informatics* 9.2 (2013), pp. 1003–1016.

[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.

[Rös+21b] C. Rösmann, A. Makarow, and T. Bertram. "Stabilising quasi-time-optimal nonlinear model predictive control with variable discretisation". In: *International Journal of Control* (2021), pp. 1–13.

[Rös19] C. Rösmann. "Time-Optimal Nonlinear Model Predictive Control: Direct Transcription Methods with Variable Discretization and Structural Sparsity Exploitation". Dissertation. TU Dortmund University, 2019.

[RR17a] J. B. Rawlings and M. J. Risbeck. "Model predictive control with discrete actuators: Theory and application". In: *Automatica* 78 (2017), pp. 258–265.

[RR17b] J. B. Rawlings and M. J. Risbeck. *On the equivalence between statements with epsilon-delta and K-functions*. Tech. rep. Madison, WI 53706, USA:

Deptartment of Chemical and Biological Engineering, University of Wisconsin-Madison, 2017.

[RW08] J. A. Rossiter and L. Wang. “Exploiting Laguerre functions to improve the feasibility/performance compromise in MPC”. In: *IEEE Conference on Decision and Control (CDC)*. 2008, pp. 4737–4742.

[Sas99] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Interdisciplinary Applied Mathematics. New York: Springer, 1999.

[SB93] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. 2nd ed. Texts in Applied Mathematics. Note: Translated by R. Bartels, W. Gautschi, and C. Witzgall. New York: Springer, 1993.

[SB94] C. Schmid and L. Biegler. “Quadratic programming methods for reduced hessian SQP”. In: *Computers & Chemical Engineering* 18.9 (1994), pp. 817–832.

[Sco+99] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. “Suboptimal Model Predictive Control (Feasibility Implies Stability)”. In: *IEEE Transactions on Automatic Control* 44.3 (1999), pp. 648–654.

[She12] R. C. Shekhar. “Variable Horizon Model Predictive Control: Robustness & Optimality”. PhD thesis. University of Cambridge, 2012.

[Sim16] V. Simoncini. “Computational Methods for Linear Matrix Equations”. In: *SIAM Review* 58.3 (2016), pp. 377–441.

[SM12] R. C. Shekhar and J. M. Maciejowski. “Robust variable horizon MPC with move blocking”. In: *Systems & Control Letters* 61.4 (2012), pp. 587–594.

[SM15] R. C. Shekhar and C. Manzie. “Optimal move blocking strategies for model predictive control”. In: *Automatica* 61 (2015), pp. 27–34.

[Son+21] S. H. Son, B. J. Park, T. H. Oh, J. W. Kim, and J. M. Lee. “Move blocked model predictive control with guaranteed stability and improved optimality using linear interpolation of base sequences”. In: *International Journal of Control* 94.11 (2021), pp. 3213–3225.

[SR99] P. O. M. Scokaert and J. B. Rawlings. “Feasibility Issues in Linear Model Predictive Control”. In: *AIChE Journal* 45.8 (1999), pp. 1649–1659.

[Ste+17] B. Stellato, T. Geyer, and P. J. Goulart. “High-Speed Finite Control Set Model Predictive Control for Power Electronics”. In: *IEEE Transactions on Power Electronics* 32.5 (2017), pp. 4007–4020.

[Ste+20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. “OSQP: an operator splitting solver for quadratic programs”. In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672.

[Tat07] P. Tatjewski. *Advanced Control of Industrial Processes: Structures and Algorithms*. Advances in Industrial Control. London: Springer, 2007.

[TJ02] P. Tøndel and T. A. Johansen. "Comlexity Reduction in Explicit Linear Lodel Predicitve Control". In: *IFAC Proceedings Volumes* 35.1 (2002). 15th IFAC World Congress, pp. 189–194.

[Tøn+03] P. Tøndel, T. Johansen, and A. Bemporad. "Evaluation of piecewise affine control via binary search tree". In: *Automatica* 39.5 (2003), pp. 945–950.

[Tun+06] S. Tuna, M. Messina, and A. Teel. "Shorter horizons for model predictive control". In: *American Control Conference (ACC)*. 2006, pp. 863–868.

[Ver+16] R. Verschueren, N. van Duijkeren, R. Quirynen, and M. Diehl. "Exploiting Convexity in Direct Optimal Control: a Sequential Convex Quadratic Programming Method". In: *IEEE Conference on Decision and Control (CDC)*. 2016, pp. 1099–1104.

[Ver+18] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl. "Towards a modular software package for embedded optimization". In: *IFAC-PapersOnLine* 51.20 (2018). 6th IFAC Conference on Nonlinear Model Predictive Control (NMPC), pp. 374–380.

[Wan+14] Y. Wang, B. O'Donoghue, and S. Boyd. "Approximate Dynamic Programming via Iterated Bellman Inequalities". In: *International Journal of Robust and Nonlinear Control* 25.10 (2014), pp. 1472–1496.

[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.

[WB10] Y. Wang and S. Boyd. "Fast Model Predictive Control Using Online Optimization". In: *IEEE Transactions on Control Systems Technology* 18.2 (2010), pp. 267–278.

[Wig03] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. 2nd ed. Springer New York, 2003.

[Wil+16] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. "Aggressive Driving with Model Predictive Path Integral Control". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1433–1440.

[Wor12] K. Worthmann. "Stability Analysis of Unconstrained Receding Horizon Control Schemes". Dissertation. Universität Bayreuth, 2012.

[Xu+20] B. Xu, J. Shen, S. Liu, Q. Su, and J. Zhang. "Research and Development of Electro-hydraulic Control Valves Oriented to Industry 4.0: A Review". In: *Chinese Journal of Mechanical Engineering* 33.1 (2020).

[YB16] M. Yu and L. T. Biegler. "A Stable and Robust NMPC Strategy with Reduced Models and Nonuniform Grids". In: *IFAC-PapersOnLine* 49.7

(2016). 11th IFAC Symposium on Dynamics and Control of Process Systems Including Biosystems (DYCOPS-CAB), pp. 31–36.

[ZA98] A. Zheng and F. Allgöwer. "Towards a Practical Nonlinear Predictive Control Algorithm with Guaranteed Stability for Large-Scale Systems". In: *American Control Conference (ACC)*. 1998, pp. 2534–2538.

[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.

[ZB09] V. M. Zavala and L. T. Biegler. "Nonlinear Programming Strategies for State Estimation and Model Predictive Control". In: *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Ed. by L. Magni, D. Raimondo, and F. Allgöwer. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, 2009, pp. 419–432.

[ZC92] E. Zafiriou and H.-W. Chiou. *On the Effect of Constraint Softening on the Stability and Performance of Model Predictive Control*. Tech. rep. Prepared for presentation at the Annual AIChE Meeting 1992. Deptartment of Chemical Engineering and Institute for Systems Research, University of Maryland, 1992.

[Zei+14] M. N. Zeilinger, M. Morari, and C. N. Jones. "Soft Constrained Model Predictive Control With Robust Stability Guarantees". In: *IEEE Transactions on Automatic Control* 59.5 (2014), pp. 1190–1202.

[ZM95] A. Zheng and M. Morari. "Stability of Model Predictive Control with Mixed Constraints". In: *IEEE Transactions on Automatic Control* 40.10 (1995), pp. 1818–1823.

Related Peer-Reviewed Publications

1. A. Makarow, C. Rösmann, and T. Bertram. “Suboptimal nonlinear model predictive control with input move-blocking”. In: *International Journal of Control* (2022), pp. 1–10. Preprint from 2021: arXiv:2109.12355 [eess.SY].
2. A. Makarow, C. Rösmann, and T. Bertram. “Suboptimale Modellprädiktive Regelung mit einem Freiheitsgrad für unterlagerte Regelkreise [Model predictive control with single degree of freedom for low-level control]”. In: *at - Automatisierungstechnik* 69.7 (2021). Special Issue: GMA FA 1.40. In German, pp. 597–607.
3. A. Makarow, C. Rösmann, and T. Bertram. “Single Degree of Freedom Model Predictive Control with Variable Horizon”. In: *American Control Conference (ACC)*. 2020, pp. 2419–2425.
4. A. Makarow, J. Braun, M. Keller, T. Bertram, G. Schoppel, and I. Glowatzky. “A Holistic Approach to the System Optimization of a Proportional Valve”. In: *International Fluid Power Conference (IFK)*. Abstract review only. 2018.
5. A. Makarow, J. Braun, C. Rösmann, and T. Bertram. “Cascaded Evolutionary Multi-Objective System Optimization for a Proportional Directional Control Valve”. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2018, pp. 1408–1413.
6. 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.
7. 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.
8. A. Makarow, C. Rösmann, M. Keller, and T. Bertram. “Vergleich der modellprädiktiven Trajektorienscharregelung mit konventionellen Regelungsverfahren [Comparison of model predictive trajectory set control with conventional control approaches]”. In: *International Federation for the Promotion of Mechanism and Machine Science D-A-CH Conference (IFToMM D-A-CH)*. Short paper in German. 2018.
9. 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.

Additional Peer-Reviewed Publications

1. C. Diehl, A. Makarow, C. Rösmann, and T. Bertram. "Time-Optimal Nonlinear Model Predictive Control for Radar-based Automated Parking". In: *IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*. 2022.

2. T. Osterburg, A. Makarow, F. Hoffmann, and T. Bertram. "Dual Model Predictive Control: Online Experimental Design for Nonlinear Systems". In: *VDI/VDE Mechatroniktagung*. Abstract review only. 2022, pp. 73–78.

3. C. Rösmann, A. Makarow, and T. Bertram. "Online Motion Planning based on Nonlinear Model Predictive Control with Non-Euclidean Rotation Groups". In: *European Control Conference (ECC)*. 2021, pp. 1583–1590.

4. C. Rösmann, A. Makarow, and T. Bertram. "Stabilising quasi-time-optimal nonlinear model predictive control with variable discretisation". In: *International Journal of Control* (2021), pp. 1–13.

5. 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.

6. C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. "Gain-Scheduling zwischen zeitoptimaler und quadratischer modellprädiktiver Regelung [Gain-scheduling between time-optimal and quadratic form based model predictive control]". In: *International Federation for the Promotion of Mechanism and Machine Science D-A-CH Conference (IFToMM D-A-CH)*. Short paper in German. 2018.

7. 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.

8. 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.

9. C. Krimpmann, A. Makarow, T. Bertram, I. Glowatzky, G. Schoppel, and H. Lausch. "Simulationsgestützte Optimierung von Gleitzustandsreglern für hydraulische Wegeventile [Simulation-based optimization of a sliding mode controller for directional valves]". In: *at - Automatisierungstechnik* 64.6 (2016). Article in German, abstract in English, pp. 443–456.

10. A. Makarow, J. Braun, C. Krimpmann, T. Bertram, G. Schoppel, and I. Glowatzky. "Regelungstechnische Modellierung eines hydraulischen Wegeventils [Control engineering modeling of a hydraulic directional control valve]". In: *VDI/VDE Mechatroniktagung*. Paper in German, abstract review only. 2015, pp. 203–208.

11. A. Makarow, J. Braun, C. Krimpmann, T. Bertram, G. Schoppel, and I. Glowatzky. “System Model of a Directional Control Valve for Control Applications”. In: *The Fourteenth Scandinavian International Conference on Fluid Power (SICFP)*. 2015, pp. 542–553.

Supervised Theses

1. K. Ramthun. “Systematische Analyse der expliziten modellprädiktiven Regelung am Beispiel eines mechatronischen Systems”. Bachelorthesis. TU Dortmund University, 2020.
2. M. Klöpper. “Design of the Model Predictive Trajectory Set Control for a Wide Operation Range of a Proportional Directional Control Valve”. Masterthesis. TU Dortmund University, 2018.
3. S. Kurzawe. “Modellprädikatives Ballancieren eines Stabs mit einem seriellen Roboterarm”. Masterthesis. TU Dortmund University, 2018.
4. M. Mokhadri. “Model Predictive Path Integral Control for Joint Space Control of a Link-Elastic Robot Arm”. Masterthesis. TU Dortmund University, 2018.
5. O. Ojekunle. “Systematische Analyse und Entwurf echtzeitfähiger modellprädiktiver Regelungen am Beispiel des inversen Pendels”. Masterthesis. TU Dortmund University, 2018.
6. A. Puzicha. “Modellprädiktive Regelung eines strukturelastischen Roboterarms im Gelenkraum”. Bachelorthesis. TU Dortmund University, 2018.
7. C. Schmickler. “Simulationsbasierte Stabilitätsuntersuchung der modellprädiktiven Trajektorienscharregelung für mechatronische Systeme”. Bachelorthesis. TU Dortmund University, 2018.
8. M. Seemann. “Systematische Analyse verschiedener Mehrgrößenregler am Beispiel eines Systems zur Mischwasserbereitung”. Bachelorthesis. TU Dortmund University, 2018.
9. J. Bahne. “Automatisches Differenzieren für die modellprädiktive Regelung”. Bachelorthesis. TU Dortmund University, 2017.
10. M. Golonka. “Systematische Analyse des PID- und des modellprädiktiven Reglerentwurfs am Beispiel eines mechatronischen Systems”. Bachelorthesis. TU Dortmund University, 2017.
11. A. Hugenroth. “Simulationsbasierte Untersuchung der modellprädiktiven Trajektorienscharreglung für ein hydraulisches Wegeventil”. Bachelorthesis. TU Dortmund University, 2016.

12. B. Meier. “Untersuchung eines lagegeberlosen Verfahrens zur Regelung von Permanentmagnetsynchronmaschinen”. Masterthesis. TU Dortmund University, 2015.

13. D. Schlattmann. “Entwurf von Konzepten zur Regelung hydraulischer Druckbegrenzungsventile”. Masterthesis. TU Dortmund University, 2015.