Ruprecht-Karls-Universität Heidelberg
Fakultät für Mathematik und Informatik

An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control

Diplomarbeit

Betreuer: Professor Dr. Dr. h. c. Hans Georg Bock

Vorgelegt von Hans Joachim Ferreau

Heidelberg, November 2006

Für meine Eltern

Zusammenfassung

Eine Echtzeit-Strategie zur Bestimmung aktiver Nebenbedingungen für das schnelle Lösen parametrischer quadratischer Programme mit Anwendungen auf die prädiktive Motorsteuerung

Beinahe jeder Algorithmus zur modellprädiktiven Regelung beruht auf der Echtzeit-Lösung konvexer quadratischer Programme. In dieser Diplomarbeit wird eine maßgeschneiderte Echtzeit-Strategie zur Bestimmung aktiver Nebenbedingungen entwickelt, um parametrische quadratische Probleme – wie sie im Rahmen der modellprädiktiven Regelung auftreten – zu lösen. Unsere Strategie nutzt die Kenntnis der Lösung des vorhergehenden quadratischen Problems unter der Annahme aus, dass sich die Menge der aktiven Nebenbedingungen von einem quadratischen Programm zum nächsten nicht wesentlich ändert. Außerdem stellen wir eine Variante vor, bei der die Rechenzeit zum Zwecke realer Echtzeit-Anwendungen begrenzt wird. Eine effiziente Implementierung der vorgeschlagenen Echtzeit-Strategie wird detailliert beschrieben und ihre Leistungsfähigkeit anhand von zwei anspruchsvollen Testbeispielen aufgezeigt. Eines davon wurde zur Steuerung eines realen Dieselmotors entworfen, bei der jedes der quadratischen Programme innerhalb weniger Millisekunden gelöst werden muss. In den vorgestellten Beispielen zeigt sich, dass unsere Echtzeit-Strategie etwa eine Größenordnung schneller als herkömmliche (Warmstart-)Algorithmen zur Lösung quadratischer Programme ist.

Schlüsselwörter: modellprädiktive Regelung, parametrische quadratische Programmierung, Echtzeit-Strategie zur Bestimmung aktiver Nebenbedingungen, Echtzeit-Optimierung, Motorsteuerung

AMS-Klassifikationen: 90C20, 34H05, 93B52, 62P30

Abstract

Nearly all algorithms for model predictive control (MPC) rely on solving convex quadratic programs in real-time. In this thesis, we develop a specially tailored online active set strategy for the fast solution of parametric quadratic programs arising in MPC. Our strategy exploits solution information of the previous quadratic program (QP) under the assumption that the set of active constraints does not change much from one QP to the next. Furthermore, we present a modification where the CPU time is limited in order to make it suitable for strict real-time applications. An efficient implementation of the proposed online active set strategy is described in detail and its performance is demonstrated with two challenging test examples. One of these was designed for controlling a real-world Diesel engine with sampling times of a few milliseconds. In these examples, our strategy turns out to be an order of magnitude faster than a standard active set QP solver (with warmstarts).

Key words: model predictive control, parametric quadratic programming, online active set strategy, real-time optimisation, engine control

AMS subject classifications: 90C20, 34H05, 93B52, 62P30

Contents

Acknowledgements

I would like to express my deep gratitude to all people who helped me while writing this thesis. First of all I thank my supervisors Professor Dr. Dr. h. c. Hans Georg Bock and Professor Dr. Moritz Diehl for intensive personal support and excellent mathematical advice. It was a great pleasure for me to share their enthusiasm in many inspiring conversations and discussions on new ideas. I also thank Dr. Johannes Schlöder, Dr. Sebastian Sager and Professsor Dr. Ekaterina Kostina for fruitful discussions on optimal control and related subjects.

Moreover, I owe many thanks to all the other members of the "Simulation and Optimization Group" (headed by Professor Dr. Dr. h. c. Hans Georg Bock and Dr. Johannes Schlöder) of the Interdisciplinary Center for Scientific Computing (IWR) in Heidelberg—among them Peter Kühl, Christian Kirches, Leonard Wirsching, Jan Albersmeyer, Andreas Potschka, Gerrit Schultz, and Tanja Binder—for many pleasant conversations on almost any topic, for sometimes "stealing" their time and, last but not least, for uncountably many cups of coffee.

During the PREDIMOT project, whose topics were closely related to most parts of this thesis, I worked together with some remarkably friendly persons: with Peter Langthaler and Peter Ortner from the Johannes Kepler Universität in Linz as well as Professore Riccardo Scattolini and Gabriele Lorini from the Politecnico di Milano. Financial support of the REGINS-PREDIMOT European project is gratefully acknowledged.

Notation

Symbols

Scalar Sets

$\mathbb{A}$	working set
$\mathbb{A}(x)$	index set of active constraints at point x
$\mathbb{F}$	working set of free variables
$\mathbb{F}(x)$	index set of free variables at point x
$\mathbb{I}$	working set complement
$\mathbb{I}(x)$	index set of inactive constraints at point x
$\mathbb{N}$	set of natural numbers (greater than 0)
$\mathbb{R}$	field of real numbers
$\mathbb{R}_{\geq 0}$	set of nonnegative real numbers
$\mathbb{R}_{>0}$	set of positive real numbers
$\mathbb{T}$	time horizon of the controlled process
$\mathbb{T}_{\mathrm{p}}$	prediction horizon
$\mathbb{X}$	working set of fixed variables
$\mathbb{X}(x)$	index set of fixed variables at point x

Vector and Matrix Sets

$\mathrm{CR}_{\mathbb{A}}$	critical region of an optimal active set $\mathbb{A}$
$\mathcal{D}$	domain of a real function
$\mathcal{F}$	feasible set of a quadratic program
$\mathcal{P}$	set of feasible parameters of a parametric quadratic program
$\mathbb{R}^n$	set of real n-dimensional vectors
$\mathbb{R}^{m \times n}$	set of real $m \times n$-dimensional matrices
$\mathcal{S}^n$	set of real symmetric $n \times n$-matrices
$\mathcal{S}^n_{\succeq 0}$	set of real symmetric positive semi-definite $n \times n$-matrices
$\mathcal{S}^n_{\succ 0}$	set of real symmetric positive definite $n \times n$-matrices

Model Predictive Control

A	system dynamics matrix (associated with process states)
B	system dynamics matrix (associated with process inputs)
c	constraint function
C	output matrix (associated with process states)
δ	sampling time
D	output matrix (associated with process inputs)
f	system dynamics ODE right hand side
g	algebraic equations function of a DAE system
l	constraint vector
M	constraint matrix (associated with process outputs)
N	constraint matrix (associated with process inputs)
n_g	number of algebraic equations of a DAE system
n_{p}	length of discrete-time prediction horizon
n_p	number of process parameters
n_u	number of process inputs
n_x	number of differential process states
n_y	number of process outputs
n_z	number of algebraic process states
$\psi(\cdot)$	Lagrange term of objective function
$\phi(\cdot)$	Mayer term of objective function
p	vector of process parameters
P	terminal penalty weight matrix
Q	objective function matrix (associated with process outputs)
R	objective function matrix (associated with process inputs)
t	time
t_{start}	start time of the controlled process
t_{end}	end time of the controlled process
t_{p}	length of prediction horizon
$u(t)$	vector of process inputs
$x(t)$	vector of differential process states
$y(t)$	vector of process outputs
$x(t)$	vector of algebraic process states

Quadratic Programs

b	constraint vector
$\underline{b_{\mathrm{B}}}$	lower bound vector
$\overline{b_{\mathrm{B}}}$	upper bound vector
$\underline{b_{\mathrm{C}}}$	lower constraints' bound vector

$\overline{b_{\mathrm{C}}}$	upper constraints' bound vector
C	active constraints matrix
g	gradient vector
G	constraint matrix
H	Hessian matrix
m	number of constraints
n	number of variables
$n_{\mathbb{A}}$	number of constraints within working set $\mathbb{A}$
n_{EC}	number of equality constraints
$n_{\mathbb{F}}$	number of free variables within working set $\mathbb{F}$
$n_{\mathbb{X}}$	number of fixed variables within working set $\mathbb{X}$
n_{Z}	dimension of restricted null space of active constraints matrix
w_0	initial value parameter vector
$x^{(k)}$	kth iterate of the primal vector
$y^{(k)}$	kth iterate of the dual vector
x^{opt}	primal solution vector
y^{opt}	dual solution vector

Algorithm

$\tilde{\cdot}$	indicates a homotopy from one QP to the next
Q	orthonormal factor of TQ factorisation of $C_{\mathbb{F}}$
R	upper triangular Cholesky factor of projected Hessian matrix
τ	homotopy parameter
$\tau_{\max}$	maximum primal-dual stepsize within current critical region
T	reverse lower triangular factor of TQ factorisation of $C_{\mathbb{F}}$
Y	matrix containing orthonormal basis of the range space of $C_{\mathbb{F}}$
Z	matrix containing orthonormal basis of the null space of $C_{\mathbb{F}}$

Test Examples

α	weighting factor for difference of end position of the free end of the chain
β	weighting factor for balls' velocities
γ	weighting factor for control action
d	spring constant
g	gravitational acceleration
L	spring's rest length
m	mass of a single ball
ξ_{wall}	wall's position along the second coordinate axis
x_{end}	desired end position of the free end of the chain

Gasoline Engine

α	actuated throttle angle
A_{th}	opening area of the throttle
C	EGR specific constant
$c_{p_{\mathrm{air}}}$	specific heat at pressure of fresh air inside the intake manifold
$c_{p_{\mathrm{egr}}}$	specific heat at exhaust gas pressure
$c_{v_{\mathrm{im}}}$	specific heat at volume of intake manifold
η_{v}	volumetric efficiency
η_{comb}	combustion efficiency
γ	specific heat ratio
Γ	engine torque
H_{v}	calorific heat of the fuel
k	throttle specific constant
k_{egr}	EGR specific constant
m_{air}	mass of fresh air inside the intake manifold
m_{egr}	mass of exhausts inside the intake manifold
N	engine rotational speed
$\mathrm{NO_x}$	NO_x emissions
p_{amb}	ambient pressure
p_{exh}	exhaust gas pressure
p_{im}	intake manifold pressure
ρ	air density
R	gas constant
R_{im}	gas constant of intake manifold
τ_{exh}	time lag of exhaust gas
T_{egr}	temperature of exhaust gas
T_{im}	temperature inside the intake manifold
u_{egr}	opening angle of EGR valve
V_{e}	engine displacement
V_{im}	volume of intake manifold
w_{e}	mass flow rate from intake manifold to cylinders
w_{egr}	mass flow rate through EGR valve
w_{fuel}	fuel mass flow rate
w_{th}	mass flow rate through throttle

Others

∞	infinity
$\forall$	for all
$\exists$	there exist

$\exists!$	there exists exactly one
$\emptyset$	empty set
2^M	power set of set M
$\square$	end of proof
$\bigcirc$	end of theorem, lemma, corollary or definition

Mathematical Expressions

Constants

$\mathbb{0}$	real matrix of appropriate dimensions with all elements zero
$\mathbb{1}$	real column vector of appropriate dimension with all components one
e_i	i-th column of the identity matrix with appropriate dimension
$O^{i,j}(\cdot)$	Givens plane rotation in the (i, j) coordinate plane
Id_n	n-dimensional identity matrix
$\mathrm{Id}_n^{\mathrm{r}}$	n-dimensional reverse identity matrix
e	base of the natural logarithm
π	twice the value of the smallest positive root of the real cosine function

Others

$[\cdot, \cdot]$	closed interval of real numbers
$(\cdot, \cdot)$	open interval of real numbers *or* two-dimensional row vector
$\cdot \stackrel{\text{def}}{=} \cdot$	defines the symbol on the left to equal the expression on the right
$\cdot \stackrel{\text{def}}{=} \cdot$	defines the symbol on the right to equal the expression on the left
$\cdot \leftarrow \cdot$	assigns the value of the variable on the left to the variable on the right
M'	transposed of matrix or vector M
M^{-1}	inverse of regular matrix M
$M^{\dagger}$	pseudoinverse of matrix M
$\lvert\cdot\rvert$	absolute value of a real number *or* cardinality of a set
$\lVert\cdot\rVert_2$	Euclidean norm of a matrix or vector
$\mathrm{im}\, M$	range space spanned by the columns of matrix M
$M^{\frac{1}{2}}$	square root of matrix M, i.e. $M^{\frac{1}{2}\prime} M^{\frac{1}{2}} = M$
$\mathrm{cond}\, M$	condition number of matrix M
$\dot{f}(t)$	first derivative of function f with respect to time t
$\ddot{f}(t)$	second derivative of function f with respect to time t
$f\big\vert_{\mathcal{X}}$	restriction of function f to set $\mathcal{X}$
$\mathcal{O}(\cdot)$	big-O notation

Abbreviations and Acronyms

Besides common expressions and SI units the following abbreviations and acronyms are used:

BDF	backward differentiation formulae
CO_2	carbon dioxide
CPU	central processing unit
DAE	differential algebraic equation
EGR	exhaust gas recirculation
HC	hydrocarbon
iff	if and only if
IVP	initial value problem
LICQ	linear independence constraint qualification
LP	linear program
KKT	Karush-Kuhn-Tucker
MAF	mass air flow
MAP	manifold absolute pressure
MPC	model predictive control
MUSCOD	multiple shooting code for direct optimal control (software package)
NLP	nonlinear program
NMPC	nonlinear model predictive control
NO_x	nitrogen oxide
`OASES`	online active set strategy (software module)
ODE	ordinary differential equation
QP	quadratic program
RHC	receding horizon control
rpm	revolutions per minute
SQP	sequential quadratic programming
$\mathrm{s.t.}$	subject to
VGT	variable geometry turbocharger
VVT	variable valve timing

Chapter 1

Introduction

Model predictive control (MPC) is an advanced control strategy which allows to determine inputs of an arbitrary process that optimise the forecasted process behaviour. These inputs, or control actions, are calculated repeatedly using a mathematical *process model* for the prediction. In doing so, the fast and reliable solution of convex *quadratic programming* problems in real-time becomes a crucial ingredient of nearly all algorithms for both linear and nonlinear model predictive control. The success of linear MPC—where just one *quadratic program (QP)* needs to be solved at each sampling instant—can even be attributed to the fact that highly efficient and reliable methods for QP solution have existed for decades, and that their computation times are much smaller than the required sampling times in typical applications. On the other hand, in nonlinear MPC algorithms, quadratic programs often arise as subproblems during the iterative nonlinear solution procedure, so that not only one, but several QPs need to be solved at each sampling instant. In most MPC algorithms, the arising QPs are treated by well-tested and efficient standard methods from optimisation.

The required *sampling time*, i.e. the time difference between two re-optimisations, strongly depends on the velocity of the process dynamics. In practice, it normally varies between some seconds or minutes, e.g. if huge distillation columns or polyethylene plants are to be controlled (cf. [25], [26] or [19]), and a few milliseconds. Very short sampling times especially arise if MPC is applied to fast mechanical systems, e.g. in the very recent field of *optimal* control applications in the automotive area. Therein, *engine control* is a particular challenge due to very fast and nonlinear dynamics, making sampling times in the order of milliseconds necessary.
When sampling times become so short that the computation times for QP solution can no longer be neglected, specialised algorithms that exploit the structure of the QPs arising in MPC problems become an interesting alternative. Basically, two approaches to fast QP solution in MPC can be distinguished:

(i) First, the explicit, or *offline QP solution*, which precomputes the QP solution for all possibly arising problem instances. This can be done quite efficiently, as shown by [8], but is limited to models with small state dimensions (below ten) and few constraints.

(ii) Second, the *online QP solution* is the classical way to treat the sequence of QPs in MPC for varying initial values.

Several QP solution methods exist, among the most prominent are *active set methods*, which come in two variants, namely *primal* [37], [39] and *dual* [45], [3] active set methods. Unfortunately, for active set methods no polynomial bound on the runtime of the algorithm can be given, as has famously been shown by Klee and Minty [56] in the context of linear programming. Furthermore, *(primal-dual) interior point methods*, cf. [91], have become a strong competitor to active set methods, and have also been proposed for use in MPC [73]. They possess relatively constant computational demands and a *polynomial* runtime guarantee can be given for them. However, interior point methods suffer from the drawback that so far no efficient warm start techniques exist.

In this thesis a new active set strategy is proposed (see also [30], [29]) that is inspired by some important observations from the field of parametric quadratic programming and can neither be classified primal nor dual. It builds on the expectation that the active set does not change much from one quadratic program to the next, but is different from conventional warm starting techniques. Our *online active set strategy* comes in two variants: while the first is just an alternative way to exactly solve the QPs arising in MPC efficiently (but without theoretical runtime limit), the second one is able to give a CPU time guarantee. This guarantee, however, comes at the expense of sometimes not solving exactly the QP that we want to solve within the given sampling time. In these circumstances—that arise e.g. after large disturbances of the controlled process—an intermediate QP that lies between the previous problem and the current one is solved, instead.
An implementation of the proposed online active set strategy, the software module `OASES`, was tested on two test examples and its performance was compared to that of existing methods for solving QPs, namely the primal active set solver `qpsol` [62] and an implementation of the explicit approach [9]. The first test example is a variant of a challenging benchmark problem (first presented in [86]) where a chain of spring connected masses is regulated back into its steady-state after a strong excitation. Second, we aim at controlling a *real-world Diesel engine* at the Institute for Design and Control of Mechatronical Systems in Linz, Austria.

The thesis is organised as follows: in Chapter 2 the required and motivating theoretical background of model predictive control, with focus on linear MPC, and parametric quadratic programming is briefly summarised. Afterwards, Chapter 3 reviews several existing and widely used methods for solving quadratic programs. Our online active set strategy, including its real-time variant, is presented in Chapter 4 which also contains a short discussion on degeneracy handling and implementation details. The mentioned test problems form the basis of a performance analysis of the proposed online active set strategy in Chapters 5 and 6. Finally, Chapter 7 is devoted to a conclusion and some ideas for future work.
The appendices comprise mathematical basics (Appendix A) and an implementation overview of the software module `OASES` (Appendix B). Ultimately, an application of fast *nonlinear* model predictive control to a gasoline engine is presented in Appendix C, which initiated the development of our online active set strategy from a practical point of view.

Chapter 2

Theoretical Background and Motivation

This chapter begins by introducing the concepts of model predictive control. Putting the focus on *linear* model predictive control naturally leads us to the description of a special optimisation problem, the so called *(parametric) quadratic program*. We show how its particular structure is exploited by the recently developed *explicit* solution approach which motivated the proposed online active set strategy.

2.1 Model Predictive Control

Main concept of *model predictive control (MPC)* is to repeatedly calculate control actions which optimise the forecasted process behaviour. The prediction is based on a mathematical *process model* leading to a so-called *open-loop optimal control problem* which is solved at each *sampling instant*. The optimised control action is applied to the system until the next sampling instant when an updated optimal control problem, incorporating the new process state, is solved. Hence, model predictive control is a *feedback* control strategy, sometimes also referred to as *receding horizon control (RHC)*.

A (continuous-time) *process model* for a time inteval $\mathbb{T} \stackrel{\text{def}}{=} [t_{\text{start}}, t_{\text{end}}] \subset \mathbb{R}$, $-\infty < t_{\text{start}} \leq t_{\text{end}} \leq \infty$, consists of

1. *process inputs*, or *controls* or *manipulated variables*, $u : \mathbb{T} \rightarrow \mathbb{R}^{n_u}$,
2. *process states*, divided into
   (a) *differential states* $x : \mathbb{T} \rightarrow \mathbb{R}^{n_x}$ and
   (b) *algebraic states* $z : \mathbb{T} \rightarrow \mathbb{R}^{n_z}$
3. *process parameters* $p \in \mathbb{R}^{n_p}$
4. *process outputs*, or *controlled variables*, $y : \mathbb{T} \rightarrow \mathbb{R}^{n_y}$,

and defines a mapping (in function spaces) from a suitable subset of process input functions[1] to the set of process output functions. This mapping is implicitly given by an initial process

[1] E.g. the set of all process input functions such that (2.1.1) has a unique solution and (2.1.2) is defined for all $t \in \mathbb{T}$.

state value and a system of *differential algebraic equations (DAE)*

$$\begin{aligned} x(t_{\text{start}}) &= w_0\,, && (2.1.1a)\\ \dot{x}(t) &= f\left(t, x(t), z(t), u(t), p\right) \quad \forall\, t \in \mathbb{T}\,, && (2.1.1b)\\ \mathbb{0} &= g\left(t, x(t), z(t), u(t), p\right) \quad \forall\, t \in \mathbb{T}\,, && (2.1.1c) \end{aligned}$$

as well as

$$y(t) \stackrel{\text{def}}{=} \hat{y}\left(t, x(t), z(t), p\right) \qquad \forall\, t \in \mathbb{T}\,, \tag{2.1.2}$$

where $w_0 \in \mathbb{R}^{n_x}$, $f:\ \mathcal{D}_f \subseteq \mathbb{R}^{1+n_x+n_z+n_u+n_p} \rightarrow \mathbb{R}^{n_x}$, $g:\ \mathcal{D}_g \subseteq \mathbb{R}^{1+n_x+n_z+n_u+n_p} \rightarrow \mathbb{R}^{n_g}$, and $\hat{y}:\ \mathcal{D}_{\hat{y}} \subseteq \mathbb{R}^{1+n_x+n_z+n_p} \rightarrow \mathbb{R}^{n_y}$.

It should be noted that there exists a great variety of different model types within the MPC context which can be roughly divided into *first principles models* and *identified models*. First principles models try to replicate, e.g., physical or chemical laws of nature whereas identified models are based on empirical measurements of the real process. The definition given above is suited for *dynamical first principles models* which will be used throughout this thesis except for Chapter 6. In the latter case *dynamical identified models* are used which were obtained by choosing the so-called *state-space representation* (2.1.1)-(2.1.2) such that it best matches the measured inputs to the measured outputs. An important class of identified models are so-called *step* or *impulse response models*, which do not include process states and are described in more detail in [18]. Another approach, which does not clearly fit into the mentioned categories, is the usage of *neural network models* [69]. Further examples for the different model types and their application in industry can be found in [72].

Model predictive control uses a process model in order to forecast the process dynamics as well as the process outputs and calculates inputs which optimise this predicted process behaviour with respect to a so-called *objective function* and subject to desired *constraints*. The forecasting is performed for a certain period, the *prediction horizon* of length $t_{\text{p}} \in \mathbb{R}_{>0}$, by integrating the model equations (2.1.1).

A (continuous-time) *objective function* measures the process performance over the prediction horizon $\mathbb{T}_{\text{p}} \stackrel{\text{def}}{=} [t_0, t_0 + t_{\text{p}}]$, $t_0 \in [t_{\text{start}}, t_{\text{end}} - t_{\text{p}}]$, and is usually of the following *Bolza type*:

$$\int_{t_0}^{t_0+t_{\text{p}}} \psi\left(t, y(t), u(t)\right)\, dt \;+\; \phi\left(y(t_0 + t_{\text{p}})\right)\,, \tag{2.1.3}$$

where $\psi:\ \mathcal{D}_\psi \subseteq \mathbb{R}^{1+n_y \times n_u} \rightarrow \mathbb{R}$ and $\phi:\ \mathcal{D}_\phi \subseteq \mathbb{R}^{n_y} \rightarrow \mathbb{R}$ are called *Lagrange* and *objective function!Mayer term*, respectively. Note that the Lagrange term measures the process performance *during* the prediction horizon whereas the Mayer term only evaluates the process output *at the end* of the prediction horizon. We use the common convention that the objective function is formulated in such a way that we aim at minimising its value.

One of the most important features of MPC is its capability to guarantee that process inputs or outputs satisfy desired *constraints* which can be written in the following general form

$$l \;\leq\; c\left(t, y(t), u(t), p\right)\,, \tag{2.1.4}$$

where $c : \mathcal{D}_c \subseteq \mathbb{R}^{1+n_y+n_u+n_p} \rightarrow \mathbb{R}^{n_c}$ is a suitable function defining, together with $l \in \mathbb{R}^{n_c}$, n_c inequality constraints. It is obvious that also equality constraints can be expressed using this formulation (although they could be included in g, too).

With these ingredients, namely Eqs. (2.1.1)-(2.1.4), we are able to formulate

Definition 2.1 (open-loop optimal control problem): *An* open-loop optimal control problem *over the prediction horizon* $\mathbb{T}_\mathrm{p} \stackrel{\mathrm{def}}{=} [t_0, t_0+t_\mathrm{p}]$, $t_\mathrm{p} \in \mathbb{R}_{>0}$, *is the task of finding an optimal process input* $u(t)$ *solving*

$$\mathrm{OCP}(t_0): \quad \min_{\substack{x(t),\, z(t),\\ u(t),\, y(t)}} \int_{t_0}^{t_0+t_\mathrm{p}} \psi\left(t, y(t), u(t)\right)\, dt \,+\, \phi\left(y(t_0+t_\mathrm{p})\right) \tag{2.1.5a}$$

$$\text{s.t.} \quad x(t_0) \;=\; w_0(t_0)\,, \tag{2.1.5b}$$

$$\dot{x}(t) \;=\; f\left(t, x(t), z(t), u(t), p\right) \quad \forall\, t \in \mathbb{T}_\mathrm{p}\,, \tag{2.1.5c}$$

$$\mathbb{0} \;=\; g\left(t, x(t), z(t), u(t), p\right) \quad \forall\, t \in \mathbb{T}_\mathrm{p}\,, \tag{2.1.5d}$$

$$y(t) \;=\; \hat{y}\left(t, x(t), z(t), p\right) \quad \forall\, t \in \mathbb{T}_\mathrm{p}\,, \tag{2.1.5e}$$

$$l \;\leq\; c\left(t, y(t), u(t), p\right) \quad \forall\, t \in \mathbb{T}_\mathrm{p}\,, \tag{2.1.5f}$$

where the notation $w_0(t_0)$ *indicates that the inital process state depends on the starting time* t_0. ○

Let us assume that the process to be controlled via MPC starts at time instant t_start, ends at time instant t_end $(\infty < t_\mathrm{start} < t_\mathrm{end} < \infty)$ and that

$$t_0 < t_1 < \ldots < t_{n_\mathrm{sample}}\,, \quad n_\mathrm{sample} \in \mathbb{N}\,, \tag{2.1.6a}$$

$$t_0 \stackrel{\mathrm{def}}{=} t_\mathrm{start}\,, \quad t_{n_\mathrm{sample}} \stackrel{\mathrm{def}}{=} t_\mathrm{end} \tag{2.1.6b}$$

is a sequence of *sampling instants* satisfying

$$t_i - t_{i-1} \;\leq\; t_\mathrm{p} \quad \forall\, i \in \{1, \ldots, n_\mathrm{sample}\}\,. \tag{2.1.7}$$

After the solution of $\mathrm{OCP}(t_i)$ the optimal process input $\breve{u}^\mathrm{opt}(t)$ is applied to the process until the next sampling instant t_{i+1}. Then the current process state is obtained (measured or estimated) and the optimal control problem $\mathrm{OCP}(t_{i+1})$ is solved with this updated initial value for the process state. This yields the model predictive control concept which is summarised in Algorithm 2.1 and illustrated in Fig. 2.1.

One may ask why it is necessary to solve the open-loop optimal control problem repeatedly: If one would choose $t_\mathrm{p} \stackrel{\mathrm{def}}{=} t_\mathrm{end} - t_\mathrm{start}$ it would suffice to solve the first problem $\mathrm{OCP}(t_\mathrm{start})$ and to apply the resulting justified if one assumes, from a purely theoretical point of view, that the model describes the real process *exactly* and that all inputs can be applied *instantaneously* to the real process.

However, these conditions are never satisfied in a real-world environment: except for very rare cases there are always discrepancies between the model and real process, known as *model-plant mismatch*, as the real process is too complex to model it exactly. Sometimes the process dynamics are not even known completely making approximations or interpolations

necessary. Moreover, unknown *disturbances* are almost always present in real-world and *measurement noise*[2] impedes the exact determination of the initial process state. On the other hand, the calculated optimal inputs often cannot be applied exactly to the real process. Since actuators, valves and even electronic devices need a short time period, known as *dead time*, to react, there is always a short delay in the application of the optimal inputs (although this could be counteracted by prediction). A further delay stems from the fact that the controller needs some time to calculate the new optimal inputs. And even if these delays are negligible, deviations between the optimised and the applied inputs may occur because the actuators are not able to behave like an, in principle, arbitrary (measurable) function $u(t)$ including discontinuities.
All these circumstances make a *feedback control* strategy mandatory for a real-world setup. The incorporation of the current process state (as initial value) at each sampling instant adjusts the predicted process behaviour to the real one leading to more reliable results. Normally the more severe the above-mentioned effects are the more sampling instants are chosen. If the sampling instants are chosen *equidistant*, i.e.

$$\delta \overset{\text{def}}{=} \frac{t_{\text{end}} - t_{\text{start}}}{n_{\text{sample}}}, \quad t_i \overset{\text{def}}{=} i \cdot \delta \quad \forall\, i \in \{1, \ldots, n_{\text{sample}}\}\,, \tag{2.1.8}$$

we call $\delta \in \mathbb{R}_{>0}$ the *sampling time*.

Algorithm 2.1 (model predictive control concept)

`input:` open-loop optimal control problem $\mathrm{OCP}(t_0)$,
sequence of sampling instants $t_0, t_1, \ldots, t_{n_{\text{sample}}-1}$ as defined in (2.1.6)

`output:` piecewise defined optimal process inputs $u^{\text{opt}} : [t_{\text{start}}, t_{\text{end}}] \to \mathbb{R}^{n_u}$

(1) Set $i \leftarrow 0$.

(2) Obtain current process state $w_0(t_i)$ and formulate $\mathrm{OCP}(t_i)$.

(3) Obtain $\breve{u}^{\text{opt}}(t)$, $t \in [t_i, t_i + t_{\text{p}}]$, by solving $\mathrm{OCP}(t_i)$.

(4) Set $u^{\text{opt}}(t) \overset{\text{def}}{=} \breve{u}^{\text{opt}}(t)\ \forall\, t \in [t_i, t_{i+1}]$ and apply $\breve{u}^{\text{opt}}(t)\big|_{[t_i, t_{i+1}]}$ to the process until t_{i+1}.

(5) `if` $i = n_{\text{sample}} - 1$:

    `stop!`

`else`

    Set $i \leftarrow i + 1$ and continue with step (2).

[2] We should emphasise that the current process state w_0 is never known exactly in practice since it has to be obtained by means of (more or less) inaccurate sensors.

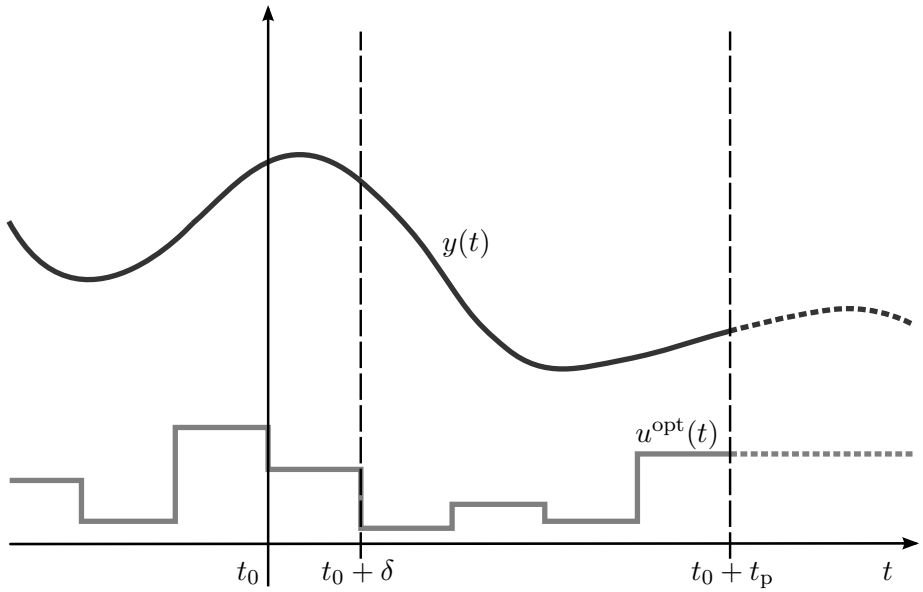


Figure 2.1: Main concept of model predictive control.

So far, our model predictive control formulation has been rather general as we did not pose further conditions on the functions f, g, $\hat{y}$, c, ψ or ϕ. These functions should be sufficiently *smooth*, e.g. twice continuously differentiable, in order to guarantee the existence (and uniqueness) of a solution but they can, in principle, be arbitrary *nonlinear* functions. The open-loop optimal control problems arising in this *nonlinear model predictive control (NMPC)* context can, e.g., be solved using the *direct multiple shooting method* (see [15], [14], [23], [24]) which is briefly summarised in Appendix C, where also an application example is given.
For ease of notation we eliminate the explicit dependencies of f, g, $\hat{y}$, c on t and p, which can be done without loss of generality:

- Our definition allows process models depending *explicitly* on time; most presentations on this topic, however, require the process model to be *time-invariant*, or *autonomous*. Explicit time dependence can be eliminated if an additional state $x_{n_x+1}(t)$ and the additional differential equation

$$\begin{aligned} x_{n_x+1}(t_\mathrm{start}) &= t_\mathrm{start}\,, && \text{(2.1.9a)} \\ \dot{x}_{n_x+1}(t) &= 1 \quad \forall\, t \in \mathbb{T} && \text{(2.1.9b)} \end{aligned}$$

  is introduced.

- *Process parameters* can be written as differential states by introduction of additional states $x_{n_x+i}(t)$, $1 \leq i \leq n_p$, and imposing the additional equations

$$\begin{aligned} x_{n_x+i}(t_\mathrm{start}) &= p_i \quad \forall\, i \in \{1, \ldots, n_p\}\,, && \text{(2.1.10a)} \\ \dot{x}_{n_x+i}(t) &= 0 \quad \forall\, t \in \mathbb{T} \quad \forall\, i \in \{1, \ldots, n_p\}\,. && \text{(2.1.10b)} \end{aligned}$$

The following presentation is restricted to *time-invariant, linear* open-loop optimal control problems as they are more directly linked to the utilisation of the proposed online active set strategy. Furthermore, from now on we make the assumption that the process model does not include algebraic variables. This means that the process state is described by a system of *ordinary differential equations (ODEs)*

$$x(t_{\text{start}}) = w_0\,, \tag{2.1.11a}$$
$$\dot{x}(t) = f\left(x(t), u(t)\right) \quad \forall\, t \in \mathbb{T}\,, \tag{2.1.11b}$$

instead of a DAE system (2.1.1). This assumption is very common within the linear model predictive control community.

2.2 Linear Model Predictive Control

The name *linear model predictive control* refers to situations in which a *linear time-invariant* process model, *linear* constraints and a *quadratic* objective function is used. This does *not* imply that the real process to be controlled has linear dynamics.

A (continuous-time) process model is called *linear time-invariant (LTI)* if it can be written in the form

$$x(t_{\text{start}}) = w_0\,, \tag{2.2.1a}$$
$$\dot{x}(t) = Ax(t) + Bu(t) \quad \forall\, t \in \mathbb{T}\,, \tag{2.2.1b}$$
$$y(t) = Cx(t) \quad \forall\, t \in \mathbb{T}\,, \tag{2.2.1c}$$

with *constant*[3] matrices $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{n_y \times n_x}$. Since almost all real processes exhibit nonlinearities, linear process models are often obtained by linearising a nonlinear model at some working point, normally at a *steady-state*.

Definition 2.2 (steady-state): *Every pair* $(\hat{x}, \hat{u})$ *satisfying*

$$\mathbb{0} = f\left(\hat{x}, \hat{u}\right) \tag{2.2.2}$$

is called a steady-state *of a system of ordinary differential equations*

$$\dot{x}(t) = f\left(x(t), u(t)\right) \quad \forall\, t \in \mathbb{T}\,. \tag{2.2.3}$$

This means that a process is at a steady-state iff it remains there if input $\hat{u}$ *is applied.* ○

Constraints for a process model are called *linear* iff they can be written as

$$l \leq My(t) + Nu(t)\,, \tag{2.2.4}$$

with constant matrices $M \in \mathbb{R}^{n_c \times n_y}$, $N \in \mathbb{R}^{n_c \times n_u}$ and a constant lower bound vector $l \in \mathbb{R}^{n_c}$. As a special case of (2.2.4), in most linear MPC problems at least *bounds* on the inputs and outputs are imposed, i.e.

$$\underline{u} \leq u(t) \leq \overline{u} \quad \forall\, t \in \mathbb{T}\,, \tag{2.2.5a}$$
$$\underline{y} \leq y(t) \leq \overline{y} \quad \forall\, t \in \mathbb{T}\,, \tag{2.2.5b}$$

[3] Linear *time-variant* process models allow for time-varying matrices $A(t)$, $B(t)$ and $C(t)$.

where $\underline{u}$, $\overline{u} \in \mathbb{R}^{n_u}$ and $\underline{y}$, $\overline{y} \in \mathbb{R}^{n_y}$. Input bounds typically express physical limitations of the actuators, output bounds are often necessary to ensure safe process operating conditions.

The objective function (of Bolza type) is *(convex) quadratic* iff it can be written as

$$\begin{aligned} &\frac{1}{2} \int_{t_0}^{t_0+t_\mathrm{p}} (y(t)-y_\mathrm{ref})' Q\,(y(t)-y_\mathrm{ref}) + (u(t)-u_\mathrm{ref})' R\,(u(t)-u_\mathrm{ref})\; dt \\ &\quad + \frac{1}{2}\,(y(t_0+t_\mathrm{p})-y_\mathrm{ref})' P\,(y(t_0+t_\mathrm{p})-y_\mathrm{ref})\;, \end{aligned} \tag{2.2.6}$$

with constant matrices $Q \in \mathcal{S}^{n_y}_{\succeq 0}$, $R \in \mathcal{S}^{n_u}_{\succ 0}$, $P \in \mathcal{S}^{n_y}_{\succeq 0}$ and constant *reference value vectors* $y_\mathrm{ref} \in \mathbb{R}^{n_y}$, $u_\mathrm{ref} \in \mathbb{R}^{n_u}$.
Matrix Q—we will discuss the meaning of P later—may penalise deviations of the process outputs from a certain reference value, therefore positive semi-definitness is assumed. Matrix R is required to be positive definite in order to penalise deviations of the process inputs from a desired reference value. Positive definiteness of R is also necessary in order to ensure that the resulting optimisation problem is strictly convex, as will be shown in Theorem 2.2. MPC problems with this type of objective are often referred to as *reference tracking problems*; also *trajectory tracking problems* where y_ref and u_ref vary with time are conceivable. In the special case where $y(t) \stackrel{\mathrm{def}}{=} x(t)\ \forall\, t \in \mathbb{T}$, $y_\mathrm{ref} \stackrel{\mathrm{def}}{=} \mathbb{0}$, $u_\mathrm{ref} \stackrel{\mathrm{def}}{=} \mathbb{0}$ they aim at *regulating* the process *to the origin*.

After these preperations we can give the following

Definition 2.3 (linear open-loop optimal control problem): *A* linear open-loop optimal control problem *over the prediction horizon* $\mathbb{T}_\mathrm{p} \stackrel{\mathrm{def}}{=} [t_0, t_0+t_\mathrm{p}]$, $t_\mathrm{p} \in \mathbb{R}_{>0}$, *is the task of finding an optimal process input* $u(t)$ *solving*

$$\begin{aligned} \mathrm{OCP}_\mathrm{lin}(t_0): \min_{\substack{x(t),\,u(t),\\ y(t)}} \; & \frac{1}{2} \int_{t_0}^{t_0+t_\mathrm{p}} (y(t)-y_\mathrm{ref})' Q\,(y(t)-y_\mathrm{ref}) + (u(t)-u_\mathrm{ref})' R\,(u(t)-u_\mathrm{ref})\; dt \\ & + \frac{1}{2}\,(y(t_0+t_\mathrm{p})-y_\mathrm{ref})' P\,(y(t_0+t_\mathrm{p})-y_\mathrm{ref}) && \text{(2.2.7a)} \\ \text{s.t.} \quad & x(t_0) \;=\; w_0(t_0)\,, && \text{(2.2.7b)} \\ & \dot{x}(t) \;=\; Ax(t)+Bu(t) \quad \forall\, t \in \mathbb{T}_\mathrm{p}\,, && \text{(2.2.7c)} \\ & y(t) \;=\; Cx(t) \qquad\quad \forall\, t \in \mathbb{T}_\mathrm{p}\,, && \text{(2.2.7d)} \\ & l \;\leq\; My(t)+Nu(t) \qquad \forall\, t \in \mathbb{T}_\mathrm{p}\,, && \text{(2.2.7e)} \end{aligned}$$

where all quantities are defined as in Eqs. (2.2.1), (2.2.4), (2.2.6). ○

2.2.1 Problem Discretisation

If $u(t)$ is allowed to be an arbitrary measurable real-valued function, $\mathrm{OCP}_\mathrm{lin}$ (and its generalisation OCP) is an infinite dimensional (over $\mathbb{R}$) optimisation problem. Although there exist necessary conditions—based on the *calculus of variations* or *Pontryagin's maximum principle* [49], [70]—for finding the optimal solution of such problems, these so-called *indirect methods* are of limited use for MPC purposes (cf. [12, p. 85-87]).

Direct methods parameterise the control functions in order to reduce the optimal control problem to a *finite* dimensional one. This loss of degrees of freedom greatly simplifies the solution of the problem but is normally irrelevant for process performance in practice. A very popular *control parameterisation* is to require that the control functions are *piecewise constant* (or *piecewise linear*) on an equidistant grid, as anticipated in Figure 2.1. If the prediction horizon $[t_0, t_0+t_\mathrm{p}]$ is divided into n_p intervals of length $\delta_\mathrm{p} \stackrel{\mathrm{def}}{=} \frac{t_\mathrm{p}}{n_\mathrm{p}}$ this can formally be written as:

$$u(t_0 + i \cdot \delta_\mathrm{p} + t) \stackrel{\mathrm{def}}{=} u_i \quad \forall\, t \in [0, \delta_\mathrm{p}) \;\; \forall\, i \in \{0, \ldots, n_\mathrm{p} - 1\}\,, \tag{2.2.8a}$$

$$u(t_0 + t_\mathrm{p}) \stackrel{\mathrm{def}}{=} u_{n_\mathrm{p}-1}\,, \tag{2.2.8b}$$

with $u_i \in \mathbb{R}^{n_u}$, $0 \leq i \leq n_\mathrm{p} - 1$. In general, it is reasonable to choose

$$\delta_\mathrm{p} \stackrel{\mathrm{def}}{=} \sigma \cdot \delta\,, \quad \sigma \in \mathbb{N}\,. \tag{2.2.9}$$

After a control parameterisation the trajectories $x(t)$ and $y(t)$ can be expressed as functions of the initial value w_0 and finitely many optimisation variables $u_0, \ldots, u_{n_\mathrm{p}-1}$; thus the optimal control problem $\mathrm{OCP}_\mathrm{lin}$ is transformed into a *quadratic program (QP)* which comprises a quadratic objective function and linear constraints[4]. Direct methods are usually subdivided into three main variants depending on the way in which these trajectories are evaluated:

- *direct single shooting* integrates the ODE system over the whole prediction horizon at once for fixed values of w_0 and u_i;
- *direct multiple shooting* [15] solves the ODE system independently on each interval $[t_0 + i \cdot \delta_\mathrm{p}, t_0 + (i+1) \cdot \delta_\mathrm{p}]$ by introducing additional *intermediate* initial values and adding *continuity constraints* to the NLP (see Section C.4 for further details);
- *direct collocation* [79] approximates the trajectory $x(t)$ by piecewise polynomials which satisfy the ODE only at a the points of a fine grid.

Also the constraints need to be discretised and their fulfilment is ensured only at a finite number of time instants, e.g. at $t_0 + i \cdot \delta_\mathrm{p}$, $1 \leq i \leq n_\mathrm{p} - 1$. Similarly, the continuous objective function is evaluated on a discrete time-grid only (of course, this is always done when using numerical quadrature formulae).

For the solution of linear open-loop optimal control problems a direct single or multiple shooting approach is often appropriate. Therefore we parameterise the controls, or process inputs, as piecewise constant functions on an equidistant grid $\mathbb{T}_\mathrm{p}^\mathrm{disc} \stackrel{\mathrm{def}}{=} \{k_0, \ldots, k_0 + n_\mathrm{p} - 1\}$. The objective function as well as the constraints are evaluated only at the time instants of this grid and thus the values of the trajectories $x(t)$ and $y(t)$ are calculated only there. We end up with a

[4] In the general case the optimal control problem OCP is transformed into a nonlinear programming (NLP) problem with a nonlinear objective function and possibly nonlinear constraints.

Definition 2.4 (discrete-time linear open-loop optimal control problem): *A* discrete-time linear open-loop optimal control problem *over the discrete-time prediction horizon* $\mathbb{T}_{\mathrm{p}}^{\mathrm{disc}} \stackrel{\mathrm{def}}{=} \{k_0, \ldots, k_0 + n_{\mathrm{p}} - 1\}$, $n_{\mathrm{p}} \in \mathbb{N}$, *is the task of finding a sequence of constant optimal process inputs* $u_{k_0}, \ldots, u_{k_0+n_{\mathrm{p}}-1}$ *solving*

$$\mathrm{OCP}_{\mathrm{lin}}^{\mathrm{disc}}(k_0): \quad \min_{\substack{x_{k_0},\ldots,x_{k_0+n_{\mathrm{p}}},\\ y_{k_0},\ldots,y_{k_0+n_{\mathrm{p}}},\\ u_{k_0},\ldots,u_{k_0+n_{\mathrm{p}}-1}}} \frac{1}{2} \sum_{k=k_0}^{k_0+n_{\mathrm{p}}-1} (y_k - y_{\mathrm{ref}})' Q (y_k - y_{\mathrm{ref}}) + (u_k - u_{\mathrm{ref}})' R (u_k - u_{\mathrm{ref}})$$

$$+ \frac{1}{2} (y_{k_0+n_{\mathrm{p}}} - y_{\mathrm{ref}})' P (y_{k_0+n_{\mathrm{p}}} - y_{\mathrm{ref}}) \tag{2.2.10a}$$

$$\text{s.t.} \quad x_{k_0} = w_0(k_0), \tag{2.2.10b}$$

$$x_{k+1} = A^{\mathrm{disc}} x_k + B^{\mathrm{disc}} u_k \quad \forall\, k \in \mathbb{T}_{\mathrm{p}}^{\mathrm{disc}}, \tag{2.2.10c}$$

$$y_k = C x_k \qquad \forall\, k \in \mathbb{T}_{\mathrm{p}}^{\mathrm{disc}} \cup \{k_0 + n_{\mathrm{p}}\}, \tag{2.2.10d}$$

$$l \leq M y_k + N u_k \qquad \forall\, k \in \mathbb{T}_{\mathrm{p}}^{\mathrm{disc}}, \tag{2.2.10e}$$

where all quantities, except for $A^{\mathrm{disc}} \in \mathbb{R}^{n_x \times n_x}$ *and* $B^{\mathrm{disc}} \in \mathbb{R}^{n_x \times n_u}$, *are defined as in Eqs.* (2.2.1), (2.2.4), (2.2.6). ○

The discrete-time system matrices A^{disc} and B^{disc} can be calculated from their continuous counterparts: standard calculus leads to the solution of the ODE system (2.2.7c)

$$x(t) = \mathrm{e}^{(t-t_0)A} x(t_0) + \int_{t_0}^{t} \mathrm{e}^{(t-s)A} B u(s)\, ds \quad \forall\, t \geq t_0. \tag{2.2.11}$$

If the process input on the intervall $[k_0, k_1] \stackrel{\mathrm{def}}{=} [t_0, t_0 + \delta_{\mathrm{p}}]$ has constant value $u_0 \in \mathbb{R}^{n_u}$, the process state at time instant $t_0 + \delta_{\mathrm{p}}$ is

$$x(t_0 + \delta_{\mathrm{p}}) = \mathrm{e}^{(t_0+\delta_{\mathrm{p}}-t_0)A} x(t_0) + \int_{t_0}^{t_0+\delta_{\mathrm{p}}} \mathrm{e}^{(t_0+\delta_{\mathrm{p}}-s)A} B u(s)\, ds \tag{2.2.12a}$$

$$= \underbrace{\mathrm{e}^{\delta_{\mathrm{p}} A}}_{\stackrel{=}{\mathrm{def}} A^{\mathrm{disc}}} x(t_0) + \underbrace{\left(\int_{t_0}^{t_0+\delta_{\mathrm{p}}} \mathrm{e}^{(t_0+\delta_{\mathrm{p}}-s)A} B\, ds \right)}_{\stackrel{=}{\mathrm{def}} B^{\mathrm{disc}}} u_0. \tag{2.2.12b}$$

It is easy to show by induction that the process states at all time instants in $\mathbb{T}_{\mathrm{p}}^{\mathrm{disc}}$ can be obtained via the same matrices A^{disc} and B^{disc} accordingly, provided that the values of $\mathbb{T}_{\mathrm{p}}^{\mathrm{disc}}$ are *equidistant*. For ease of notation, we drop the superscript "disc" from A^{disc}, B^{disc} and $\mathbb{T}_{\mathrm{p}}^{\mathrm{disc}}$ in the remainder of this thesis if an equidistant discrete-time prediction horizon is used.

2.2.2 Closed-Loop Stability

Now, we will give a short discussion on the meaning of the so-called *terminal penalty weight matrix* P in Eqs. (2.2.6), (2.2.7a) and (2.2.10a). It is introduced in order to compensate the finiteness of the prediction horizon $\mathbb{T}_\mathrm{p}$: due to (online) solution complexity the prediction horizon is usually much shorter than the total runtime of the controlled process, i.e. $t_\mathrm{p} \ll t_\mathrm{end} - t_\mathrm{start}$. Thus it may happen that optimal process inputs for the time interval $[t_i, t_i + t_\mathrm{p}]$ lead to very poor process performance afterwards. Of course, there will be re-optimisations until $t_i + t_\mathrm{p}$ but too short-sighted actions can spoil future behaviour, anyway, and it may even happen that the controller causes the process to start oscillating. This observation, which has also great practical relevance, is topic of a huge number of articles which investigate (necessary and) sufficient conditions for *stability* of a controlled process (see e.g. [76], [55], [13], [20], [61] and the references therein).
We consider a (discrete-time) time-invariant linear process model as described by equations (2.2.10b)-(2.2.10d). Let us assume that the corresponding optimal control problem $\mathrm{OCP}_\mathrm{lin}^\mathrm{disc}(k_0)$ is feasible for all $w_0 \stackrel{\mathrm{def}}{=} w_0(k_0) \in \mathbb{R}^{n_x}$ and its (unique) optimal solution is the sequence $u_{k_0}(w_0), \ldots, u_{k_0+n_\mathrm{p}-1}(w_0)$. Then we can define a (usually nonlinear) vector-valued mapping

$$\begin{aligned} J: \quad \mathbb{R}^{n_x} &\longrightarrow \mathbb{R}^{n_u} \\ w_0 &\longmapsto u_{k_0}(w_0)\,, \end{aligned} \tag{2.2.13}$$

which enables us to write the ODE system of the *closed-loop controlled* process model as

$$x_{k_0} = w_0\,, \tag{2.2.14a}$$
$$x_{k+1} = Ax_k + BJ(x_k) \quad \forall\, k \in \mathbb{T}_\mathrm{p} \tag{2.2.14b}$$
$$= (A + BJ)(x_k) \quad \forall\, k \in \mathbb{T}_\mathrm{p}\,. \tag{2.2.14c}$$

If $(\hat{x}, \hat{u}) \in \mathbb{R}^{n_x+n_u}$ denotes an arbitrary *steady-state* of the process model and

$$y_\mathrm{ref} \stackrel{\mathrm{def}}{=} C\hat{x}\,, \quad u_\mathrm{ref} \stackrel{\mathrm{def}}{=} \hat{u} \tag{2.2.15}$$

is chosen, $J(\hat{x}) = \hat{u}$ holds because the objective function has optimal value 0 for the choice $u_{k_i} = \hat{u}\ \forall\, i \in \{0, \ldots, n_\mathrm{p} - 1\}$. Thus, if the closed-loop controlled process is at this steady-state it will stay there. The controlled process is called *closed-loop asymptotically stable* if it returns to the steady-state $(\hat{x}, \hat{u})$ from every initial process state value:

Definition 2.5 (closed-loop asymptotic stability): *Let a discrete-time time-invariant linear process model with steady-state $(\hat{x}, \hat{u})$, a corresponding open-loop optimal control problem $\mathrm{OCP}_\mathrm{lin}^\mathrm{disc}(k_0)$ (which is feasible for all $w_0 \in \mathbb{R}^{n_x}$) satisfying the definitions* (2.2.15) *and a map J as in* (2.2.13) *be given.*
Then the process model closed-loop controlled through J is called closed-loop asymptotically stable *iff*

$$\|x_k - \hat{x}\|_2 \;\to\; 0 \quad \textit{as} \quad k \;\to\; \infty\,, \tag{2.2.16}$$

no matter from which initial process state $w_0 \in \mathbb{R}^{n_x}$ the closed-loop control is started. ○

It is easy to show that a closed-loop controlled process model is closed-loop asymptotically stable if and only if the norm of all eigenvalues of the mapping $A + BJ$ in Eq. (2.2.14c) is smaller than one. Under some mild conditions (stabilisability[5] and detectability[5]), it can be shown that linear MPC is closed-loop asymptotically stable if an infinite prediction horizon is used (cf. e.g. [2, p. 773]). For linear MPC with a finite prediction horizon the following result holds [74]:

Theorem 2.1 (stability of linear MPC): *Let*

$$\min_{u_{k_0},\ldots,u_{k_0+n_\mathrm{p}-1}} \quad \frac{1}{2}\sum_{k=k_0}^{k_0+n_\mathrm{p}-1} x_k' Q x_k + u_k' R u_k + \frac{1}{2} x_{k_0+n_\mathrm{p}}' P x_{k_0+n_\mathrm{p}} \tag{2.2.17a}$$

$$\text{s.t.} \quad x_{k_0} = w_0(k_0)\,, \tag{2.2.17b}$$

$$x_{k+1} = Ax_k + Bu_k \quad \forall\, k \geq k_0\,, \tag{2.2.17c}$$

$$\overline{x} \leq Mx_k \quad \forall\, k \geq k_0\,, \tag{2.2.17d}$$

$$\overline{u} \leq Nu_k \quad \forall\, k \geq k_0\,, \tag{2.2.17e}$$

with $\overline{x} \in \mathbb{R}^{n_x}$, $\overline{u} \in \mathbb{R}^{n_u}$ and $\overline{x}, \overline{u} < \mathbb{0}$, be a discrete-time linear open-loop optimal control problem with vectors and matrices defined as in Definition 2.4. If, in addition, (A, B) is stabilisable, $\left(Q^{\frac{1}{2}}, A\right)$ is detectable, and if P is the (unique) solution of the discrete algebraic Riccati equation

$$P = Q + A'PA - A'PB\left(R + B'PB\right)^{-1} B'PA\,. \tag{2.2.18}$$

Then there exists a finite *value $n_\mathrm{p} \stackrel{\mathrm{def}}{=} n_\mathrm{p}^* \in \mathbb{N}$ such that the sequence of optimal process inputs $u_{k_0}, \ldots, u_{k_0+n_\mathrm{p}^*-1}$ as well as the optimal objective function value of* (2.2.17) *are also optimal for the choice $n_\mathrm{p} \stackrel{\mathrm{def}}{=} \infty$ (without the summand including P). Thus, also the optimal control problem* (2.2.17) *with finite prediction horizon n_p^* is closed-loop asymptotically stable.* ○

Proof: Can be found in [74]. □

This result shows that it is possible to replace the linear open-loop optimal control problem over an *infinite horizon* ($n_\mathrm{p} = \infty$) by a *finite* one whithout losing optimality and *stability*. Since it only states the existence of such an $n_\mathrm{p}^* \in \mathbb{N}$ the question remains open: how to choose n_p in practice? The proof of Theorem 2.1 is based on the observation that there always exists a time instant n_p^* as from which no input or state constraint would be violated even if they were omitted from the problem formulation (yielding the so-called *linear-quadratic regulator* [53]). If such an n_p^* is chosen as length of the finite prediction horizon optimality of the solution is preserved. Therefore, it is suggested in [64], where a similar strategy for the nonlinear case is presented, to ensure that n_p "is 'large' compared to the system dynamics". Of course, this is not a rigorous answer but as a rule-of-thumb it should suffice to choose the length of the prediction horizon a few times larger than the time the process needs to return into a steady-state after a strong pertubation.

[5] For a definition see any textbook on control theory, e.g. [2] or [90].

2.2.3 Condensing into a Smaller Scale Parametric Quadratic Program

In this section we will show how the discretised linear open-loop optimal control problem $\mathrm{OCP}_{\mathrm{lin}}^{\mathrm{disc}}(k_0)$, which is a *parametric quadratic program* (cf. Definition 2.11), can be transformed into a smaller scale one. For ease of notation, we consider only the case when the process is to be regulated to the origin (i.e. $y_k \overset{\mathrm{def}}{=} x_k \ \forall\, k \geq k_0$, $y_{\mathrm{ref}} \overset{\mathrm{def}}{=} \mathbb{0}$, $u_{\mathrm{ref}} \overset{\mathrm{def}}{=} \mathbb{0}$), adaptations to the general situation are straightforward.

Using Eq. (2.2.10c) all process states at time instants greater than k_0 can be expressed via the inital process state x_{k_0} and the input sequence $u_{k_0}, \ldots, u_{k_0+n_{\mathrm{p}}-1}$:

$$
\begin{aligned}
x_{k_0+1} &= Ax_{k_0} + Bu_{k_0}\,, && \text{(2.2.19a)}\\
x_{k_0+2} &= A\left(Ax_{k_0} + Bu_{k_0}\right) + Bu_{k_0+1} \;=\; A^2x_{k_0} + ABu_{k_0} + Bu_{k_0+1}\,, && \text{(2.2.19b)}\\
&\;\;\vdots\\
x_{k_0+j} &= A^j x_{k_0} + \sum_{i=0}^{j-1} A^{j-1-i}Bu_{k_0+i}\,, \quad j \in \{0,\ldots,n_{\mathrm{p}}\}\,. && \text{(2.2.19c)}
\end{aligned}
$$

In order to reformulate $\mathrm{OCP}_{\mathrm{lin}}^{\mathrm{disc}}(k_0)$ we introduce the following augmented quantities:

$$
\bar{x} \overset{\mathrm{def}}{=} \begin{pmatrix} x_{k_0} \\ x_{k_0+1} \\ \vdots \\ x_{k_0+n_{\mathrm{p}}} \end{pmatrix}, \qquad \bar{u} \overset{\mathrm{def}}{=} \begin{pmatrix} u_{k_0} \\ u_{k_0+1} \\ \vdots \\ u_{k_0+n_{\mathrm{p}}-1} \end{pmatrix}, \qquad \text{(2.2.20a)}
$$

$$
\bar{Q} \overset{\mathrm{def}}{=} \begin{pmatrix} Q & & & & \\ & Q & & & \\ & & \ddots & & \\ & & & Q & \\ & & & & P \end{pmatrix}, \qquad \bar{R} \overset{\mathrm{def}}{=} \begin{pmatrix} R & & & \\ & R & & \\ & & \ddots & \\ & & & R \end{pmatrix}, \qquad \text{(2.2.20b)}
$$

$$
\bar{A} \overset{\mathrm{def}}{=} \begin{pmatrix} \mathrm{Id} \\ A \\ A^2 \\ \vdots \\ A^{n_{\mathrm{p}}-1} \\ A^{n_{\mathrm{p}}} \end{pmatrix}, \qquad \bar{B} \overset{\mathrm{def}}{=} \begin{pmatrix} \mathbb{0} & & & & \\ B & & & & \\ AB & B & & & \\ \vdots & \ddots & \ddots & & \\ A^{n_{\mathrm{p}}-2}B & \cdots & AB & B & \\ A^{n_{\mathrm{p}}-1}B & A^{n_{\mathrm{p}}-2}B & \cdots & AB & B \end{pmatrix}, \qquad \text{(2.2.20c)}
$$

$$
\bar{M} \overset{\mathrm{def}}{=} \begin{pmatrix} M & & & \mathbb{0} \\ & M & & \vdots \\ & & \ddots & \vdots \\ & & M & \mathbb{0} \end{pmatrix}, \quad \bar{N} \overset{\mathrm{def}}{=} \begin{pmatrix} N & & & \\ & N & & \\ & & \ddots & \\ & & & N \end{pmatrix}, \quad \bar{l} \overset{\mathrm{def}}{=} \begin{pmatrix} l \\ l \\ \vdots \\ l \end{pmatrix}, \qquad \text{(2.2.20d)}
$$

wherein $\bar{x} \in \mathbb{R}^{(n_{\mathrm{p}}+1)\cdot n_x}$, $\bar{u} \in \mathbb{R}^{n_{\mathrm{p}}\cdot n_u}$, $\bar{Q} \in \mathbb{R}^{(n_{\mathrm{p}}+1)\cdot n_x \times (n_{\mathrm{p}}+1)\cdot n_x}$, $\bar{R} \in \mathbb{R}^{n_{\mathrm{p}}\cdot n_u \times n_{\mathrm{p}}\cdot n_u}$, $\bar{A} \in \mathbb{R}^{(n_{\mathrm{p}}+1)\cdot n_x \times n_x}$, $\bar{B} \in \mathbb{R}^{(n_{\mathrm{p}}+1)\cdot n_x \times n_{\mathrm{p}}\cdot n_u}$, $\bar{M} \in \mathbb{R}^{n_{\mathrm{p}}\cdot n_c \times (n_{\mathrm{p}}+1)\cdot n_x}$, $\bar{N} \in \mathbb{R}^{n_{\mathrm{p}}\cdot n_c \times n_{\mathrm{p}}\cdot n_u}$,

$\bar{l} \in \mathbb{R}^{n_\mathrm{p} \cdot n_c}$. Then the discrete linear open-loop optimal control problem $\mathrm{OCP}_{\mathrm{lin}}^{\mathrm{disc}}(k_0)$ (for regulating the process to the origin) can be written as follows:

$$\begin{aligned} \min_{\bar{u}} \quad & \tfrac{1}{2}\bar{x}'\bar{Q}\bar{x} + \bar{u}'\bar{R}\bar{u} && \text{(2.2.21a)} \\ \text{s.t.} \quad & x_{k_0} = w_0(k_0)\,, && \text{(2.2.21b)} \\ & \bar{x} = \bar{A}x_{k_0} + \bar{B}\bar{u}\,, && \text{(2.2.21c)} \\ & \bar{l} \leq \bar{M}\bar{x} + \bar{N}\bar{u}\,. && \text{(2.2.21d)} \end{aligned}$$

Substituting (2.2.21b) and (2.2.21c) into the objective (2.2.21a) and the constraints (2.2.21d) yields

$$\begin{aligned} \min_{\bar{u}} \quad & \tfrac{1}{2}\bar{u}'\left(\bar{B}'\bar{Q}\bar{B} + \bar{R}\right)\bar{u} + \bar{u}'\left(\bar{B}'\bar{Q}\bar{A}\right)w_0(k_0) + \tfrac{1}{2}w_0(k_0)'\bar{A}'\bar{Q}\bar{A}w_0(k_0) && \text{(2.2.22a)} \\ \text{s.t.} \quad & \bar{l} \leq \bar{M}\bar{A}w_0(k_0) + \left(\bar{M}\bar{B} + \bar{N}\right)\bar{u}\,. && \text{(2.2.22b)} \end{aligned}$$

This leads to

Theorem 2.2 (linear MPC and parametric QPs): *The discrete-time linear open-loop optimal control problem* (2.2.10) *(with $Q \in \mathcal{S}_{\succeq 0}^{n_y}$, $P \in \mathcal{S}_{\succeq 0}^{n_u}$, $R \in \mathcal{S}_{\succ 0}^{n_y}$) for a given constant $w_0 \in \mathbb{R}^{n_x}$ is a parametric quadratic program of the form*

$$\begin{aligned} \min_{\bar{u}} \quad & \tfrac{1}{2}\bar{u}'\bar{H}\bar{u} + \bar{u}'\bar{F}w_0 && \text{(2.2.23a)} \\ \text{s.t.} \quad & \bar{G}\bar{u} \geq \bar{l} - \bar{E}w_0\,, && \text{(2.2.23b)} \end{aligned}$$

where $\bar{H} \in \mathbb{R}^{n_\mathrm{p} \cdot n_u \times n_\mathrm{p} \cdot n_u}$, $\bar{F} \in \mathbb{R}^{n_\mathrm{p} \cdot n_u \times n_x}$, $\bar{G} \in \mathbb{R}^{n_\mathrm{p} \cdot n_c \times n_\mathrm{p} \cdot n_u}$, $\bar{E} \in \mathbb{R}^{n_\mathrm{p} \cdot n_c \times n_x}$, and the other quantities are defined as in Eqs. (2.2.20)*. Moreover, the matrix $\bar{H}$ is positive definite.* ○

Proof: The first statement follows directly from the discussion above by setting the matrices $\bar{H} \stackrel{\mathrm{def}}{=} \bar{B}'\bar{Q}\bar{B} + \bar{R}$, $\bar{F} \stackrel{\mathrm{def}}{=} \bar{B}'\bar{Q}\bar{A}$, $\bar{G} \stackrel{\mathrm{def}}{=} \bar{M}\bar{B} + \bar{N}$, $\bar{E} \stackrel{\mathrm{def}}{=} \bar{M}\bar{A}$ and the remark that the last summand of Eq. (2.2.22a) can be omitted since it is constant for fixed $w_0(k_0)$. It is easy to show that a QP of the same form is obtained for reference tracking problems.
It remains to prove that $\bar{H}$ is positive definite: $Q \in \mathcal{S}_{\succeq 0}^{n_x}$ and $P \in \mathcal{S}_{\succeq 0}^{n_x}$ imply that $\bar{Q}$ is positive semi-definite and thus also $\bar{B}'\bar{Q}\bar{B}$. Furthermore, $R \in \mathcal{S}_{\succ 0}^{n_u}$ implies that also $\bar{R}$ is positive definite. Since $\bar{H}$ is a sum of a positive semi-definite and a positive definite matrix it follows $\bar{H} \in \mathcal{S}_{\succ 0}^{n_\mathrm{p} \cdot n_u}$. □

Following [15], we call the transition from the large structured QP (2.2.21) to the smaller, but less structured QP (2.2.23) *condensing*. As a generalisation of Theorem 2.2, it can be shown that the solution of a (discretised) nonlinear MPC open-loop control problem is equivalent to the solution of a *nonlinear program (NLP)*. Usage of the *direct multiple shooting* approach [15] leads to specially structured NLPs which can efficiently be solved via a *sequential quadratic programming (SQP) method* [71], [85]. This class of methods is based on the successive solution of a *sequence* of quadratic programs, instead of a single one as in linear MPC (see also Section 4.7.2).

2.3 Quadratic Programming

In Section 2.2 we have seen that linear open-loop optimal control problems can be expressed as (parametric) *quadratic programs*:

Definition 2.6 (quadratic program): *The optimisation problem*

$$\mathrm{QP}: \qquad \min_{x\in\mathbb{R}^n} \quad \tfrac{1}{2}x'Hx + x'g \tag{2.3.1a}$$

$$\text{s.t.} \quad Gx \;\geq\; b\,, \tag{2.3.1b}$$

with

- *the* Hessian matrix $H \in \mathcal{S}^n \stackrel{\text{def}}{=} \{M \in \mathbb{R}^{n\times n} \mid M = M'\}$,
- *the* gradient vector $g \in \mathbb{R}^n$,
- *the* constraint matrix $G \in \mathbb{R}^{m\times n}$, *and*
- *the* constraint vector $b \in \mathbb{R}^m$,

is called a quadratic program. ○

Therein, the inequality constraints (2.3.1b) can also contain equality constraints, upper constraints' bounds as well as *bounds* on single variables x_i, $1 \leq i \leq n$, by virtue of a proper choice of G and b.
We denote the i-th row of the constraint matrix G by the vector G_i'; the matrix composed of the rows corresponding to constraints in any (ordered) index set $\mathbb{A} \subseteq \{1,\ldots,m\}$ is denoted by $G_\mathbb{A}$. The corresponding part of the *constraint vector* b (or any other vector $v \in \mathbb{R}^m$) is denoted by $b_\mathbb{A}$ ($v_\mathbb{A}$).

Definition 2.7 (feasibility, boundedness and convexity of a QP): *A quadratic program as defined in Definition 2.6 is called*

- feasible *iff its feasible set*

$$\mathcal{F} \stackrel{\text{def}}{=} \left\{\check{x} \in \mathbb{R}^n \mid G\check{x} \geq b\right\} \tag{2.3.2}$$

  is nonempty and infeasible *otherwise;*

- bounded (from below) *iff there exists a number* $\alpha \in \mathbb{R}$ *such that*

$$\alpha \leq \frac{1}{2}\check{x}'H\check{x} + \check{x}'g \quad \forall\, \check{x} \in \mathcal{F} \tag{2.3.3}$$

  and unbounded *otherwise;*

- convex *iff its Hessian matrix* H *is positive semi-definite, i.e.*

$$H \in \mathcal{S}^n_{\succeq 0}\,, \quad \mathcal{S}^n_{\succeq 0} \stackrel{\text{def}}{=} \left\{M \in \mathcal{S}^n \mid v'Mv \geq 0\ \forall\, v \in \mathbb{R}^n\right\} \tag{2.3.4}$$

  and nonconvex *otherwise;*

- strictly convex *iff its Hessian matrix H is positive definite, i.e.*

$$H \in \mathcal{S}^n_{\succ 0}\,, \quad \mathcal{S}^n_{\succ 0} \stackrel{\text{def}}{=} \left\{ M \in \mathcal{S}^n \mid v'Mv > 0 \;\forall\, v \in \mathbb{R}^n \setminus \{\mathbb{0}\} \right\}. \tag{2.3.5}$$

○

According to Theorem 2.2, all QPs arising within the linear MPC context have a positive definite Hessian matrix. Thus we make the standing assumption that from now on *all QPs are strictly convex*, unless stated otherwise. This also implies that all QPs are bounded from below because of the following

Lemma 2.1 (boundedness of strictly convex QPs): *Every strictly convex quadratic program of the form* (2.3.1) *is bounded from below.* ○

Proof: If we omit the constraints it is obvious from standard calculus that the unconstrained QP ($\mathcal{F} = \mathbb{R}^n$) has exactly one global minimiser at $\check{x} \stackrel{\text{def}}{=} -H^{-1}g$. Since the optimal objective function value cannot decrease when the feasible set is made smaller, i.e. $\mathcal{F} \subset \mathbb{R}^n$, we can choose $\alpha \stackrel{\text{def}}{=} \frac{1}{2}\check{x}'H\check{x} + \check{x}'g$ as a lower bound on all objective function values of the original QP. □

This also shows that a strictly convex quadratic program always has a solution if it is feasible:

Theorem 2.3 (Frank-Wolfe Theorem): *If a quadratic program* (2.3.1) *is bounded from below on a nonempty feasible set $\mathcal{F}$ (as defined in* (2.3.2)*), then the objective function attains its infimum on $\mathcal{F}$, i.e.*

$$\exists\, x^{\text{opt}} \in \mathcal{F}: \quad \frac{1}{2}x^{\text{opt}\prime}Hx^{\text{opt}} + x^{\text{opt}\prime}g \;\leq\; \frac{1}{2}\check{x}'H\check{x} + \check{x}'g \;\; \forall\, \check{x} \in \mathcal{F}\,. \tag{2.3.6}$$

○

Proof: If $\mathcal{F}$ is compact this is true for any continuous objective function. A proof for the general case can be found in the appendix of [35]. □

Duality is an important concept in linear programming that can also be extended to convex quadratic programming [27] (and also to general nonlinear programming [89]): the main idea is to formulate a second, the *dual*, problem which can be shown (under mild conditions) to have the same optimal objective function value as the original, the *primal*, one. Moreover, the dual objective function value at any dual feasible point provides a *lower bound* on the optimal *primal* objective function value. These theoretical properties are very helpful when proving optimality of a certain point and also lead to interesting practical methods for solving quadratic programs, as will be demonstrated in Chapter 3.

Definition 2.8 (dual quadratic program): *We define the* dual quadratic program *of the QP* (2.3.1) *to be the problem*

$$\begin{aligned} \text{QP}^{\text{dual}}: \qquad \max_{x\in\mathbb{R}^n,\, y\in\mathbb{R}^m} \quad & -\tfrac{1}{2}x'Hx + y'b && \text{(2.3.7a)}\\ \text{s.t.} \quad & Hx + g = G'y\,, && \text{(2.3.7b)}\\ & y \geq \mathbb{0}\,, && \text{(2.3.7c)} \end{aligned}$$

where all quantities are definied as in Definition 2.6.
The notions of feasibility, boundedness and convexity (cf. Definition 2.7) also apply to the dual QP; its feasible set *is defined as*

$$\mathcal{F}^{\mathrm{dual}} \stackrel{\mathrm{def}}{=} \left\{ (\check{x}, \check{y}) \in \mathbb{R}^n \mid H\check{x} + g = G'\check{y},\ \check{y} \geq \mathbb{0} \right\}, \tag{2.3.8}$$

accordingly. ○

Since an extensive treatment of duality is beyond the scope of this thesis, we only summarise the main result:

Theorem 2.4 (solution of primal and dual QP): *Let a strictly convex primal and the corresponding dual quadratic program (as defined in Definitions 2.6 and 2.8) be given. Then the following holds:*

(i) If x^{opt} *is a solution to* QP (2.3.1) *then a solution* $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ *to* $\mathrm{QP}^{\mathrm{dual}}$ *exists.*

(ii) If a solution $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ *to* $\mathrm{QP}^{\mathrm{dual}}$ *exists then* x^{opt} *is a solution to* QP (2.3.1)*.*

(iii) In either case

$$\frac{1}{2}x^{\mathrm{opt}\prime}Hx^{\mathrm{opt}} + x^{\mathrm{opt}\prime}g \;=\; -\frac{1}{2}x^{\mathrm{opt}\prime}Hx^{\mathrm{opt}} + y^{\mathrm{opt}\prime}b \tag{2.3.9}$$

holds. ○

Proof: Can be found in [27], where a very similar result for convex QPs was first published (note that our variant of the second proposition requires the invertibility of H). □

Corollary 2.1 (bounds on the optimal objective function values): *Let a* feasible, *strictly convex primal quadratic program with optimal solution* x^{opt} *and the corresponding dual be given (see Definitions 2.6 and 2.8). Then the objective function value of the dual at an arbitrary feasible point provides a* lower bound *on the optimal objective function value of the primal, i.e.*

$$\frac{1}{2}x^{\mathrm{opt}\prime}Hx^{\mathrm{opt}} + x^{\mathrm{opt}\prime}g \;\geq\; -\frac{1}{2}\check{x}'H\check{x} + \check{y}'b \quad \forall\ (\check{x}, \check{y}) \in \mathcal{F}^{\mathrm{dual}}. \tag{2.3.10}$$

○

Proof: Since the primal QP is feasible and bounded from below (cf. Lemma 2.1) a solution must exists according to Theorem 2.3. Thus Theorem 2.4 guarantees the existence of an optimal dual solution $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ implying

$$\frac{1}{2}x^{\mathrm{opt}\prime}Hx^{\mathrm{opt}} + x^{\mathrm{opt}\prime}g \;=\; -\frac{1}{2}x^{\mathrm{opt}\prime}Hx^{\mathrm{opt}} + y^{\mathrm{opt}\prime}b \;\geq\; -\frac{1}{2}\check{x}'H\check{x} + \check{y}'b\,. \tag{2.3.11}$$

for all feasible pairs $(\check{x}, \check{y})$. □

Corollary 2.2 (feasibility of primal QP): *A strictly convex quadratic program is feasible if and only if its dual is bounded (from above).* ○

Proof: If a strictly convex QP is feasible Theorem 2.4(i) ensures the existence of an optimal solution of its dual. Thus, its dual is bounded from above.
If a strictly convex QP is infeasible Theorem 2.4(ii) implies that its dual cannot possess an optimal solution. Since its dual is feasible, $\left(-H^{-1}g, \mathbb{0}\right)$ is always a feasible point, it must be unbounded (from above). □

In order to formulate explicit optimality conditions for quadratic programs we need the following definitions:

Definition 2.9 (active and inactive constraints): *Let a feasible quadratic program of the form* (2.3.1) *be given. A constraint* $G_i'x \geq b_i$, $1 \leq i \leq m$, *is called* active *at* $\check{x} \in \mathcal{F}$ *iff*

$$G_i'\check{x} \;=\; b_i \tag{2.3.12}$$

holds and inactive *otherwise. The (disjoint) index sets*

$$\begin{aligned} \mathbb{A}(\check{x}) &\stackrel{\text{def}}{=} \left\{i \in \{1,\ldots,m\} \mid G_i'\check{x} = b_i\right\}, \\ \mathbb{I}(\check{x}) &\stackrel{\text{def}}{=} \left\{i \in \{1,\ldots,m\} \mid G_i'\check{x} > b_i\right\} \end{aligned}$$

are called set of active constraints, *or more common* active set, *at* $\check{x}$ *and* set of inactive constraints *at* $\check{x}$, *respectively. If* x^{opt} *is an optimal solution of the quadratic program the correponding active set* $\mathbb{A}(x^{\text{opt}})$ *is called* optimal active set. ○

Definition 2.10 (working set): *Let a feasible quadratic program of the form* (2.3.1) *be given. Then arbitrary index sets*

$$\begin{aligned} \mathbb{A} &\subseteq \{1,\ldots,m\}, \\ \mathbb{I} &\stackrel{\text{def}}{=} \{1,\ldots,m\} \setminus \mathbb{A} \end{aligned}$$

are called working set *and* working set!working set complement, *respectively. Their cardinalities are denoted with*

$$\begin{aligned} n_{\mathbb{A}} &\stackrel{\text{def}}{=} |\mathbb{A}|, \\ n_{\mathbb{I}} &\stackrel{\text{def}}{=} |\mathbb{I}|. \end{aligned}$$ ○

Now we can state the following optimality conditions which are special variants of the general nonlinear case (cf. [54], [57]):

Theorem 2.5 (Karush-Kuhn-Tucker conditions): *Let* QP (2.3.1) *be a strictly convex and feasible quadratic program. Then there exists a unique* $x^{\text{opt}} \in \mathbb{R}^n$ *and at least one working set* $\mathbb{A} \subseteq \mathbb{A}(x^{\text{opt}})$ *and a vector* $y^{\text{opt}} \in \mathbb{R}^m$ *which satisfy the following conditions:*

$$Hx^{\text{opt}} - G_{\mathbb{A}}'y_{\mathbb{A}}^{\text{opt}} = -g, \tag{2.3.13a}$$
$$G_{\mathbb{A}}x^{\text{opt}} = b_{\mathbb{A}}, \tag{2.3.13b}$$
$$G_{\mathbb{I}}x^{\text{opt}} \geq b_{\mathbb{I}}, \tag{2.3.13c}$$
$$y_{\mathbb{I}}^{\text{opt}} = \mathbb{0}, \tag{2.3.13d}$$
$$y_{\mathbb{A}}^{\text{opt}} \geq \mathbb{0}. \tag{2.3.13e}$$

Furthermore,

(i) x^{opt} *is the* unique *global minimiser of the primal QP* (2.3.1)*,*

(ii) $(x^{\mathrm{opt}}, y^{\mathrm{opt}})$ *is an optimal solution of the dual QP* (2.3.7)*.*

○

Proof: A proof can be found in any textbook on optimisation, e.g. in [17, p. 244]. □

Note that neither the set $\mathbb{A}$ nor the dual solution y^{opt} are necessarily unique. If all rows of the matrix $G_{\mathbb{A}}$ are *linearly independent* and $\mathbb{A}$ is fixed, however, y^{opt} would be uniquely determined from Eqs. (2.3.13a) and (2.3.13b):

Lemma 2.2 (invertibility of the KKT matrix): *Let the Hessian matrix* H *be positive definite. Then the so-called* KKT matrix

$$\begin{pmatrix} H & G'_{\mathbb{A}} \\ G_{\mathbb{A}} & \mathbb{0} \end{pmatrix} \tag{2.3.14}$$

is invertible if and only if $G_{\mathbb{A}}$ *has full row rank.* ○

Proof: It is obvious that the KKT matrix is singular if $G_{\mathbb{A}}$ does not have full row rank. A straigtforward proof of the other direction can be found in [65, p. 445]. □

If $\mathbb{A} = \mathbb{A}(x^{\mathrm{opt}})$, the condition that $G_{\mathbb{A}}$ has full row rank is called *linear independence constraint qualification (LICQ)*. Unfortunately, we cannot make this assumption in general within our algorithm, as we will see in Chapter 4.

2.3.1 Parametric Quadratic Programming

Quadratic programs arising in model predictive control only depend on the current process state w_0. Its (initial) value affects the gradient and the constraint vector but does not change the Hessian and the constraint matrix, as shown in Theorem 2.2. This is exactly the situation where *parametric* quadratic programming can be applied: a (possibly infinite) sequence of QPs with constant matrices but varying vectors.

Definition 2.11 (parametric quadratic program): *The optimisation problem*

$$\mathrm{QP}(w_0): \quad \min_{x \in \mathbb{R}^n} \quad \tfrac{1}{2} x' H x + x' g(w_0) \tag{2.3.15a}$$

$$\text{s.t.} \quad G x \geq b(w_0)\,, \tag{2.3.15b}$$

with $H \in \mathbb{R}^{n \times n}$*,* $G \in \mathbb{R}^{m \times n}$*,* $w_0 \in \mathbb{R}^{n_x}$ *and*

$$g(w_0) \stackrel{\mathrm{def}}{=} h + F' w_0\,, \tag{2.3.16a}$$

$$b(w_0) \stackrel{\mathrm{def}}{=} l + E w_0\,, \tag{2.3.16b}$$

(with $F \in \mathbb{R}^{n_x \times n}$*,* $E \in \mathbb{R}^{m \times n_x}$*,* $h \in \mathbb{R}^n$*,* $l \in \mathbb{R}^m$*) is called a* parametric quadratic program. ○

For an arbitrary but fixed w_0 we yield an ordinary quadratic program of the form (2.3.1) and therefore all definitions and results presented so far also carry over to a parametric quadratic program. But since the gradient vector $g(w_0)$ and the constraint vector $b(w_0)$ are both affine functions of the current process state w_0, the feasible set (Definition 2.7), its optimal solution (Theorem 2.5), the set of active and inactive constraints at a certain point (Definition 2.9) as well as its dual (Definition 2.8) also depend on w_0. Therefore these quantities are written as $\mathcal{F}(w_0)$, $x^{\mathrm{opt}}(w_0)$, $\mathbb{A}\big(w_0, x^{\mathrm{opt}}(w_0)\big)$, $\mathbb{I}\big(w_0, x^{\mathrm{opt}}(w_0)\big)$, and $\mathrm{QP}^{\mathrm{dual}}(w_0)$, respectively—but, for notational convenience, we will sometimes drop this dependence when it is clear from the context.
Variations of the constraint vector may lead to infeasible QPs for certain values of w_0 and thus we introduce the following

Definition 2.12 (set of feasible parameters): *The set*

$$\mathcal{P} \overset{\mathrm{def}}{=} \big\{ w_0 \in \mathbb{R}^{n_x} \mid \mathcal{F}(w_0) \neq \emptyset \big\} \tag{2.3.17}$$

is called set of feasible parameters *of a parametric quadratic program.* ○

It has some special properties which are crucial for the online active set strategy presented in this thesis:

Theorem 2.6 (convexity and closedness of the set of feasible parameters): *The set of feasible parameters of a parametric quadratic program* $\mathrm{QP}(w_0)$ *as defined in Definition 2.12 is* convex[6] *and* closed. ○

Proof: In order to prove convexity of $\mathcal{P}$, we have to show: if two arbitrary but fixed quadratic programs $\mathrm{QP}\big(w_0^{(1)}\big)$ and $\mathrm{QP}\big(w_0^{(2)}\big)$ are feasible, i.e. $w_0^{(1)}$, $w_0^{(2)} \in \mathcal{P}$, also every quadratic program $\mathrm{QP}\big(\tau w_0^{(1)} + (1-\tau) w_0^{(2)}\big)$, $\tau \in [0,1] \subset \mathbb{R}$, is feasible, which means $\tau w_0^{(1)} + (1-\tau) w_0^{(2)} \in \mathcal{P}$.
If $\mathrm{QP}\big(w_0^{(1)}\big)$ and $\mathrm{QP}\big(w_0^{(2)}\big)$ are feasible there exist $\check{x}^{(1)}$, $\check{x}^{(2)} \in \mathbb{R}^n$ such that

$$G\check{x}^{(1)} \geq b\big(w_0^{(1)}\big) \quad \text{and} \quad G\check{x}^{(2)} \geq b\big(w_0^{(2)}\big)$$

hold. By multiplying these inequalities by $\tau \in [0,1]$ and $(1-\tau)$, respectively, and adding the results together

$$\tau G\check{x}^{(1)} + (1-\tau) G\check{x}^{(2)} \geq \tau b\big(w_0^{(1)}\big) + (1-\tau) b\big(w_0^{(2)}\big)$$

is obtained (since both τ and $(1-\tau)$ are nonnegative). Substituting Eq. (2.3.16b) yields

$$\begin{aligned} G\big(\tau\check{x}^{(1)} + (1-\tau)\check{x}^{(2)}\big) &\geq \big(\tau l + (1-\tau) l\big) + E\big(\tau w_0^{(1)} + (1-\tau) w_0^{(2)}\big) \\ &= b\big(\tau w_0^{(1)} + (1-\tau) w_0^{(2)}\big) \end{aligned}$$

which shows $\tau\check{x}^{(1)} + (1-\tau)\check{x}^{(2)} \in \mathcal{F}\big(\tau w_0^{(1)} + (1-\tau) w_0^{(2)}\big)$ and hence $\tau w_0^{(1)} + (1-\tau) w_0^{(2)} \in \mathcal{P}$.

[6] See Definition A.1

Second, we show (similar to [10]) that $\mathcal{P}$ is closed, i.e. its complement $\mathbb{R}^{n_x} \setminus \mathcal{P}$ is open: Corollary 2.2 shows that $\check{w}_0 \in \mathbb{R}^{n_x} \setminus \mathcal{P}$ is equivalent to the unboundedness of $\mathrm{QP}^{\mathrm{dual}}(\check{w}_0)$. Moreover,

$$\mathrm{QP}^{\mathrm{dual}}(\check{w}_0) \text{ unbounded} \iff \exists\, \check{y} \in \mathbb{R}^m : \ \check{y} \geq \mathbb{0} \ \wedge \ \check{y}' b(\check{w}_0) > 0 \tag{2.3.18}$$

obviously holds. For fixed $\check{y} \geq \mathbb{0}$, the value $\check{y}_0' b(\check{w}_0)$ depends *continuously* on $\check{w}_0$ as $b(\check{w}_0)$ depends affinely on $\check{w}_0$. Thus, there exists a neighbourhood $\mathcal{N}(\check{w}_0)$ of $\check{w}_0$ such that

$$\check{y}' b(\hat{w}_0) > 0 \quad \forall\, \hat{w}_0 \in \mathcal{N}(\check{w}_0)\,. \tag{2.3.19}$$

Since $\check{w}_0$ was arbitrary, this proves that $\mathbb{R}^{n_x} \setminus \mathcal{P}$ is open and therefore $\mathcal{P}$ is closed. □

The set of feasible parameters $\mathcal{P}$ is not only convex and closed but it also can be subdivided into a special collection of polyhedra[7], the so-called *critical regions* [8]:

Definition 2.13 (critical region): *Let a strictly convex parametric quadratic program* $\mathrm{QP}(w_0)$ *with the set of feasible parameters* $\mathcal{P}$ *be given. Moreover, let* $x^{\mathrm{opt}}(w_0)$, $w_0 \in \mathcal{P}$, *denote its unique optimal (primal) solution and* $\mathbb{A}\big(w_0, x^{\mathrm{opt}}(w_0)\big)$ *the corresponding active set (see Definition 2.9). Then, for every index set* $\mathbb{A} \subseteq \{1, \ldots, m\}$, *the set*

$$\mathrm{CR}_{\mathbb{A}} \stackrel{\mathrm{def}}{=} \big\{ w_0 \in \mathcal{P} \mid \mathbb{A} = \mathbb{A}\big(w_0, x^{\mathrm{opt}}(w_0)\big) \big\} \tag{2.3.20}$$

○

is called a critical region *of* $\mathcal{P}$.

Theorem 2.7 (partition of the set of feasible parameters): *For a strictly convex parametric quadratic program* $\mathrm{QP}(w_0)$ *the following hold:*

(i) All closures of critical regions $\mathrm{cl}\,(\mathrm{CR}_{\mathbb{A}_i})$ *are closed polyhedra*[7] *with pairwise disjoint interiors.*

(ii) The set of feasible parameters $\mathcal{P}$ *can be subdivided into a finite number of closures of critical regions:*

$$\mathcal{P} \ = \ \bigcup_{i=1}^{2^m} \mathrm{cl}\,(\mathrm{CR}_{\mathbb{A}_i})\,, \quad \mathbb{A}_i \subseteq \{1, \ldots, m\}\,. \tag{2.3.21}$$

○

Proof: We only prove this theorem for the situation in which the linear independence constraint qualification (LICQ) is satisfied for all $w_0 \in \mathcal{P}$; an extension to the general case can be found in [60].
(i): Since this first part is trivial for empty critical regions we assume without loss of generality that $\mathrm{CR}_{\mathbb{A}} \neq \emptyset$ for an arbitrary $\mathbb{A} \subseteq \{1, \ldots, m\}$. This means that there exists a $w_0 \in \mathcal{P}$ for which $\mathbb{A} = \mathbb{A}\big(w_0, x^{\mathrm{opt}}(w_0)\big)$ is the active set corresponding to an optimal solution x^{opt} of $\mathrm{QP}(w_0)$ satisfying the optimality conditions of Theorem 2.5. By substituting

$$x^{\mathrm{opt}}(w_0) = H^{-1} G_{\mathbb{A}}' y_{\mathbb{A}}^{\mathrm{opt}}(w_0) - H^{-1} g(w_0) \tag{2.3.22}$$

[7] See Definition A.3

they can be written as

$$\begin{aligned} G_{\mathbb{A}} H^{-1} G_{\mathbb{A}}' y_{\mathbb{A}}^{\mathrm{opt}}(w_0) &= b_{\mathbb{A}}(w_0) + G_{\mathbb{A}} H^{-1} g(w_0)\,, && \text{(2.3.23a)} \\ G_{\mathbb{I}} H^{-1} G_{\mathbb{A}}' y_{\mathbb{A}}^{\mathrm{opt}}(w_0) &> b_{\mathbb{I}}(w_0) + G_{\mathbb{A}} H^{-1} g(w_0)\,, && \text{(2.3.23b)} \\ y_{\mathbb{I}}^{\mathrm{opt}}(w_0) &= \mathbb{0}\,, && \text{(2.3.23c)} \\ y_{\mathbb{A}}^{\mathrm{opt}}(w_0) &\geq \mathbb{0}\,. && \text{(2.3.23d)} \end{aligned}$$

Note that the third KKT condition (2.3.13c) is strictly satisfied as $\mathbb{A} = \mathbb{A}\big(w_0, x^{\mathrm{opt}}(w_0)\big)$. This leads to

$$\begin{aligned} y_{\mathbb{A}}^{\mathrm{opt}}(w_0) &= \left(G_{\mathbb{A}} H^{-1} G_{\mathbb{A}}'\right)^{-1} \left(b_{\mathbb{A}}(w_0) + G_{\mathbb{A}} H^{-1} g(w_0)\right)\,, && \text{(2.3.24a)} \\ G_{\mathbb{I}} H^{-1} G_{\mathbb{A}}' y_{\mathbb{A}}^{\mathrm{opt}}(w_0) &> b_{\mathbb{I}}(w_0) + G_{\mathbb{A}} H^{-1} g(w_0)\,, && \text{(2.3.24b)} \\ y_{\mathbb{I}}^{\mathrm{opt}}(w_0) &= \mathbb{0}\,, && \text{(2.3.24c)} \\ y_{\mathbb{A}}^{\mathrm{opt}}(w_0) &\geq \mathbb{0}\,, && \text{(2.3.24d)} \end{aligned}$$

in which $G_{\mathbb{A}} H^{-1} G_{\mathbb{A}}'$ is invertible because of the LICQ. Finally, by substituting Eqs. (2.3.16a) and (2.3.16b) we obtain that $\mathbb{A}$ is the active set of an optimal solution as long as the following linear inequalities hold:

$$\begin{aligned} &\Big(G_{\mathbb{I}} H^{-1} G_{\mathbb{A}}' \left(G_{\mathbb{A}} H^{-1} G_{\mathbb{A}}'\right)^{-1} \left(E_{\mathbb{A}} + G_{\mathbb{A}} H^{-1} F'\right) && \text{(2.3.25a)} \\ &\qquad - \left(E_{\mathbb{I}} + G_{\mathbb{A}} H^{-1} F'\right)\Big) w_0 > \; G_{\mathbb{I}} H^{-1} G_{\mathbb{A}}' \left(G_{\mathbb{A}} H^{-1} G_{\mathbb{A}}'\right)^{-1} l_{\mathbb{A}} + l_{\mathbb{I}}\,. \\ &\left(G_{\mathbb{A}} H^{-1} G_{\mathbb{A}}'\right)^{-1} \left(E_{\mathbb{A}} + G_{\mathbb{A}} H^{-1} F'\right) w_0 \geq \; \left(G_{\mathbb{A}} H^{-1} G_{\mathbb{A}}'\right)^{-1} l_{\mathbb{A}}\,. && \text{(2.3.25b)} \end{aligned}$$

□

Thus, we derived an explicit representation of a (nonempty) critical region $\mathrm{CR}_{\mathbb{A}}$. Its closure with respect to the standard topology of $\mathbb{R}^{n_x}$ is obtained by replacing "$>$" with "$\geq$" in Eqs. (2.3.25) and is thus a closed polyhedron.
By construction, the strictly convex quadratic program $\mathrm{QP}(w_0)$ is feasible for every $w_0 \in \mathcal{P}$ which guarantees the existence of an unique optimal solution $x^{\mathrm{opt}}(w_0)$, according to Theorem 2.5, and a corresponding unique optimal active set. Therefore, the critical regions are pairwise disjoint and hence their closures can only overlap at their boundaries.

(ii): Since an optimal active set exists for every $w_0 \in \mathcal{P}$, the set of feasible parameters $\mathcal{P}$ equals the union of all critical regions. $\mathcal{P}$ also equals the union of all *closures* of critical regions as it is closed (i.e. $\mathcal{P} = \mathrm{cl}\,(\mathcal{P})$, cf. Theorem 2.6). The number of closures of critical regions is finite because the number of index sets $\mathbb{A}$ is 2^m.

We will see in Chapter 4 that these facts—namely the convexity of the set of feasible parameters as well as its partition into closed, convex, polyhedral critical regions—are very important ingredients for the proposed online active set strategy; they are depicted in Figure 2.3.1.

The proof of Theorem 2.7 also gives us some insight into the structure of the optimal solution $x^{\mathrm{opt}}(w_0)$ of the parametric quadratic program $\mathrm{QP}(w_0)$. We summarise this important result in the following

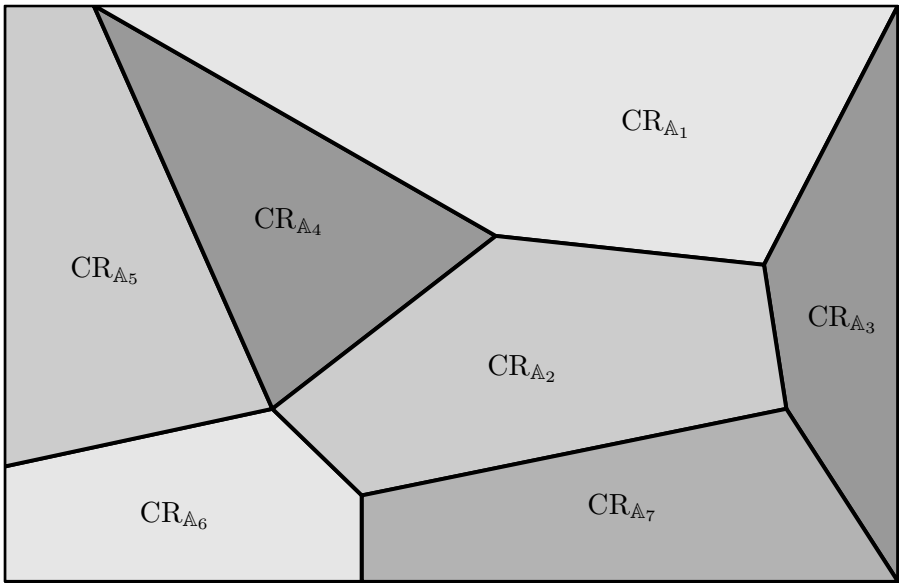


Figure 2.2: Partition of the set of feasible parameters $\mathcal{P}$ into critical regions.

Theorem 2.8 (piecewise affine optimal solution): *Let a strictly convex parametric quadratic program* $\mathrm{QP}(w_0)$ *and its set of feasible parameters* $\mathcal{P}$ *be given. Then the following is true:*

(i) Its optimal solution is a piecewise affine *and* continuous *function*

$$x^{\mathrm{opt}} : \mathcal{P} \longrightarrow \mathbb{R}^n ,$$

(ii) its optimal objective function value is a piecewise quadratic *and* continuous *function*

$$\begin{aligned} \nu^{\mathrm{opt}} : \quad \mathcal{P} &\longrightarrow \mathbb{R} \\ w_0 &\longmapsto \frac{1}{2} x^{\mathrm{opt}}(w_0)' H x^{\mathrm{opt}}(w_0) + x^{\mathrm{opt}}(w_0)' g(w_0) . \end{aligned}$$

The notion "piecewise" means that there exists a finite partition of $\mathcal{P}$ *into polyhedral critical regions such that the restrictions of* x^{opt} *and* ν^{opt} *to each critical region are affine or quadratic, respectively.* ○

Proof: Again, we only prove these results for the situation in which the linear independence constraint qualification (LICQ) is satisfied for all $w_0 \in \mathcal{P}$ and refer to [60] for an extension to the general case.
Combining Eqs. (2.3.22) and (2.3.24a) yields an explicit affine representation of $x^{\mathrm{opt}}(w_0)$ over each closure of a critical region. Thus, x^{opt} is piecewise affine over $\mathcal{P}$ and continuous over each closure of a critical region. The boundary between two closures of critical regions belongs to both closed regions and as the optimum is unique, the solution must also be continuous across these boundaries (see also [8]).
The second part of the theorem follows trivially from the first. □

Continuity of the optimal solution function x^{opt} was already stated by Fiacco [32] in the context of sensitivity analysis in nonlinear programming; Zafiriou [92] proved that x^{opt} is piecewise affine in order to obtain stability results. Our formulation which explicitly uses a polyhedral partition of $\mathcal{P}$ was introduced by Bemporad et al. [8] (and refined by Mayne et Rakovic [60]) in order to derive a practical method for the offline solution of parametric quadratic programs arising from MPC problems.

2.3.2 Explicit (Offline) Solution of Parametric Quadratic Programs

The third step of Algorithm 2.1 requires the solution of an open-loop optimal control problem at each sampling instant during the runtime of the controlled process. Although this task reduces to a simple optimisation problem if the process model (and the constraints) is linear and the objective function is quadratic, namely a (strictly) convex quadratic program, it may become computationally prohibitive if very short sampling times are necessary. Thus, instead of solving each quadratic program during the runtime of the process using a standard QP solver (see Chapter 3), [8] proposed to solve all possibly occuring QPs beforehand, i.e. solving the parametric quadratic program $\mathrm{QP}(w_0)$, and look up the solution when needed. Theorem 2.8 guarantees that only a finite number of critical regions and the correponding explicit affine representation of the solution have to be stored, making this explicit, or "offline", approach tractable. Since available (online) computing power is very limited (and memory quite cheap) in most practical applications, explicit model predictive control soon became very popular among the engineers of the MPC community. We outline the main concept in Algorithm 2.2.

Skipping technical details, we briefly explain the offline step (0) and the online step (3) of the explicit linear MPC approach:
The parametric quadratic programm $\mathrm{QP}(w_0)$, also referred to as "*multi-parametric*" quadratic program to emphasise that w_0 is usually nonscalar, is solved as follows [8]: first, an arbitrary parameter $\hat{w}_0$ in the interior of a critical region is determined by solving an appropriate *linear program (LP)*. Then the quadratic program $\mathrm{QP}(\hat{w}_0)$ is solved which enables the determination of a polyhedral representation $\{w \in \mathcal{P} \mid \hat{A}w \geq \hat{b}\}$ of the critical region $\mathrm{CR}_{\hat{\mathbb{A}}}$ with $\hat{w}_0 \in \mathrm{CR}_{\hat{\mathbb{A}}}$ as well as an affine representation $\hat{C}w + \hat{d}$ of the optimal solution over $\mathrm{CR}_{\hat{\mathbb{A}}}$. Afterwards, the complement $\mathcal{P} \setminus \mathrm{CR}_{\hat{\mathbb{A}}}$ can easily be divided into a partition of $\hat{m} \stackrel{\mathrm{def}}{=} \dim \hat{b}$ convex polyhedra $\mathcal{P}_1, \ldots, \mathcal{P}_{\hat{m}}$ by successive changes of the defining inequalities $\hat{A}_i w \leq \hat{b}_i$ into $\hat{A}_i w > \hat{b}_i$. Finally, these steps are recursively performed for $\mathcal{P}_1, \ldots, \mathcal{P}_{\hat{m}}$. Further refinements such as reduction of the number of QPs to be solved and linear dependence handling are decribed in [77], [75].

Step (3) can be implemented straightforward by just checking all polyhedral representations, i.e. checking if $\hat{A}w_0 \geq \hat{b}$, until the correct critical region is found and then calculating the optimal solution via $\hat{C}w_0 + \hat{d}$. Since the number of critical regions may become very large, [78] proprosed the construction of a binary search tree (however, this idea does not reduce the offline complexity).

Although the explicit approach sounds quite appealing, it has a main drawback: since the number of possible critical regions grows exponentially in the number of constraints (up to 2^m different active sets) it is limited to *low dimensional* parameter spaces $\mathcal{P}$, i.e. to process

models comprising only very few states[8]. Otherwise the offline computation and storage requirements as well as the online effort for finding the correct critical region soon become prohibitively large. A further serious problem in practice is that online tuning becomes nearly impossible as the offline computation time blows up.
Therefore, several techniques for reducing the offline complexity at the expense of a *suboptimal* online performance and slight constraint violations were presented in [7], [51], [78]. The main idea is to combine several "small" critical regions to a "bigger" one. A different procedure called *partial enumeration* is proposed in [68]: although exponentially many critical regions exist only a very small fraction of them really becomes relevant during the runtime of the process. Thus, instead of calculating all critical regions, only (a guess of) this fraction is calculated and stored in a cache. If the critical region of the current QP belongs to the cache its affine representation of the optimal solution is used. Otherwise, while applying some suboptimal heuristical control action, the QP is solved online using a standard QP solver and the corresponding critical region is added to the cache, afterwards.

Algorithm 2.2 (explicit linear model predictive control concept)

`input:` discrete-time linear open-loop optimal control problem $\mathrm{OCP}_{\mathrm{lin}}^{\mathrm{disc}}(k_0)$, sequence of sampling instants $t_0, t_1, \ldots, t_{n_{\mathrm{sample}}}$

`output:` piecewise defined optimal process inputs $u^{\mathrm{opt}} : [0, t_{\mathrm{end}}] \rightarrow \mathbb{R}^{n_u}$

(0) Compute and store an explicit piecewise affine representation of the solution x^{opt} to the parametric quadratic program $\mathrm{QP}(w_0)$ (before start of process!).

(1) Set $i \leftarrow 0$.

(2) Obtain current process state $w_0(t_i)$.

(3) (a) Determine a critical region CR_{A_i} such that $w_0(t_i) \in \mathrm{CR}_{A_i}$.

(b) Obtain first optimal process input $u_{k_0} = (x_1^{\mathrm{opt}}, \ldots, x_{n_u}^{\mathrm{opt}})'$ from the explicit affine representation of x^{opt} over the critical region CR_{A_i}.

(4) Set $u^{\mathrm{opt}}(t) \stackrel{\mathrm{def}}{=} u_{k_0} \ \forall\, t \in [t_i, t_{i+1}]$ and apply u_{k_0} to the process until t_{i+1}.

(5) `if` $i = n_{\mathrm{sample}} - 1$:

`stop!`

`else`

Set $i \leftarrow i + 1$ and continue with step (2).

[8] State space dimensions of about five seem to be currently tractable via explicit MPC.

Chapter 3

Existing Methods for Solving Quadratic Programs

Having introduced the explicit, or offline approach for treatment of parametric quadratic programs, this chapter is devoted to a short summary of existing solution methods for quadratic programs. All methods to be presented are able to solve quadratic programs arising in the *online* context of model predictive control but (almost) none of them was written with this application in mind. We describe them for two reasons: first, our online active set strategy is based on the so-called *null space* based *primal active set method* and also inherits some features of the *dual active set approach*. Second, we will use an active-set method as comparison in several MPC benchmark tests in Chapters 5 and 6, as such methods are widely used in practice. Also *interior-point methods* are briefly mentioned for completeness.

3.1 Primal Active Set Methods

Let us consider the task of solving a strictly convex quadratic program, as defined in Definition 2.6. If the inequality constraints which are active at the solution, say $\mathbb{A} \stackrel{\text{def}}{=} \mathbb{A}(x^{\text{opt}})$, are known beforehand this problem reduces to the following *equality constrained* quadratic program:

$$\mathrm{QP}_{\text{ec}}: \qquad \min_{x \in \mathbb{R}^n} \quad \tfrac{1}{2} x' H x + x' g \tag{3.1.1a}$$

$$\text{s.t.} \quad G_{\mathbb{A}} x \;=\; b_{\mathbb{A}}\,. \tag{3.1.1b}$$

Without loss of generality, we assume that the matrix $G_{\mathbb{A}}$ has full row rank because otherwise a suitable linearly independent subset of active constraints could be chosen. If QP_{ec} is also feasible Theorem 2.5 implies the following necessary and sufficient condition for the optimal solution:

$$\begin{pmatrix} H & G_{\mathbb{A}}' \\ G_{\mathbb{A}} & \mathbb{0} \end{pmatrix} \begin{pmatrix} x^{\text{opt}} \\ -y_{\mathbb{A}}^{\text{opt}} \end{pmatrix} \;=\; \begin{pmatrix} -g \\ b_{\mathbb{A}} \end{pmatrix}. \tag{3.1.2}$$

Thus, solving QP_{ec} becomes equivalent to the solution of a linear system whose matrix is invertible, according to Lemma 2.2. Since this is an rather trivial task *active set methods*

aim at reducing a QP (2.3.1) to a $\mathrm{QP}_{\mathrm{ec}}$ (3.1.1) by identifying (a suitable subset of) the optimal active set. An early active set algorithm for (general) quadratic programs was given in [33]. The basic idea is indeed much older since also the famous *simplex method* [22] for *linear programming* can be interpreted as specialised active set method (see e.g. [41]); and the first implementations for the solution of quadratic programs were extensions of the simplex method [88], [22].

Primal active set methods start with a feasible point $x^{(0)}$ (if such a point exists) and a working set $\mathbb{A}^{(0)} \subseteq \mathbb{A}(x^{(0)})$ which serves as an *initial guess* for the optimal active set. Then a sequence of feasible iterates $x^{(k)}$ and correponding working sets $\mathbb{A}^{(k)}$, $k \geq 0$, are determined: assuming that $\mathbb{A}^{(k)}$ is indeed an *optimal* working set, the next iterate

$$x^{(k+1)} \stackrel{\text{def}}{=} x^{(k)} + \Delta x^{(k)} \tag{3.1.3}$$

is the optimal solution if and only if it solves Eq. (3.1.2):

$$\begin{pmatrix} H & G'_{\mathbb{A}^{(k)}} \\ G_{\mathbb{A}^{(k)}} & \mathbb{0} \end{pmatrix} \begin{pmatrix} x^{(k+1)} \\ -y^{(k+1)}_{\mathbb{A}^{(k)}} \end{pmatrix} = \begin{pmatrix} -g \\ b_{\mathbb{A}^{(k)}} \end{pmatrix} \tag{3.1.4}$$

$$\iff \quad \begin{pmatrix} H & G'_{\mathbb{A}^{(k)}} \\ G_{\mathbb{A}^{(k)}} & \mathbb{0} \end{pmatrix} \begin{pmatrix} \Delta x^{(k)} \\ -y^{(k+1)}_{\mathbb{A}^{(k)}} \end{pmatrix} = - \begin{pmatrix} Hx^{(k)} + g \\ \mathbb{0} \end{pmatrix}. \tag{3.1.5}$$

The reason why system (3.1.5) is solved, instead of (3.1.4), is that $\mathbb{A}^{(k)}$ is only a *guess* for the optimal active set. Thus, when moving from $x^{(k)}$ to $x^{(k+1)}$ along $\Delta x^{(k)}$ it may happen that an inactive constraint becomes violated which renders $x^{(k+1)}$ infeasible. In order to avoid this (primal) infeasibility, the next iterate $x^{(k+1)}$ is chosen as

$$x^{(k+1)} \stackrel{\text{def}}{=} x^{(k)} + \tau^{(k)} \Delta x^{(k)}, \quad \tau^{(k)} \in \mathbb{R}_{\geq 0} \tag{3.1.6a}$$

with

$$\tau^{(k)} \stackrel{\text{def}}{=} \min \left\{ 1, \; \min_{i \notin \mathbb{A}^{(k)}} \left\{ \frac{b_i - G'_i x^{(k)}}{G'_i \Delta x^{(k)}} \;\middle|\; G'_i \Delta x^{(k)} < 0 \right\} \right\}. \tag{3.1.6b}$$

This choice of $\tau^{(k)}$ ensures that

$$G_{\mathbb{I}^{(k)}} x^{(k+1)} \;=\; G_{\mathbb{I}^{(k)}} x^{(k)} + \tau^{(k)} G^{(k)}_{\mathbb{I}^{(k)}} \Delta x^{(k)} \;\geq\; b_{\mathbb{I}^{(k)}}, \quad \mathbb{I}^{(k)} \stackrel{\text{def}}{=} \{1, \ldots, m\} \setminus \mathbb{A}^{(k)} \tag{3.1.7}$$

holds, while $G_{\mathbb{A}^{(k)}} x^{(k+1)} = b_{\mathbb{A}^{(k)}}$ is guaranteed by the choice of $\Delta x^{(k)}$ (cf. Eq. (3.1.4)). If Eq. (3.1.6b) leads to $\tau^{(k)} < 1$ the constraint which caused this limitation of $\tau^{(k)}$—the so-called *blocking constraint*—is *added* to the working set, yielding the next working set $\mathbb{A}^{(k+1)}$, and the next iterate is determined in the above mentioned manner.
If there is no blocking constraint, i.e. $\tau^{(k)} = 1$, a *full step* is taken implying that the optimal solution of the quadratic program (2.3.1) is found provided that $\mathbb{A}^{(k)}$ is really the optimal active set. We can check this by looking at the dual solution vector $y^{(k+1)}_{\mathbb{A}^{(k)}}$: if the *unique* optimal solution $x^{(k)}$ of $\mathrm{QP}_{\mathrm{ec}}$ subject to the equality constraints $G_{\mathbb{A}^{(k)}} x^{(k)} = b_{\mathbb{A}^{(k)}}$ is found the next step direction $\Delta x^{(k+1)}$ must be zero. Therefore Eq. (3.1.5) shows that the first optimality condition (2.3.13a) of Theorem 2.5 is satisfied. Moreover, conditions (2.3.13b) and (2.3.13c) are fulfilled by construction; condition (2.3.13d) can be met by setting

$y^{(k+1)}_{\mathbb{I}^{(k)}} \stackrel{\text{def}}{=} \mathbb{0}$. Thus, according to the last optimaltiy condition (2.3.13e), the current iterate $x^{(k+1)} = x^{(k)}$ is indeed optimal for the inequality constraint quadratic program (2.3.1) if and only if each component of $y^{(k+1)}_{\mathbb{A}^{(k)}}$ is nonnegative. If this is the case we have found the optimal solution of (2.3.1), otherwise we *drop* one the constraints corresponding to a negative component of $y^{(k+1)}_{\mathbb{A}^{(k)}}$ from the current working set and proceed with determing a new step direction $\Delta x^{(k+1)}$, again[1].
A formal summary of the primal active set method is given in Algorithm 3.1 (cf. [65]):

Algorithm 3.1 (primal active set method)

`input:` strictly convex quadratic program QP of the form (2.3.1), initial guesses for solution $x^{(0)}$ and optimal active set $\mathbb{A}^{(0)}$ (both optional)

`output:` optimal solution x^{opt} of QP and working set $\mathbb{A}$ as defined in Theorem 2.5 (or message that QP is infeasible)

(1) Set $k \leftarrow 0$ and obtain feasible starting point $x^{(0)}$ and working set $\mathbb{A}^{(0)} \subseteq \mathbb{A}(x^{(0)})$.
If such a point does not exists: `stop (QP infeasible)!`

(2) Calculate $\Delta x^{(k)}$ and $y^{(k+1)}_{\mathbb{A}^{(k)}}$ from Eq. (3.1.5).

(3) `if` $\Delta x^{(k)} = \mathbb{0}$:
  `if` $y^{(k+1)}_{\mathbb{A}^{(k)}} \geq \mathbb{0}$:
    Optimal solution of QP found: set $x^{\mathrm{opt}} \leftarrow x^{(k)}$ and $\mathbb{A} \leftarrow \mathbb{A}^{(k)}$. `stop!`
  `else`
    Drop a constraint $j \in \mathbb{A}^{(k)}$ with $y^{(k+1)}_j < 0$ from working set,
    i.e. $\mathbb{A}^{(k+1)} \leftarrow \mathbb{A}^{(k)} \setminus \{j\}$, and continue with step (2).

(4) Compute step length $\tau^{(k)}$ via Eq. (3.1.6b) and set $x^{(k+1)} \leftarrow x^{(k)} + \tau^{(k)} \Delta x^{(k)}$.

(5) `if` $\tau^{(k)} < 1$:
    Add a *blocking constraint* $j = \underset{i \notin \mathbb{A}^{(k)}}{\arg\min} \frac{b_i - G_i' x^{(k)}}{G_i' \Delta x^{(k)}}$ to working set,
    i.e. $\mathbb{A}^{(k+1)} \leftarrow \mathbb{A}^{(k)} \cup \{j\}$.
  `else`
    Set $\mathbb{A}^{(k+1)} \leftarrow \mathbb{A}^{(k)}$.

(6) Set $k \leftarrow k+1$ and continue with step (2).

[1] It can be shown, see e.g. [65, p. 459–461], that the dropped constraint remains satisfied along the new step direction $\Delta x^{(k+1)}$.

Some steps of Algorithm 3.1 need further attention:

Initialisation: If no feasible starting point is given by the user the algorithm has to find one in the first step, also known as *Phase I* (see e.g. [34]). The idea is to formulate an auxiliary (linear) problem for which a feasible point is known and whose solution delivers a feasible starting point for the original problem. For our QP formulation such a *phase I*, or *feasibility problem* can be the following

$$\min_{p\in\mathbb{R}^{m_p},\, x\in\mathbb{R}^n} \quad \mathbb{1}'p \tag{3.1.8a}$$

$$\text{s.t.} \quad G^+x + p \;\geq\; b^+\,, \tag{3.1.8b}$$

$$G^-x \geq\; b^-\,, \tag{3.1.8c}$$

$$p \;\geq\; \mathbb{0}\,, \tag{3.1.8d}$$

where (3.1.8b) describes a relaxation of the m_p, $0 \leq m_p \leq m$, constraints with positive components of the constraint vector, i.e. $b^+ > \mathbb{0}$, and (3.1.8c) describes the $(m - m_p)$ constraints with nonpositive components of the constraint vector, i.e. $b^- \leq \mathbb{0}$. Then the choice

$$x^{(0)} \;\stackrel{\text{def}}{=}\; \mathbb{0}\,, \quad p^{(0)} \;\stackrel{\text{def}}{=}\; b^+ \tag{3.1.9}$$

is obviously a feasible point for the auxiliary problem (3.1.8). Furthermore, the original problem (2.3.1) is feasible if and only if the auxiliary problem has an optimal objective value of 0. If that is the case all components of p must be zero and the remaining optimisation variables x form a feasible starting point for the quadratic program (2.3.1). The initial working set can be chosen as a (linearly independent) subset of the active constraints at the starting point.

According to [45], "computational experience indicates that, unless a feasible point is available, on the average between one-third to one-half of the total effort required to solve a QP is expended in phase I." If, as in model predictive control, a sequence of neighbouring QPs is to be solved optimal solution and corresponding working set of the last QP can be used to initialise a primal active set solver. This *warm start* idea not only can save the phase I but also may reduce the number of iterations significantly. But due to changes of the constraint vector the former solution may become infeasible which makes a phase I neccessary and thus ruins the possible benefit of warm starts.

Dropping a constraint: If several active constraints correspond to a negative component of the dual solution vector in step (3) the question arises: which one should be removed from the working set? A common choice is to select the constraint

$$j \;=\; \arg\min_{i\in\mathbb{A}^{(k)}} y_i^{(k)}. \tag{3.1.10}$$

It "works quite well" [34] in practice "but has the disadvantage that it is susceptible to the scaling of the constraints." [65]

Linear independence of active constraints: The theoretical derivation of the primal active set algorithm is based on the assumption that matrix $G_{\mathbb{A}^{(k)}}$ has full row rank at each iteration $k \geq 0$. Provided that a linearly independent intial working set $\mathbb{A}^{(0)}$ is chosen, this assumption can only be violated when a constraint is added to working set in step (5), as the deletion of a row cannot lead to rank deficiency. Since the step direction is chosen such

that all active constraints remain satisfied for all step lengths no constraint which is linearly dependent from them can become a blocking constraint, and thus cannot be added to the working set.
However, there may be points at which the *active* set is linearly dependent, so-called *degenerated points*. At such points successive deletion and addition of constraints with zero step size in between can happen (each leaving the *working* set linearly independent). And it may be that the sequence of working sets obtained be deleting and adding constraints at such a degenerate point repeats itself after finitely many steps, a phenomenon known as *cycling*. "Fortunately, the occurence of cycling is rare" and "simple heuristic strategies almost always succeed in breaking the deadlock" [42]. In contrast, [65] states that "most QP implementations simply ignore the possibility of cycling."

Finally, we want to mention that Algorithm 3.1 *terminates after a finite number of iterations* at the optimal solution of a strictly convex and feasible quadratic program (2.3.1) provided that no cycling occurs (cf. [65, p. 466–467]).

In the next two subsections we will have a closer look at how to solve system (3.1.5) efficiently.

3.1.1 Null Space Method

Solving system (3.1.5) can be interpreted as solving an equality constrained, strictly convex quadratic program similar to $\mathrm{QP}_{\mathrm{ec}}$ (see [38]):

$$\min_{\Delta x^{(k)} \in \mathbb{R}^n} \quad \tfrac{1}{2} \Delta x^{(k)\prime} H \Delta x^{(k)} + \Delta x^{(k)\prime} (H x^{(k)} + g) \tag{3.1.11a}$$

$$\text{s. t.} \qquad G_{\mathbb{A}^{(k)}} \Delta x^{(k)} = \mathbb{0}\,. \tag{3.1.11b}$$

The equality constraint implies that a point is feasible if and only if it lies completely in the *null space*[2] of the active constraints matrix $G_{\mathbb{A}^{(k)}}$. So, if $Z^{(k)} \in \mathbb{R}^{n \times (n - n_{\mathbb{A}})}$ is a matrix whose columns form a basis of the null space of $G_{\mathbb{A}^{(k)}}$, i.e. $G_{\mathbb{A}^{(k)}} Z^{(k)} = \mathbb{0}$, every feasible point can be written as

$$\Delta x^{(k)} \;=\; Z^{(k)} \Delta x_Z^{(k)}\,, \quad \Delta x_Z^{(k)} \in \mathbb{R}^{n - n_{\mathbb{A}}}\,. \tag{3.1.12}$$

A null space basis matrix $Z^{(k)}$ can be obtained by calculating a *QR factorisation*[3] of $G'_{\mathbb{A}^{(k)}}$:

$$\begin{pmatrix} Y^{(k)} & Z^{(k)} \end{pmatrix} \begin{pmatrix} U^{(k)} \\ \mathbb{0} \end{pmatrix} \stackrel{\text{def}}{=} V^{(k)} \begin{pmatrix} U^{(k)} \\ \mathbb{0} \end{pmatrix} \;=\; G'_{\mathbb{A}^{(k)}}\,, \tag{3.1.13}$$

where $V^{(k)} \in \mathbb{R}^{n \times n}$ is an orthonormal and $U^{(k)} \in \mathbb{R}^{n_{\mathbb{A}} \times n_{\mathbb{A}}}$ an upper triangular matrix; $Y^{(k)} \in \mathbb{R}^{n \times n_{\mathbb{A}}}$ and $Z^{(k)} \in \mathbb{R}^{n \times (n - n_{\mathbb{A}})}$ are orthonormal matrices containing bases of the range and the null space of $G_{\mathbb{A}^{(k)}}$, respectively.

Substituting Eq. (3.1.12) into (3.1.11) leads to the following *unconstrained* quadratic problem:

$$\min_{\Delta x^{(k)}} \quad \tfrac{1}{2} \Delta x_Z^{(k)\prime} Z^{(k)\prime} H Z^{(k)} \Delta x_Z^{(k)} + \Delta x_Z^{(k)\prime} Z^{(k)\prime} \left(H x^{(k)} + g \right) \tag{3.1.14}$$

[2] See Definition A.4.
[3] See Theorem A.2.

whose solution is

$$\Delta x_Z^{(k)} = -\left(Z^{(k)\prime} H Z^{(k)}\right)^{-1} Z^{(k)\prime}\left(Hx^{(k)} + g\right) \tag{3.1.15a}$$

$$\iff \quad R^{(k)\prime} R^{(k)} \Delta x_Z^{(k)} = -Z^{(k)\prime}\left(Hx^{(k)} + g\right). \tag{3.1.15b}$$

Therein $R^{(k)\prime}R^{(k)}$ is the *Cholesky decomposition*[4] of the *projected Hessian matrix* $Z^{(k)\prime}HZ^{(k)}$, with an upper triangular matrix $R^{(k)} \in \mathbb{R}^{(n-n_{\mathbb{A}})\times(n-n_{\mathbb{A}})}$. Its existence is guaranteed by the positive definiteness of H and the fact that the basis matrix $Z^{(k)}$ has full column rank. Since $R^{(k)}$ is an upper triangular matrix, Eq. (3.1.15b) is easily solved via a *forward* and a *backward substitution*.

Then the associated dual solution vector can be obtained as

$$H\Delta x^{(k)} - G'_{\mathbb{A}^{(k)}} y_{\mathbb{A}^{(k)}}^{(k+1)} = -\left(Hx^{(k)} + g\right) \tag{3.1.16a}$$

$$\iff \quad y_{\mathbb{A}^{(k)}}^{(k+1)} = \left(G_{\mathbb{A}^{(k)}} G'_{\mathbb{A}^{(k)}}\right)^{-1} G_{\mathbb{A}^{(k)}}\left(HZ^{(k)}\Delta x_Z^{(k)} + Hx^{(k)} + g\right), \tag{3.1.16b}$$

$$\iff \quad U^{(k)} y_{\mathbb{A}^{(k)}}^{(k+1)} = Y^{(k)}\left(HZ^{(k)}\Delta x_Z^{(k)} + Hx^{(k)} + g\right), \tag{3.1.16c}$$

where $G_{\mathbb{A}^{(k)}} G'_{\mathbb{A}^{(k)}}$ is invertible because $G_{\mathbb{A}^{(k)}}$ has full row rank. Eq. (3.1.15b) can be solved via a backward substitution as $U^{(k)}$ is an upper triangular matrix.

The *null space method* uses Eqs. (3.1.15b) and (3.1.16c) to calculate the solution of the KKT system (3.1.5); the matrix factorisations are introduced in order to calculate a null space basis matrix, which greatly simplifies the calculation of $y_{\mathbb{A}^{(k)}}^{(k+1)}$, and to avoid explicitly inverting the projected Hessian matrix. Inverting the projected Hessian matrix as well as calculating the matrix factorisations from scratch requires $\mathcal{O}(n^3)$ floating-point operations. So, the factorisations seem to be of limited use as they change whenever a constraint is added to or deleted from the working set. But because of the simple nature of these changes, *update schemes* for Cholesky and QR decomposition were described in [36], [44], [21] which reduce the effort to obtain the changed factorisations to $\mathcal{O}(n^2)$. Thus, also the number of floating-point operations for solving the KKT system only grows quadratically in the number of optimisation variables. We will discuss these *matrix updates* in more detail in Section 4.3.3 as our online active set strategy is based on the null space approach and also makes use of them.

Two well-known implementations of the null space method for quadratic programming are `qpsol` [62] and `qpopt` [63]. We also note that the null space method is applicable as long as the *projected* Hessian is positive definite, which not necessarily requires the Hessian matrix to be positive definite; an extension to *indefinite quadratic programs* is described in [40]. Furthermore, since $Z^{(k)}$ is chosen orthonormal, the condition number[5] of the projected Hessian is the same as that of the Hessian itself. This makes the null space method numerically more stable than the range space method, which we present next.

[4] See Theorem A.1.
[5] See Definition A.1.

3.1.2 Range Space Method

Assuming the Hessian matrix H to be positive definite, the KKT system (3.1.5) can also be solved by calculating the inverse of the KKT matrix explicitly:

$$\begin{pmatrix} H & G'_{\mathbb{A}^{(k)}} \\ G_{\mathbb{A}^{(k)}} & \mathbb{0} \end{pmatrix} \begin{pmatrix} H^{-1} - H^{-1}G'_{\mathbb{A}^{(k)}}W^{(k)}G_{\mathbb{A}^{(k)}}H^{-1} & H^{-1}G'_{\mathbb{A}^{(k)}}W^{(k)} \\ W^{(k)}G_{\mathbb{A}^{(k)}}H^{-1} & -W^{(k)} \end{pmatrix} = \mathrm{Id}, \quad (3.1.17)$$

where $W^{(k)} \stackrel{\text{def}}{=} \left(G_{\mathbb{A}^{(k)}}H^{-1}G'_{\mathbb{A}^{(k)}}\right)^{-1} \in \mathbb{R}^{n_{\mathbb{A}} \times n_{\mathbb{A}}}$. Exploiting common subexpressions leads to the following solution formulae for system (3.1.5):

$$y^{(k+1)}_{\mathbb{A}^{(k)}} = W^{(k)}G_{\mathbb{A}^{(k)}}H^{-1}\left(Hx^{(k)} + g\right), \quad (3.1.18a)$$

$$\Delta x^{(k)} = H^{-1}G'_{\mathbb{A}^{(k)}}y^{(k+1)}_{\mathbb{A}^{(k)}} - H^{-1}\left(Hx^{(k)} + g\right) \quad (3.1.18b)$$

$$= \left(H^{-1}G'_{\mathbb{A}^{(k)}}W^{(k)}G_{\mathbb{A}^{(k)}}H^{-1} - H^{-1}\right)\left(Hx^{(k)} + g\right). \quad (3.1.18c)$$

This representation of the solution is called *range space approach* because the Hessian matrix is projected to the *range space* of the active constraints. This form has the disadvantage that the condition number of $G_{\mathbb{A}^{(k)}}H^{-1}G'_{\mathbb{A}^{(k)}}$ is that of the Hessian *multiplied* with the *squared condition number* of $G'_{\mathbb{A}^{(k)}}$, which renders the range space method inappropriate if the active constraints matrix is ill-conditioned; the same holds if the Hessian matrix H is nearly singular.

On the other hand, this approach becomes attractive if the Hessian matrix is easy to invert and the number of constraints in the working set remains small. This is in contrast to the null space approach where the dimension of the projected Hessian $Z^{(k)\prime}HZ^{(k)}$, and thus the number of correponding linear algebra operations, *decreases* with the number of active constraints.

Eqs. (3.1.18) are not directly applied to calculate the primal step direction and the dual solution vector, instead, as in the null space method, matrix factorisations are used. [37] proposed a Cholesky decomposition of H

$$H = R'R, \quad R \in \mathbb{R}^{n \times n} \text{ upper triangular}, \quad (3.1.19)$$

and a QR factorisation of $G_{\mathbb{A}^{(k)}}R^{-1}$. These factorisations are updated in each iteration as explained in [36], [44], [21].

3.2 Dual Active Set Methods

In this section we give a short description of *dual active set methods* which have some similarities to our proposed online active set strategy. While primal active set solvers start at a primal feasible point and produce a sequence of primal feasible iterates, dual active set methods maintain *dual* feasibility until an iterate becomes also primal feasible, and hence optimal. This approach is equivalent to solving the dual of the quadratic program $\mathrm{QP}^{\mathrm{dual}}$ (see Definition 2.8) with a primal active set solver (cp. [34]). We present the famous dual active set method by Goldfarb and Idnani [50], [45] which is applicable to strictly convex quadratic programs. For an extension to convex QPs we refer to [16].

One motivation for developing dual QP methods is the trivial but important observation that the pair

$$\left(x^{(0)}, y^{(0)}\right) \stackrel{\text{def}}{=} \left(-H^{-1}g, \mathbb{0}\right) \in \mathcal{F}^{\text{dual}} \tag{3.2.1}$$

can serve as a dual feasible starting point for solving QP^{dual} (with an empty working set $\mathbb{A}^{(0)}$). Thus, besides this computationally cheap matrix-vector calculation, no *Phase I* is necessary!

In the following we divide the dual vector $y^{(k)}$ into an active part $y^{(k)}_{\mathbb{A}^{(k)}}$ and an inactive part $y^{(k)}_{\mathbb{I}^{(k)}}$, where $\mathbb{I}^{(k)} \stackrel{\text{def}}{=} \{1, \ldots, m\} \setminus \mathbb{A}^{(k)}$ is the working set complement (note that $\mathbb{I}^{(k)}$ may contain currently violated constraints). After obtaining $\left(x^{(0)}, y^{(0)}_{\mathbb{A}^{(0)}}\right) = x^{(0)}$, it is checked if this point is also primal feasible, i.e. if $Gx^{(0)} \geq b$ is satisfied. In this case the *unconstrained minimum* $x^{(0)}$ is already the optimal solution. Otherwise a violated (primal) constraint, say $G'_q x^{(0)} < b_q$ with $1 \leq q \leq m$, is selected which shall be satisfied (with equality) by the next iterate $\left(x^{(1)}, y^{(1)}_{\mathbb{A}^{(1)}}\right)$. More generally, at iteration k we want to perform a step in the primal and the dual variables such that a violated constraint $q \notin \mathbb{A}^{(k)}$ becomes active, and hence feasible, at iteration $k+1$:

$$x^{(k+1)} \stackrel{\text{def}}{=} x^{(k)} + \tau \Delta x^{(k)}, \tag{3.2.2a}$$

$$y^{(k+1)}_{\mathbb{A}^{(k+1)} \cup \{q\}} \stackrel{\text{def}}{=} P\left(\begin{pmatrix} y^{(k)}_{\mathbb{A}^{(k)}} \\ y^{(k)}_q \end{pmatrix} + \tau \begin{pmatrix} \Delta y^{(k)}_{\mathbb{A}^{(k)}} \\ 1 \end{pmatrix}\right) \tag{3.2.2b}$$

for an arbitrary $k \in \mathbb{N} \cup \{0\}$ and a fixed $\tau \in \mathbb{R}_{\geq 0}$—the definition of the next working set $\mathbb{A}^{(k+1)}$ and the projection matrix P will be introduced soon. Note that the component of the dual vector correponding to the qth constraint $y^{(k)}_q$ does not need to be zero as constraint q is not feasible. The step directions are determined as follows:

$$\Delta x^{(k)} \stackrel{\text{def}}{=} \left(H^{-1} G'_{\mathbb{A}^{(k)}} W^{(k)} G_{\mathbb{A}^{(k)}} H^{-1} - H^{-1}\right) G'_q, \tag{3.2.3a}$$

$$\Delta y^{(k)}_{\mathbb{A}^{(k)}} \stackrel{\text{def}}{=} -W^{(k)} G_{\mathbb{A}^{(k)}} H^{-1} G'_q. \tag{3.2.3b}$$

Therein $\Delta x^{(k)}$ is chosen such that all (primal) constraints in the working set $\mathbb{A}^{(k)}$ remain active, cf. Eqs. (3.1.18) of the primal range-space method. The *primal-dual step length* τ should be the minimum step length in the primal variables such that the qth constraint becomes feasible (i.e. active); on the other hand τ must be small enough to maintain feasibility of the dual variables:

$$\tau^{\text{prim}} \stackrel{\text{def}}{=} \begin{cases} \infty & \text{if } \Delta x^{(k)} = \mathbb{0} \\ \frac{G'_q x^{(k)} - b_q}{G'_q \Delta x^{(k)}} & \text{else} \end{cases}, \tag{3.2.4a}$$

$$\tau^{\text{dual}} \stackrel{\text{def}}{=} \min_{i \in \mathbb{A}^{(k)}} \left\{ -\frac{y^{(k)}_{\mathbb{A}^{(k)}}}{\Delta y^{(k)}_i} \;\middle|\; \Delta y^{(k)}_i < 0 \right\}, \tag{3.2.4b}$$

$$\tau \stackrel{\text{def}}{=} \min\left\{\tau^{\text{prim}}, \tau^{\text{dual}}\right\}, \tag{3.2.4c}$$

where the minimum over an empty set is defined as ∞, which is greater than any real number.

If the primal step direction $\Delta x^{(k)}$ is *not* zero a *primal-dual step* is taken, trying to make the qth constraint active while maintaining dual feasibility. Two cases can occur:

1. $\tau = \tau^{\text{prim}}$: A *full step* in the primal variables can be taken, q is added to the working set. This means that $\mathbb{A}^{(k+1)} \stackrel{\text{def}}{=} \mathbb{A}^{(k)} \cup \{q\}$ and $P \stackrel{\text{def}}{=} \mathrm{Id}_{\left|\mathbb{A}^{(k+1)}\right|}$ is chosen in Eq. (3.2.2b).

2. $\tau = \tau^{\text{dual}}$: Only a *partial step* can be taken as the *blocking constraint*

$$j \stackrel{\text{def}}{=} \operatorname*{arg\,min}_{i \in \mathbb{A}^{(k)}} \left\{ -\frac{y^{(k)}_{\mathbb{A}^{(k)}}}{\Delta y^{(k)}_i} \;\middle|\; \Delta y^{(k)}_i < 0 \right\} \tag{3.2.5}$$

   must be dropped from the working set in order to keep dual feasibility; constraint q remains infeasible. Thus, in Eq. (3.2.2b), $\mathbb{A}^{(k+1)} \stackrel{\text{def}}{=} \mathbb{A}^{(k)} \setminus \{j\}$ is defined and $P \stackrel{\text{def}}{=} P_j$ deletes component $y^{(k)}_j$ from the right hand side vector (i.e. P_j equals the $\left|\mathbb{A}^{(k)} + 1\right| \times \left|\mathbb{A}^{(k)} + 1\right|$ identity matrix from which one row is deleted).

If the primal step direction $\Delta x^{(k)}$ is zero the qth constraint cannot be satisfied while all other (primal) constraints in $A^{(k)}$ remain active. Thus, no primal step is taken in this case. Instead, provided that $\tau^{\text{dual}} < \infty$, a *partial* dual step is performed which annihilates one component of $\Delta y^{(k+1)}_{\mathbb{A}^{(k)}}$ and allows to drop the corresponding active constraint from the working set ($\mathbb{A}^{(k+1)}$ and $P \stackrel{\text{def}}{=} P_j$ as in the second case above). If such a constraint does not exist, i.e. $\tau = \tau^{\text{dual}} = \infty$, the quadratic program is *infeasible*.
After a partial step new step directions $\Delta x^{(k+1)}$, $\Delta y^{(k+1)}_{\mathbb{A}^{(k+1)}}$ are determined for the updated working set $\mathbb{A}^{(k+1)}$ and constraint q is tried to made active, i.e. feasible, again. As soon as a full step can be taken (if the quadratic program is feasible this must occur if the working set is empty, at the latest), a new violated constraint q is chosen and the whole procedure is repeated. If no violated constraint can be found the primal and dual feasible solution $\left(x^{\text{opt}}, y^{\text{opt}}\right)$ of $\mathrm{QP}^{\text{dual}}$ is found, which also delivers the solution x^{opt} of the corresponding QP. We formalise this dual active set method in Algorithm 3.2 (cf. [45]).

It should be mentioned that a violated constraint q which became active may become inactive and afterwards violated again; the choice of the step directions only ensures that active constraints remain active. But since it can be shown that the (primal) objective function value strictly decreases in every iteration—provided that no *cycling* due to primal degeneracy occurs, see page 31—finite termination of Algorithm 3.2 is guaranteed [45].
The step direction computations in Eqs. (3.2.3) are very similar to that of the range-space method (cf. Section 3.1.2) and similar matrix factorisations and formulae for matrix updates after a working set change exist. Therefore, also recalling that there is no necessity of a phase I, dual methods can be implemented rather efficiently.
A recent implementation particularly suited for large-scale, sparse Hessian and constraint matrices is QPSchur [3]. It is based on a third possibility for solving the KKT system (3.1.2), the so-called *Schur complement* (see e.g. [41]).

Algorithm 3.2 (dual active set method)

`input:` strictly convex quadratic program QP

`output:` optimal solution x^{opt} of QP and working set $\mathbb{A}$ as definied in Theorem 2.5 (or message that QP is infeasible)

(1) Set $k \leftarrow 0$, obtain feasible starting point $\left(x^{(0)}, y^{(0)}\right) \stackrel{\mathrm{def}}{=} \left(-H^{-1}g, \mathbb{0}\right)$ and corresponding working set $\mathbb{A}^{(0)} \stackrel{\mathrm{def}}{=} \emptyset$.

(2) Choose a violated constraint $q \in \left\{i \notin \mathbb{A}^{(k)} \;\middle|\; G_i' x^{(k)} < b_i\right\}$. If such a constraint does not exist the optimal solution is found: set $x^{\mathrm{opt}} \leftarrow x^{(k)}$ and $\mathbb{A} \leftarrow \mathbb{A}^{(k)}$. `stop!`

(3) Calculate primal and dual step directions $\Delta x^{(k)}$ and $\Delta y^{(k)}_{\mathbb{A}^{(k)}}$ from Eqs. (3.2.3).

(4) Compute step length τ (and τ^{prim}, τ^{dual}) via Eqs. (3.2.4).

(5) `if` $\Delta x^{(k)} = \mathbb{0}$:

`if` $\tau^{\mathrm{dual}} = \infty$:

`stop (QP infeasible)!`

`else` $(\tau^{\mathrm{dual}} < \infty)$

Remove blocking constraint $j = \underset{i \in \mathbb{A}^{(k)}}{\arg\min} \left\{ -\frac{y^{(k)}}{\Delta y_i^{(k)}} \;\middle|\; \Delta y_i^{(k)} < 0 \right\}$ from working set, i.e. $\mathbb{A}^{(k+1)} \leftarrow \mathbb{A}^{(k)} \setminus \{j\}$.

Set $x^{(k+1)} \leftarrow x^{(k)}$, $y^{(k+1)}_{\mathbb{A}^{(k+1)} \cup \{q\}} \leftarrow P_j \left(\begin{pmatrix} y^{(k)}_{\mathbb{A}^{(k)}} \\ y^{(k)}_q \end{pmatrix} + \tau \begin{pmatrix} \Delta y_{\mathbb{A}^{(k)}} \\ 1 \end{pmatrix} \right)$

as well as $k \leftarrow k+1$ and continue with step (3).

(6) `if` $\tau = \tau^{\mathrm{prim}}$:

Add the formerly violated constraint q to the working set, i.e. $\mathbb{A}^{(k+1)} \leftarrow \mathbb{A}^{(k)} \cup \{q\}$.

Set $x^{(k+1)} \leftarrow x^{(k)} + \tau \Delta x^{(k)}$, $y^{(k+1)}_{\mathbb{A}^{(k+1)}} \leftarrow \begin{pmatrix} y^{(k)}_{\mathbb{A}^{(k)}} \\ y^{(k)}_q \end{pmatrix} + \tau \begin{pmatrix} \Delta y_{\mathbb{A}^{(k)}} \\ 1 \end{pmatrix}$

as well as $k \leftarrow k+1$ and continue with step (2).

`else` $(\tau = \tau^{\mathrm{dual}})$

Remove blocking constraint $j = \underset{i \in \mathbb{A}^{(k)}}{\arg\min} \left\{ -\frac{y^{(k)}}{\Delta y_i^{(k)}} \;\middle|\; \Delta y_i^{(k)} < 0 \right\}$ from working set, i.e. $\mathbb{A}^{(k+1)} \leftarrow \mathbb{A}^{(k)} \setminus \{j\}$.

Set $x^{(k+1)} \leftarrow x^{(k)}$, $y^{(k+1)}_{\mathbb{A}^{(k+1)} \cup \{q\}} \leftarrow P_j \left(\begin{pmatrix} y^{(k)}_{\mathbb{A}^{(k)}} \\ y^{(k)}_q \end{pmatrix} + \tau \begin{pmatrix} \Delta y_{\mathbb{A}^{(k)}} \\ 1 \end{pmatrix} \right)$

as well as $k \leftarrow k+1$ and continue with step (3).

3.3 Interior Point Methods

So-called *primal-dual interior point methods* have emerged as a strong competitor to active set methods. Initially developed for linear programming, they were extended to convex quadratic programming and to general nonlinear programming afterwards. Since a detailed description is beyond the scope of this thesis we refer to [91] for an overview. The main idea can be summarised as follows: first observe that the KKT optimality conditions (2.3.13) imply that a primal-dual pair $\left(x^{(k)}, y^{(k)}\right)$, $k \geq 0$, is optimal if and only if

$$\begin{aligned} Hx^{(k)} - G'y^{(k)} &= -g\,, && \text{(3.3.1a)} \\ Gx^{(k)} &\geq b\,, && \text{(3.3.1b)} \\ y^{(k)} &\geq \mathbb{0}\,, && \text{(3.3.1c)} \\ y_i^{(k)} \left(Gx^{(k)} - b\right)_i &= 0 \quad \forall\, i \in \{1, \ldots, m\}\,. && \text{(3.3.1d)} \end{aligned}$$

Interior-point methods relax the so-called *complementary slackness condition* (3.3.1d) to

$$y_i^{(k)} \left(Gx^{(k)} - b\right)_i = \mu^{(k)} \quad \forall\, i \in \{1, \ldots, m\} \qquad \text{(3.3.1d')}$$

for some $\mu^{(k)} \in \mathbb{R}_{>0}$ and produce a sequence of iterates $\left(x^{(k)}, y^{(k)}\right)$ which strictly satisfy Eqs. (3.3.1b) and (3.3.1c). The optimal primal-dual solution is finally found by ensuring $\mu^{(k)} \to 0$ for $k \to \infty$.

One famous implementation for convex quadratic programs is `LOQO` [81]; another one for general NLPs is IPOPT [82]. For interior point methods, a *polynomial runtime guarantee* can be given and they posses relatively constant computational demands. But they suffer the drawback that no efficient *warm start* techniques exist so far. "For large QPs with many active inequality constraints the interior point approach is expected to require far fewer iterations than an active set method to arrive at the solution. However, each of the interior points iterations is many times more expensive than the iterations performed in an active set method." [4].

Interior-point methods have also been proposed for use in model predictive control [73]. Comparisions with active set solvers indicate that it depends on the problem's characteristics which method should be preferred [5], [4].

Chapter 4

An Online Active Set Strategy for Model Predictive Control

4.1 Main Idea

Inspired by the explicit solution approach, but aiming to avoid its prohibitive offline computational cost, we propose an *online active set strategy* for use in model predictive control. It builds on the expectation that the active set does not change much from one quadratic program to the next, but is different from conventional warm starting techniques. For transition from the old QP to a new one, we propose to move on a straight line in the parameter space, i.e., in the set $\mathcal{P}$. As this set is convex, cf. Theorem 2.6, we can be sure that all QPs on this line remain feasible and can be solved. As long as we stay in one *critical region*, the QP solution depends affinely on w_0. If we have to cross the boundaries of critical regions during our way on the line, which is illustrated in Fig. 4.1, Theorem 2.8 ensures that the solution can be continuously continued.

Let us assume that we have solved a parametric quadratic program of the form (2.3.15) for a certain initial state w_0 and (after one sampling time) want to solve it again for a new initial state vector w_0^{new} with unknown solution $\left(x_{\text{new}}^{\text{opt}}, y_{\text{new}}^{\text{opt}}\right)$. By setting

$$\Delta w_0 \overset{\text{def}}{=} w_0^{\text{new}} - w_0\,, \tag{4.1.1a}$$

$$\Delta g \overset{\text{def}}{=} g(w_0^{\text{new}}) - g(w_0) = F'\Delta w_0\,, \tag{4.1.1b}$$

$$\Delta b \overset{\text{def}}{=} b(w_0^{\text{new}}) - b(w_0) = E\Delta w_0\,, \tag{4.1.1c}$$

we can re-parameterise gradient and right hand side vector as follows:

$$\tilde{w}_0\colon [0,1] \to \mathbb{R}^{n_x}, \qquad \tilde{w}_0(\tau) \overset{\text{def}}{=} w_0 + \tau\Delta w_0\,, \tag{4.1.2a}$$

$$\tilde{g}\colon [0,1] \to \mathbb{R}^{n}, \qquad \tilde{g}(\tau) \overset{\text{def}}{=} g(w_0) + \tau\Delta g\,, \tag{4.1.2b}$$

$$\tilde{b}\colon [0,1] \to \mathbb{R}^{m}, \qquad \tilde{b}(\tau) \overset{\text{def}}{=} b(w_0) + \tau\Delta b\,. \tag{4.1.2c}$$

This leads to a re-parameterised form of $\mathrm{QP}(w_0)$:

$$\mathrm{QP}(\tau): \qquad \min_x \quad \tfrac{1}{2}x'Hx + x'\tilde{g}(\tau) \tag{4.1.3a}$$

$$\text{s.t.} \qquad Gx \geq \tilde{b}(\tau)\,. \tag{4.1.3b}$$

According to our assumption, we know the solution x^{opt} and y^{opt} (and a corresponding working set $\mathbb{A}$) of $\mathrm{QP}(w_0)$ and want to solve $\mathrm{QP}(w_0^{\mathrm{new}})$. The basic idea of our online active set strategy, which has previously been proposed by [11] in a different context, is to move from w_0 towards w_0^{new}, and thus from $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ towards $\left(x_{\mathrm{new}}^{\mathrm{opt}}, y_{\mathrm{new}}^{\mathrm{opt}}\right)$, while keeping primal and dual feasibility (i.e. optimality) for all intermediate points. This means that we are looking for homotopies

$$\tilde{x}^{\mathrm{opt}}\colon [0,1] \to \mathbb{R}^n, \qquad \tilde{x}^{\mathrm{opt}}(0) = x^{\mathrm{opt}},\ \tilde{x}^{\mathrm{opt}}(1) = x_{\mathrm{new}}^{\mathrm{opt}}\,, \tag{4.1.4a}$$

$$\tilde{y}^{\mathrm{opt}}\colon [0,1] \to \mathbb{R}^m, \qquad \tilde{y}^{\mathrm{opt}}(0) = y^{\mathrm{opt}},\ \tilde{y}^{\mathrm{opt}}(1) = y_{\mathrm{new}}^{\mathrm{opt}}\,, \tag{4.1.4b}$$

$$\tilde{\mathbb{A}}\colon [0,1] \to 2^{\{1,\ldots,m\}}, \qquad \tilde{\mathbb{A}}(0) = \mathbb{A},\ \tilde{\mathbb{A}}(\tau) \subseteq \{1,\ldots,m\}\,, \tag{4.1.4c}$$

$$\tilde{\mathbb{I}}\colon [0,1] \to 2^{\{1,\ldots,m\}}, \qquad \tilde{\mathbb{I}}(\tau) \stackrel{\mathrm{def}}{=} \{1,\ldots,m\} \setminus \tilde{\mathbb{A}}(\tau)\,, \tag{4.1.4d}$$

which satisfy the conditions of Theorem 2.5 at every point $\tau \in [0,1]$:

$$\begin{pmatrix} H & G'_{\tilde{\mathbb{A}}(\tau)} \\ G_{\tilde{\mathbb{A}}(\tau)} & \mathbb{0} \end{pmatrix} \begin{pmatrix} \tilde{x}^{\mathrm{opt}}(\tau) \\ -\tilde{y}^{\mathrm{opt}}_{\tilde{\mathbb{A}}(\tau)}(\tau) \end{pmatrix} = \begin{pmatrix} -\tilde{g}(\tau) \\ \tilde{b}_{\tilde{\mathbb{A}}(\tau)}(\tau) \end{pmatrix}, \tag{4.1.5a}$$

$$G_{\tilde{\mathbb{I}}(\tau)}\tilde{x}^{\mathrm{opt}}(\tau) \geq b_{\tilde{\mathbb{I}}(\tau)}(\tau)\,, \tag{4.1.5b}$$

$$\tilde{y}^{\mathrm{opt}}_{\tilde{\mathbb{I}}(\tau)}(\tau) = \mathbb{0}\,, \tag{4.1.5c}$$

$$\tilde{y}^{\mathrm{opt}}_{\tilde{\mathbb{A}}(\tau)}(\tau) \geq \mathbb{0}\,. \tag{4.1.5d}$$

This implies that $\tilde{x}^{\mathrm{opt}}(\tau)$ and $\tilde{y}^{\mathrm{opt}}(\tau)$ are *piecewise linear* functions and that $\tilde{x}^{\mathrm{opt}}(\tau)$ is also *continuous*, as shown in Theorem 2.8. Thus, locally we must have a relation of the form

$$\tilde{x}^{\mathrm{opt}}(\tau) \stackrel{\mathrm{def}}{=} x^{\mathrm{opt}} + \tau\Delta x^{\mathrm{opt}}\,, \tag{4.1.6a}$$

$$\tilde{y}^{\mathrm{opt}}_{\tilde{\mathbb{A}}(\tau)} \stackrel{\mathrm{def}}{=} y^{\mathrm{opt}}_{\mathbb{A}} + \tau\Delta y^{\mathrm{opt}}_{\mathbb{A}}\,, \tag{4.1.6b}$$

which holds for sufficiently small $\tau \in [0, \tau_{\max}]$, $\tau_{\max} \in \mathbb{R}_{\geq 0}$.

Because we start from an optimal solution we know that conditions (4.1.5) are satisfied at $\tau = 0$. Therefore equality (4.1.5a) is satisfied for all $\tau \in [0, \tau_{\max}]$ if and only if

$$\begin{pmatrix} H & G'_{\mathbb{A}} \\ G_{\mathbb{A}} & \mathbb{0} \end{pmatrix} \begin{pmatrix} \Delta x^{\mathrm{opt}} \\ -\Delta y^{\mathrm{opt}}_{\mathbb{A}} \end{pmatrix} = \begin{pmatrix} -\Delta g \\ \Delta b_{\mathbb{A}} \end{pmatrix} \tag{4.1.7}$$

holds. Because it will be ensured that all rows of $G_{\mathbb{A}}$ are linearly independent, Eq. (4.1.7) has a unique solution, as shown in Lemma 2.2.
The active set stays constant as long as no previously inactive constraint becomes active (cf. (4.1.5b)), i.e.

$$G'_i\left(x^{\mathrm{opt}} + \tau\Delta x^{\mathrm{opt}}\right) = b_i(w_0) + \tau\Delta b_i \tag{4.1.8}$$

for some $i \in \tilde{\mathbb{I}}(0)$, and no previously active constraint becomes inactive (cf. (4.1.5d)), i.e.

$$y_i^{\mathrm{opt}} + \tau\Delta y_i = 0 \tag{4.1.9}$$

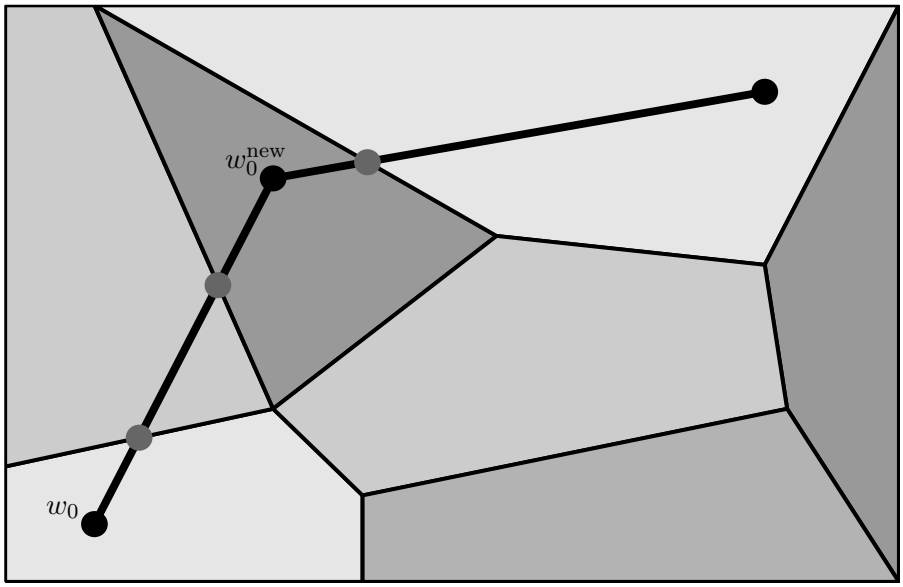


Figure 4.1: Homotopy paths from one QP to the next across multiple critical regions.

for some $i \in \tilde{\mathbb{A}}(0)$. Therefore, we determine the maximum possible *homotopy step length* $\tau_{\max}$ as follows[1]:

$$\tau_{\max}^{\text{prim}} \overset{\text{def}}{=} \min_{i \in \tilde{\mathbb{I}}(0)} \left\{ \frac{b_i(w_0) - G_i' x^{\text{opt}}}{G_i' \Delta x^{\text{opt}} - \Delta b_i} \;\middle|\; G_i' \Delta x^{\text{opt}} < \Delta b_i \right\} \in \mathbb{R}_{\geq 0}\,, \tag{4.1.10a}$$

$$\tau_{\max}^{\text{dual}} \overset{\text{def}}{=} \min_{i \in \tilde{\mathbb{A}}(0)} \left\{ -\frac{y_i^{\text{opt}}}{\Delta y_i} \;\middle|\; \Delta y_i < 0 \right\} \in \mathbb{R}_{\geq 0}\,, \tag{4.1.10b}$$

$$\tau_{\max} \overset{\text{def}}{=} \min \left\{ 1, \tau_{\max}^{\text{prim}}, \tau_{\max}^{\text{dual}} \right\} \in [0,1]\,. \tag{4.1.10c}$$

This choice of $\tau_{\max}$ ensures that conditions (4.1.5b) and (4.1.5d) remain fulfilled. Moreover, if we define $\Delta y_{\mathbb{I}}^{\text{opt}} \overset{\text{def}}{=} \mathbb{0}$ then also equality (4.1.5c) holds for all $\tau \in [0, \tau_{\max}]$.

Our online active set strategy is summarised in Algorithm 4.1 (where the homotopy interval $[0,1]$ is implicitly rescaled after each working set change, for notational simplicity and implementation elegance).

[1] Again, the minimum over an empty set is defined as ∞.

Algorithm 4.1 (online active set strategy)

`input:` data and solution $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ of $\mathrm{QP}(w_0)$,
corresponding working set $\mathbb{A}$,
new parameter $w_0^{\mathrm{new}} \in \mathcal{P}$

`output:` solution pair $\left(x_{\mathrm{new}}^{\mathrm{opt}}, y_{\mathrm{new}}^{\mathrm{opt}}\right)$ of $\mathrm{QP}(w_0^{\mathrm{new}})$,
corresponding working set $\mathbb{A}^{\mathrm{new}}$

(1) Calculate Δw_0, Δg and Δb via Eqs. (4.1.1).

(2) Calculate *primal* and *dual step directions* Δx^{opt} and Δy^{opt} via Eq. (4.1.7).

(3) Determine maximum *homotopy step length* $\tau_{\max}$ from Eqs. (4.1.10).

(4) Obtain *optimal solution* of $\mathrm{QP}(\tilde{w}_0)$:

(a) $\tilde{w}_0 \leftarrow w_0 + \tau_{\max} \Delta w_0$,

(b) $\tilde{x}^{\mathrm{opt}} \leftarrow x^{\mathrm{opt}} + \tau_{\max} \Delta x^{\mathrm{opt}}$,

(c) $\tilde{y}^{\mathrm{opt}} \leftarrow y^{\mathrm{opt}} + \tau_{\max} \Delta y^{\mathrm{opt}}$.

(5) `if` $\tau_{\max} = 1$:
Optimal solution of $\mathrm{QP}(w_0^{\mathrm{new}})$ found.
Set $x_{\mathrm{new}}^{\mathrm{opt}} \leftarrow \tilde{x}^{\mathrm{opt}}$, $y_{\mathrm{new}}^{\mathrm{opt}} \leftarrow \tilde{y}^{\mathrm{opt}}$ and $\mathbb{A}^{\mathrm{new}} \leftarrow \mathbb{A}$. `stop!`

(6) `if` $\tau_{\max} = \tau_{\max}^{\mathrm{dual}}$:
Remove a *dual blocking constraint* $j \in \mathbb{A}$ $\left(\tau_{\max}^{\mathrm{dual}} = -\frac{y_j^{\mathrm{opt}}}{\Delta y_j}\right)$ from working set,
i.e. $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$.
`elseif` $\tau_{\max} = \tau_{\max}^{\mathrm{prim}}$:
Add a *primal blocking constraint* j $\left(\tau_{\max}^{\mathrm{prim}} = \frac{b_j(w_0) - G_j' x^{\mathrm{opt}}}{G_j' \Delta x^{\mathrm{opt}} - \Delta b_j}\right)$ to working set,
i.e. $\mathbb{A} \leftarrow \mathbb{A} \cup \{j\}$, while ensuring linear independence (see Section 4.5.1).

(7) Set $w_0 \leftarrow \tilde{w}_0$, $x^{\mathrm{opt}} \leftarrow \tilde{x}^{\mathrm{opt}}$, $y^{\mathrm{opt}} \leftarrow \tilde{y}^{\mathrm{opt}}$ and continue with step (1).

4.2 Real-Time Variant

One advantage of our online active set strategy is that it produces a *sequence of optimal solutions* for QPs on the homotopy path. Thus, it is possible to interrupt this sequence after every partial step and start a new homotopy from the current iterate towards the next QP. In particular, no *Phase I* as in standard active set methods is neccessary because every iterate is optimal and therefore feasible. Of course, if we interrupt the homotopy before the solution is reached we may stop at an infeasible point with respect to the QP we want to solve.
In a real-time scenario one can try to find the optimal solution of the current QP within a given sampling time. But if too many working set changes are nessesary to get from the solution of the old QP to that of the current QP one can just stop the solution of the current QP and start a new homotopy towards the solution of the new one. If the solution of the new QP requires fewer working set changes than computable within the given sampling time the online active set strategy may make up for some unperformed changes from the last QP. This situation is illustrated in Figure 4.2 wherein only two working set changes are allowed per QP.
The computational effort per working set change is known rather exactly, see Section 4.6.1. So, if one obtains an estimate for the number of optimal active set changes from one QP to the next, e.g. from closed-loop simulations, it is easy to estimate the *possible sampling time length*.

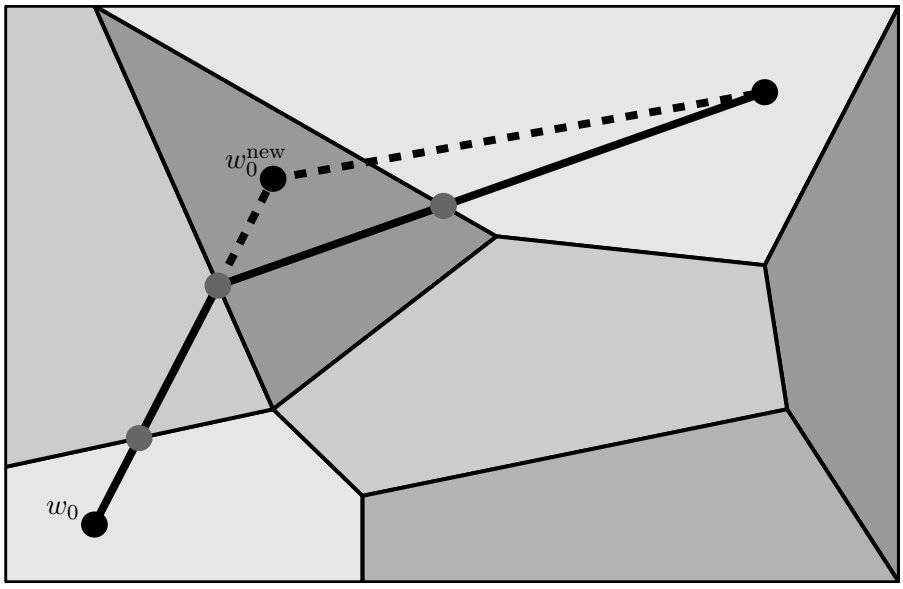


Figure 4.2: Homotopy paths (solid) from one QP to the next with limited number of working set changes.

Note that our online active set strategy has some features similar to the dual active set method, see Section 3.2, and its adaptation to fast MPC [84]: both allow QP warm starting

without a phase I. When iterations are terminated prematurely, however, our method solves a QP that is exactly known to lie on the straight line between $\mathrm{QP}(w_0)$ and $\mathrm{QP}(w_0^{\mathrm{new}})$, while the dual active set method delivers in each iteration the solution to an unknown primal QP. Using the real-time variant of our online active set strategy, it is reasonable to assume a greater probability (compared with the dual approach) of reaching at least the *confidence region* of the measured initial state w_0.

4.3 Implementation Details

4.3.1 Bounds and Constraints

Instead of the general formulation (2.3.15), our online active set strategy was implemented for QPs of the following form:

$$\min_x \quad \tfrac{1}{2}x'Hx + x'g(w_0) \tag{4.3.1a}$$

$$\text{s.t.} \quad \underline{b_{\mathrm{B}}}(w_0) \leq \ x \ \leq \overline{b_{\mathrm{B}}}(w_0)\,, \tag{4.3.1b}$$

$$\underline{b_{\mathrm{C}}}(w_0) \leq Gx \leq \overline{b_{\mathrm{C}}}(w_0)\,, \tag{4.3.1c}$$

where $G \in \mathbb{R}^{m\times n}$, $\underline{b_{\mathrm{B}}}(w_0)$, $\overline{b_{\mathrm{B}}}(w_0) \in \mathbb{R}^n$ and $\underline{b_{\mathrm{C}}}(w_0)$, $\overline{b_{\mathrm{C}}}(w_0) \in \mathbb{R}^m$ for all $w_0 \in \mathcal{P}$. This distinction between *constraints* and *bounds* seems adequate because bounds arise naturally in the context of model predictive control and special treatment of them can lead to substantial computional savings as described in [39]. See also Section 4.6.1 where complexity issues are addressed.

Similar to Definition 2.9 we give the following

Definition 4.1 (free and fixed variables): *Let a feasible quadratic program of the form* (4.3.1) *be given. A variable* x_i, $1 \leq i \leq n$, *is called* fixed *(and the corresponding bound* active*)* $\check{x} \in \mathcal{F}$ *iff*

$$\check{x}_i = \underline{b_{\mathrm{B}}}(w_0)_i \ \vee \ \check{x}_i = \overline{b_{\mathrm{B}}}(w_0)_i$$

holds and free *otherwise. The (disjoint) index sets*

$$\begin{aligned}\mathbb{F}(\check{x}) &\stackrel{\mathrm{def}}{=} \big\{i \in \{1,\ldots,n\} \mid \check{x}_i \text{ free}\big\}\,,\\ \mathbb{X}(\check{x}) &\stackrel{\mathrm{def}}{=} \big\{i \in \{1,\ldots,n\} \mid \check{x}_i \text{ fixed}\big\}\end{aligned}$$

are called set of free variables *and* set of fixed variables, *respectively.* ○

Definition 4.2 (working set of variables): *Let a feasible quadratic program of the form* (4.3.1) *be given. Then arbitrary index sets*

$$\begin{aligned}\mathbb{F} &\subseteq \{1,\ldots,n\}\,,\\ \mathbb{X} &\stackrel{\mathrm{def}}{=} \{1,\ldots,n\} \setminus \mathbb{F}\end{aligned}$$

are called working set!of free variables *and* working set!of fixed variables, *respectively. Their cardinalities are denoted by*

$$\begin{aligned}n_{\mathbb{F}} &\stackrel{\mathrm{def}}{=} |\mathbb{F}|\,,\\ n_{\mathbb{X}} &\stackrel{\mathrm{def}}{=} |\mathbb{X}|\,.\end{aligned}$$ ○

For every feasible point x of the QP (4.3.1) there exist corresponding working sets of free and fixed variables $\mathbb{F} \subseteq \mathbb{F}(x)$ and $\mathbb{X}$ as well as a working set $\mathbb{A} \subseteq \mathbb{A}(x)$. That means that we can rearrange the components of x such that

$$\begin{pmatrix} \mathbb{0} & \mathrm{Id}_{n_\mathbb{X}} \\ C_\mathbb{F} & C_\mathbb{X} \end{pmatrix} \begin{pmatrix} x_\mathbb{F} \\ x_\mathbb{X} \end{pmatrix} = \begin{pmatrix} b_\mathbb{X} \\ b_\mathbb{A} \end{pmatrix} \tag{4.3.2}$$

is valid, where $C \stackrel{\text{def}}{=} G_\mathbb{A}$ and $b_\mathbb{X} \in \mathbb{R}^{n_\mathbb{X}}$ and $b_\mathbb{A} \in \mathbb{R}^{n_\mathbb{A}}$ contain suitable subsets of the components of $\underline{b_\mathrm{B}}(w_0)_\mathbb{X}$, $\overline{b_\mathrm{B}}(w_0)_\mathbb{X}$ and $\underline{b_\mathrm{C}}(w_0)_\mathbb{A}$, $\overline{b_\mathrm{C}}(w_0)_\mathbb{A}$, respectively. We call C *active constraints matrix* and the left hand side matrix of Eq. (4.3.2) *augmented active constraints matrix*. This representation of fixed variables and active constraints will be useful in Section 4.3.3 when matrix updates are to be described.

4.3.2 Null Space Approach

Our implementation is based on the null space approach (cf. Section 3.1.1) for solving the KKT system (4.1.7). For this choice several reasons were decisive: first, as explained in Chapter 3, the null space method is particularly numerically stable and, in contrast to the range space method and the dual approach, no positive definite Hessian matrix is required; instead, a positive definite *projected Hessian* matrix is sufficient which facilitates extensions for dealing with positive semi-definite Hessian matrices (including linear objective functions). Furthermore, computational savings due to the distinction of bounds and constraints, which seems well justified within MPC problems, are "most readily achieved in null space methods." [34] Finally, when using the null space approach the more bounds and constraints are active the less computational effort is required per working set change. So, the proposed online active set strategy takes the most computational time per working set change if the controlled system is near the steady-state and almost no active set changes occur. If, e.g. after a strong pertubation, the controlled system is far from its steady-state and typically many optimal active set changes are neccessary our online active set strategy can perform more working set changes per sampling time than near the steady-state. Section 4.6.1 illustrates that a significant amount of computational effort is saved if many bounds become active.

The distinction of bounds and constraints makes necessary adaptations of the matrix decompositions and of the way the KKT system (4.1.7) is solved in order to determine the primal-dual step direction. Therefore, both matrices are subdivided into parts corresponding to free and fixed variables, respectively:

$$\begin{pmatrix} C_\mathbb{F} & C_\mathbb{X} \end{pmatrix} \begin{pmatrix} x_\mathbb{F} \\ x_\mathbb{X} \end{pmatrix} \stackrel{\text{def}}{=} Cx\,, \tag{4.3.3a}$$

$$\begin{pmatrix} x_\mathbb{F} \\ x_\mathbb{X} \end{pmatrix}' \begin{pmatrix} H_\mathbb{F} & H_\mathrm{M} \\ H'_\mathrm{M} & H_\mathbb{X} \end{pmatrix} \begin{pmatrix} x_\mathbb{F} \\ x_\mathbb{X} \end{pmatrix} \stackrel{\text{def}}{=} x'Hx\,, \tag{4.3.3b}$$

where $H_\mathbb{F} \in \mathbb{R}^{n_\mathbb{F} \times n_\mathbb{F}}$, $H_\mathbb{X} \in \mathbb{R}^{n_\mathbb{X} \times n_\mathbb{X}}$, $H_\mathrm{M} \in \mathbb{R}^{n_\mathbb{F} \times n_\mathbb{X}}$ and $C_\mathbb{F} \in \mathbb{R}^{n_\mathbb{A} \times n_\mathbb{F}}$, $C_\mathbb{X} \in \mathbb{R}^{n_\mathbb{A} \times n_\mathbb{X}}$. Accordingly, not the whole active constraint matrix C is decomposed but only that part which corresponds to the free variables $\mathbb{F}$. Instead of the common QR decomposition a

variant called *TQ factorisation*, as proposed in [39], is maintained during the iterations:

$$C'_{\mathbb{F}} = V \begin{pmatrix} U \\ \mathbb{0} \end{pmatrix} \tag{4.3.4a}$$

$$\iff \quad C_{\mathbb{F}} = \begin{pmatrix} U \\ \mathbb{0} \end{pmatrix}' V' = \begin{pmatrix} U \\ \mathbb{0} \end{pmatrix}' \mathrm{Id}^{\mathrm{r}}_{n_{\mathbb{F}}} \mathrm{Id}^{\mathrm{r}}_{n_{\mathbb{F}}} V', \quad \mathrm{Id}^{\mathrm{r}}_{n_{\mathbb{F}}} \stackrel{\mathrm{def}}{=} \begin{pmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{pmatrix} \tag{4.3.4b}$$

$$\iff \quad C_{\mathbb{F}} = \begin{pmatrix} \mathbb{0} & T \end{pmatrix} Q'_{\mathbb{F}} \tag{4.3.4c}$$

where $V \in \mathbb{R}^{n_{\mathbb{F}} \times n_{\mathbb{F}}}$ is an orthonormal and $U \in \mathbb{R}^{n_{\mathbb{A}} \times n_{\mathbb{A}}}$ an upper triangular matrix. Thus, $T \stackrel{\mathrm{def}}{=} U' \mathrm{Id}^{\mathrm{r}}_{n_{\mathbb{A}}}$ is a *reverse lower triangular matrix* and $Q_{\mathbb{F}} \stackrel{\mathrm{def}}{=} \mathrm{Id}^{\mathrm{r}}_{n_{\mathbb{F}}} V$ is orthonormal because both factors $\mathrm{Id}^{\mathrm{r}}_{n_{\mathbb{F}}}$ and V are. Matrix $Q_{\mathbb{F}}$ is subdivided into

$$\begin{pmatrix} Z_{\mathbb{F}} & Y_{\mathbb{F}} \end{pmatrix} \stackrel{\mathrm{def}}{=} Q_{\mathbb{F}} \tag{4.3.5}$$

where $Z_{\mathbb{F}} \in \mathbb{R}^{n_{\mathbb{F}} \times (n_{\mathbb{F}} - n_{\mathbb{A}})}$ contains a basis of the null space restricted to free variables and $Y_{\mathbb{F}} \in \mathbb{R}^{n_{\mathbb{F}} \times n_{\mathbb{A}}}$ is formed by range space basis vectors of $C_{\mathbb{F}}$. This leads to the following

Definition 4.3 (restricted null space): *Let $Q_{\mathbb{F}}$ be an orthonormal matrix as defined in Eqs.* (4.3.4) *and let $Z_{\mathbb{F}}$ denote the $(n_{\mathbb{F}} - n_{\mathbb{A}})$ leftmost columns of $Q_{\mathbb{F}}$. Then*

$$\operatorname{im} Z_{\mathbb{F}} \subseteq \mathbb{R}^{n_{\mathbb{F}}}$$

is called restricted null space *of the active constraints. Its dimension is denoted by*

$$n_{\mathrm{Z}} \stackrel{\mathrm{def}}{=} n_{\mathbb{F}} - n_{\mathbb{A}} \,. \qquad \bigcirc$$

A Cholesky decomposition is only calculated for the Hessian projected to the restricted null space of $C_{\mathbb{F}}$:

$$R'R \stackrel{\mathrm{def}}{=} Z'_{\mathbb{F}} H_{\mathbb{F}} Z_{\mathbb{F}} \,, \tag{4.3.6}$$

where $R \in \mathbb{R}^{n_{\mathrm{Z}} \times n_{\mathrm{Z}}}$ is an upper triangular matrix.

After the adaptation of the matrix decompositions we now have a closer look at the way the *primal-dual step direction* is determined. To this end the KKT system (4.1.7) is subdivided into free and fixed variables:

$$\begin{pmatrix} H_{\mathbb{F}} & H_{\mathrm{M}} & \mathbb{0} & C'_{\mathbb{F}} \\ H'_{\mathrm{M}} & H_{\mathbb{X}} & \mathrm{Id}_{n_{\mathbb{X}}} & C'_{\mathbb{X}} \\ \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} & \mathbb{0} & \mathbb{0} \\ C_{\mathbb{F}} & C_{\mathbb{X}} & \mathbb{0} & \mathbb{0} \end{pmatrix} \begin{pmatrix} \Delta x_{\mathbb{F}} \\ \Delta x_{\mathbb{X}} \\ \Delta y_{\mathbb{X}} \\ \Delta y_{\mathbb{A}} \end{pmatrix} = \begin{pmatrix} -\Delta g_{\mathbb{F}} \\ -\Delta g_{\mathbb{X}} \\ \Delta b_{\mathbb{X}} \\ \Delta b_{\mathbb{A}} \end{pmatrix}, \tag{4.3.7}$$

where $\Delta x_{\mathbb{F}} \in \mathbb{R}^{n_{\mathbb{F}}}$ and $\Delta x_{\mathbb{X}} \in \mathbb{R}^{n_{\mathbb{X}}}$ denote the *primal step direction* of free and fixed variables, respectively; $\Delta y_{\mathbb{X}} \in \mathbb{R}^{n_{\mathbb{X}}}$ and $\Delta y_{\mathbb{A}} \in \mathbb{R}^{n_{\mathbb{A}}}$ denote the *dual step direction* of active bounds and constraints, respectively; $\Delta g_{\mathbb{F}} \in \mathbb{R}^{n_{\mathbb{F}}}$ and $\Delta g_{\mathbb{X}} \in \mathbb{R}^{n_{\mathbb{X}}}$ denote the *gradient step direction* for free and fixed variables, respectively; $\Delta b_{\mathbb{X}} \in \mathbb{R}^{n_{\mathbb{X}}}$ denotes the step direction of the active bounds vectors[2] and $\Delta b_{\mathbb{A}} \in \mathbb{R}^{n_{\mathbb{A}}}$ denotes the step direction of the active constraints vectors[2].

[2] A suitable subset of the lower and upper (constraints') bounds vectors, to be more precise.

Then we use the orthonormal matrix $Q_{\mathbb{F}}$ to perform a coordinate transformation: with the definition

$$S \stackrel{\text{def}}{=} \begin{pmatrix} Z'_{\mathbb{F}} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ Y'_{\mathbb{F}} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{A}}} \end{pmatrix} \in \mathbb{R}^{n+n_{\mathbb{X}}+n_{\mathbb{A}}} \tag{4.3.8}$$

we obtain

$$S \begin{pmatrix} H_{\mathbb{F}} & H_{\mathrm{M}} & \mathbb{0} & C'_{\mathbb{F}} \\ H'_{\mathrm{M}} & H_{\mathbb{X}} & \mathrm{Id}_{n_{\mathbb{X}}} & C'_{\mathbb{X}} \\ \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} & \mathbb{0} & \mathbb{0} \\ C_{\mathbb{F}} & C_{\mathbb{X}} & \mathbb{0} & \mathbb{0} \end{pmatrix} S' = \begin{pmatrix} Z'_{\mathbb{F}}H_{\mathbb{F}}Z_{\mathbb{F}} & Z'_{\mathbb{F}}H_{\mathbb{F}}Y_{\mathbb{F}} & Z'_{\mathbb{F}}H_{\mathrm{M}} & \mathbb{0} & Z'_{\mathbb{F}}C'_{\mathbb{F}} \\ Y'_{\mathbb{F}}H_{\mathbb{F}}Z_{\mathbb{F}} & Y'_{\mathbb{F}}H_{\mathbb{F}}Y_{\mathbb{F}} & Y'_{\mathbb{F}}H_{\mathrm{M}} & \mathbb{0} & Y'_{\mathbb{F}}C'_{\mathbb{F}} \\ H'_{\mathrm{M}}Z_{\mathbb{F}} & H'_{\mathrm{M}}Y_{\mathbb{F}} & H_{\mathbb{X}} & \mathrm{Id}_{n_{\mathbb{X}}} & C'_{\mathbb{X}} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} & \mathbb{0} & \mathbb{0} \\ C_{\mathbb{F}}Z_{\mathbb{F}} & C_{\mathbb{F}}Y_{\mathbb{F}} & C_{\mathbb{X}} & \mathbb{0} & \mathbb{0} \end{pmatrix}, \tag{4.3.9a}$$

$$\begin{pmatrix} \Delta x_{\mathbb{F}}^{Z} \\ \Delta x_{\mathbb{F}}^{Y} \\ \Delta x_{\mathbb{X}} \\ \Delta y_{\mathbb{X}} \\ \Delta y_{\mathbb{A}} \end{pmatrix} \stackrel{\text{def}}{=} S \begin{pmatrix} \Delta x_{\mathbb{F}} \\ \Delta x_{\mathbb{X}} \\ \Delta y_{\mathbb{X}} \\ \Delta y_{\mathbb{A}} \end{pmatrix}, \tag{4.3.9b}$$

$$\begin{pmatrix} -\Delta g_{\mathbb{F}}^{Z} \\ -\Delta g_{\mathbb{F}}^{Y} \\ -\Delta g_{\mathbb{X}} \\ \Delta b_{\mathbb{X}} \\ \Delta b_{\mathbb{A}} \end{pmatrix} \stackrel{\text{def}}{=} S \begin{pmatrix} -\Delta g_{\mathbb{F}} \\ -\Delta g_{\mathbb{X}} \\ \Delta b_{\mathbb{X}} \\ \Delta b_{\mathbb{A}} \end{pmatrix}. \tag{4.3.9c}$$

This leads to the following linear system for determination of the primal-dual step directions

$$\begin{pmatrix} R'R & Z'_{\mathbb{F}}H_{\mathbb{F}}Y_{\mathbb{F}} & Z'_{\mathbb{F}}H_{\mathrm{M}} & \mathbb{0} & \mathbb{0} \\ Y'_{\mathbb{F}}H_{\mathbb{F}}Z_{\mathbb{F}} & Y'_{\mathbb{F}}H_{\mathbb{F}}Y_{\mathbb{F}} & Y'_{\mathbb{F}}H_{\mathrm{M}} & \mathbb{0} & T' \\ H'_{\mathrm{M}}Z_{\mathbb{F}} & H'_{\mathrm{M}}Y_{\mathbb{F}} & H_{\mathbb{X}} & \mathrm{Id}_{n_{\mathbb{X}}} & C'_{\mathbb{X}} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & T & C_{\mathbb{X}} & \mathbb{0} & \mathbb{0} \end{pmatrix} \begin{pmatrix} \Delta x_{\mathbb{F}}^{Z} \\ \Delta x_{\mathbb{F}}^{Y} \\ \Delta x_{\mathbb{X}} \\ -\Delta y_{\mathbb{X}} \\ -\Delta y_{\mathbb{A}} \end{pmatrix} = \begin{pmatrix} -\Delta g_{\mathbb{F}}^{Z} \\ -\Delta g_{\mathbb{F}}^{Y} \\ -\Delta g_{\mathbb{X}} \\ \Delta b_{\mathbb{X}} \\ \Delta b_{\mathbb{A}} \end{pmatrix} \tag{4.3.10}$$

with the following solutions:

$$\begin{aligned} \Delta x_{\mathbb{X}} &= \Delta b_{\mathbb{X}}\,, && \text{(4.3.11a)} \\ \Delta x_{\mathbb{F}}^{Y} &= T^{-1}\left(\Delta b_{\mathbb{A}} - C_{\mathbb{X}}\Delta x_{\mathbb{X}}\right), && \text{(4.3.11b)} \\ \Delta x_{\mathbb{F}}^{Z} &= -R^{-1}(R')^{-1}\left(\Delta g_{\mathbb{F}}^{Z} + Z'_{\mathbb{F}}\left(H_{\mathbb{F}}Y_{\mathbb{F}}\Delta x_{\mathbb{F}}^{Y} + H_{\mathrm{M}}\Delta x_{\mathbb{X}}\right)\right), && \text{(4.3.11c)} \\ \Delta y_{\mathbb{A}} &= (T')^{-1}\left(\Delta g_{\mathbb{F}}^{Y} + Y'_{\mathbb{F}}\left(H_{\mathbb{F}}\Delta x_{\mathbb{F}} + H_{\mathrm{M}}\Delta x_{\mathbb{X}}\right)\right), && \text{(4.3.11d)} \\ \Delta y_{\mathbb{X}} &= H'_{\mathrm{M}}\Delta x_{\mathbb{F}} + H_{\mathbb{X}}\Delta x_{\mathbb{X}} + C'_{\mathbb{X}}\Delta y_{\mathbb{A}} + \Delta g_{\mathbb{X}}\,, && \text{(4.3.11e)} \\ \text{with}\quad \Delta x_{\mathbb{F}} &\stackrel{\text{def}}{=} Z_{\mathbb{F}}\Delta x_{\mathbb{F}}^{Z} + Y_{\mathbb{F}}\Delta x_{\mathbb{F}}^{Y}\,. && \text{(4.3.11f)} \end{aligned}$$

These calculations can be simplified by exploiting common subexpressions. Moreover, it is possible to accelerate the calculation if the currently active bounds $b_{\mathbb{X}}$ or constraints' bounds $b_{\mathbb{A}}$ (cf. Eq. (4.3.2)) do not depend on w_0, and thus $\Delta b_{\mathbb{X}} = \mathbb{0}$ or $\Delta b_{\mathbb{A}} = \mathbb{0}$.

4.3.3 Matrix Updates

Until the solution and a corresponding optimal working set is found, the current working set must be modified by adding or removing a bound or a constraint in each iteration. Furthermore, decompositions of the projected Hessian matrix $Z_{\mathbb{F}}' H Z_{\mathbb{F}}$ and the active constraints matrix $C_{\mathbb{F}}$ have to be maintained in order to efficiently compute new step directions. However, re-computation in each iteration would foil this benefit because calculations of both the Cholesky decomposition as well as the TQ factorisation require $\mathcal{O}(n^3)$ floating-point operations. Instead, because a single working set change affects these decompositions in a rather simple way, it is possible to reduce the effort to $\mathcal{O}(n^2)$ floating-point operations (per iteration) by using so-called *matrix updates*.
In this subsection, we will describe the matrix updates used in our implementation which are specially tailored to the context where bounds and constraints are distinguished. The presentation is based on [39], complexity issues are examined in Section 4.6.1. We start with a brief summary of Givens plane rotations which are a necessary prerequisite for the proposed matrix updates.

Givens Plane Rotations

A *Givens plane rotation* can be expressed as a matrix of the following form (cf. [43] and e.g. [46]):

$$O^{i,j}(\varphi) \stackrel{\text{def}}{=} \begin{pmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & \cos\varphi & & & & \sin\varphi & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & -\sin\varphi & & & & \cos\varphi & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{pmatrix}, \quad \varphi \in [0, 2\pi)\,. \tag{4.3.12}$$

Herein φ can be chosen is such a way that the j-th component of a vector $v \in \mathbb{R}^n$ becomes zero if v is premultiplied by $O^{i,j}(\varphi)$:

$$\big(O^{i,j}(\varphi)v\big)_k = \begin{cases} v_i \cos\varphi + v_j \sin\varphi & \text{if } k = i \\ -v_i \sin\varphi + v_j \cos\varphi & \text{if } k = j \\ v_k & \text{else} \end{cases} \tag{4.3.13}$$

which implies that

$$\big(O^{i,j}(\varphi)v\big)_j = 0 \iff \cos\varphi = \frac{v_i}{\sqrt{v_i^2 + v_j^2}} \;\wedge\; \sin\varphi = \frac{v_j}{\sqrt{v_i^2 + v_j^2}}\,. \tag{4.3.14}$$

By definition every matrix $O^{i,j}(\varphi)$ is orthonormal with determinant one. Therefore premultiplication by $O^{i,j}(\varphi)$ can be interpreted as a counterclockwise rotation in the (i,j) coordinate plane, which explains the name.

Successive application of Givens plane rotations allows to introduce certain zero patterns into a vector or, especially, another matrix. For example it is possible to transform an arbitrary matrix into an upper triangular matrix. While this can also be done, even at lower computational costs, via *Gaussian elimination*, a very important advantage of Givens plane rotations is that they are particularly numerically stable because of their orthonormality.
In practice, formulae different from those given in (4.3.14) for computation of $\cos\varphi$ and $\sin\varphi$ are used in order to prevent possible overflow [21]. Furthermore, computational savings are possible when multiplying $O^{i,j}(\varphi)$ with a matrix. Of course, from (4.3.13) it is evident that only two rows (or columns, if $O^{i,j}(\varphi)$ is multiplied from the right) have to be involved into the calculation. But moreover, there are ways to reduce the number of multiplications neccessary per step from four, as in (4.3.13), to three or even to two—so-called *fast plane rotations* [47], [1]. However, this comes at the expense of considerable overhead which can, even in the case of large matrices, outweigh the benefit [48]. In our implementation we tried (4.3.13), which requires four multiplications and two additions, and a variant described in [21], which requires three multiplications and three additions, and found both almost equally efficient.

Matrix Permutations

When applying matrix updates it is sometimes helpful to permute the vector of free variables $x_\mathbb{F}$ which results in rearrangements of rows or columns of the involved matrices. Therefore, before descriptions of the actual matrix updates are given, we show the mathematical justification of these permutations:
Permutation of the vector of free variables $x_\mathbb{F}$ is equivalent to multiplying it with a non-singular square matrix P:

$$\hat{x}_\mathbb{F} \stackrel{\text{def}}{=} P x_\mathbb{F}\,, \quad \text{where } P \in \{0,1\}^{n\times n},\ P'P = \mathrm{Id}\,. \tag{4.3.15}$$

This leads to the following expressions:

$$C_\mathbb{F} x_\mathbb{F} = C_\mathbb{F} P' P x_\mathbb{F} = \hat{C}_\mathbb{F} \hat{x}_\mathbb{F}, \tag{4.3.16a}$$
$$C_\mathbb{F} Q_\mathbb{F} = C_\mathbb{F} P' P Q_\mathbb{F} = \hat{C}_\mathbb{F} \hat{Q}_\mathbb{F}, \tag{4.3.16b}$$
$$Z'_\mathbb{F} H_\mathbb{F} Z_\mathbb{F} = Z'_\mathbb{F} P' P H_\mathbb{F} P' P Z_\mathbb{F} = \hat{Z}'_\mathbb{F} \hat{H}_\mathbb{F} \hat{Z}_\mathbb{F}, \tag{4.3.16c}$$
$$\text{where } \hat{C}_\mathbb{F} \stackrel{\text{def}}{=} C_\mathbb{F} P',\ \hat{Q}_\mathbb{F} \stackrel{\text{def}}{=} P Q_\mathbb{F},\ \hat{Z}_\mathbb{F} \stackrel{\text{def}}{=} P Z_\mathbb{F},\ \hat{H}_\mathbb{F} \stackrel{\text{def}}{=} P H_\mathbb{F} P'. \tag{4.3.16d}$$

This means that we have to rearrange the columns of $C_\mathbb{F}$, the rows of $Q_\mathbb{F}$ and $Z_\mathbb{F}$ (and $Y_\mathbb{F}$) as well as the rows and the colums of $H_\mathbb{F}$ in the same way as the components of vector $x_\mathbb{F}$; the matrices R and T are not affected. Because the resulting transformed QP is completely equivalent to the original one we omit matrix P from now on.
These permutations are implemented by means of an *index list* of free variables which is realised as a double linked list. Elements of $x_\mathbb{F}$ and the mentioned matrices are accessed via this index list which is necessary anyway if explicit re-storing while working with submatrices shall be avoided. The latter is also the reason why an index list of active constraints is held, too. It is obvious that the order of (active) constraints within a QP is arbitrary.

When illustrating certain matrix modification processes the following symbols are used:

- $\times$ denotes a non-zero element that is not modified,
- $*$ denotes a non-zero element that is modified,
- $\bigcirc$ denotes a previously non-zero element that is annihilated,
- $\circledast$ denotes a previously zero element that is filled in,
- $\cdot$ denotes a zero element that is not modified (same as blank),
- $-$ denotes an element of a row or a column to be removed from a matrix.

Adding a Constraint to Working Set

First, we consider the case when a constraint is added to the working set. According to the above-mentioned remarks on matrix permutations we assume without loss of generality that the newly active constraint is added as the last row of C. Thus, the row number of C $(= G_{\mathbb{A}})$, the column number of $Y_{\mathbb{F}}$ and the dimension of T increase by one while the column number of $Z_{\mathbb{F}}$ decreases by one. Let

$$c'_{\text{new}} = \begin{pmatrix} c_{\mathbb{F}}^{\text{new}\prime} & c_{\mathbb{X}}^{\text{new}\prime} \end{pmatrix} \in \mathbb{R}^n, \tag{4.3.17a}$$

$$t'_{\text{new}} = \begin{pmatrix} t_Z^{\text{new}\prime} & t_Y^{\text{new}\prime} \end{pmatrix} \stackrel{\text{def}}{=} c_{\mathbb{F}}^{\text{new}\prime} Q_{\mathbb{F}} \in \mathbb{R}^{n_{\mathbb{F}}} \tag{4.3.17b}$$

denote the row of C corresponding to the newly active constraint (again, optimisation variables are permuted properly) and the new last row of T, respectively. Then the following equation holds:

$$C_{\mathbb{F}}^{\text{new}} Q_{\mathbb{F}} = \begin{pmatrix} C_{\mathbb{F}} \\ c_{\mathbb{F}}^{\text{new}\prime} \end{pmatrix} Q_{\mathbb{F}} = \begin{pmatrix} \mathbb{0} & T \\ t_Z^{\text{new}\prime} & t_Y^{\text{new}\prime} \end{pmatrix}. \tag{4.3.18}$$

In order to transform the right hand side of (4.3.18) into the reverse lower triangalur matrix T_{new} a sequence of Givens plane rotations is applied from the right. For the case $n_{\mathbb{F}} = 7$ and $n_{\mathbb{A}} = 3$ $(n_{\mathbb{A}}^{\text{new}} = 4)$ this can be illustrated as follows:

$$\begin{pmatrix} & & & & & & \times \\ & & & & & \times & \times \\ & & & & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & & & & & \times \\ & & & & & \times & \times \\ & & & & \times & \times & \times \\ \bigcirc & * & \times & \times & \times & \times & \times \end{pmatrix}$$

$$\rightsquigarrow \begin{pmatrix} & & & & & & \times \\ & & & & & \times & \times \\ & & & & \times & \times & \times \\ \cdot & \bigcirc & * & \times & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & & & & & \times \\ & & & & & \times & \times \\ & & & & \times & \times & \times \\ \cdot & \cdot & \bigcirc & * & \times & \times & \times \end{pmatrix}$$

Using the notation introduced in Eq. (4.3.12) this transformation formally means

$$T_{\text{new}} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbb{0} & T \\ t_Z^{\text{new}\prime} & t_Y^{\text{new}\prime} \end{pmatrix} \cdot O^{2,1}(\varphi_1) \cdot \ldots \cdot O^{n_Z, n_Z - 1}(\varphi_{n_Z - 1}) \tag{4.3.19a}$$

$$= \begin{pmatrix} \mathbb{0} & 0 & T \\ \mathbb{0} & \theta_{\text{new}} & t_Y^{\text{new}\prime} \end{pmatrix}, \quad \theta_{\text{new}} \neq 0,$$

$$Q_{\mathbb{F}}^{\text{new}} \stackrel{\text{def}}{=} Q_{\mathbb{F}} \cdot O^{2,1}(\varphi_1) \cdot \ldots \cdot O^{n_Z, n_Z - 1}(\varphi_{n_Z - 1}). \tag{4.3.19b}$$

Note that $Q_{\mathbb{F}}^{\text{new}}$ is also an orthonormal matrix since all Givens plane rotation matrices are orthonormal. By definition, the null space basis matrix $Z_{\mathbb{F}}$ is transformed the same way as

$Q_{\mathbb{F}}$ in Eq. (4.3.19b). Note, however, that the rightmost column of $Z_{\mathbb{F}}$ becomes the leftmost column of $Y_{\mathbb{F}}^{\text{new}}$ since the dimension of the null space decreased by one when adding a new constraint to the working set[3]. The transformation of $Z_{\mathbb{F}}$ also affects the Cholesky factor of the reduced Hessian matrix $Z_{\mathbb{F}}' H Z_{\mathbb{F}}$ in the following way:

$$\begin{pmatrix} \times & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \circ & * & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \cdot & \circ & * & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \cdot & \cdot & \circ & * \end{pmatrix}$$

$$\begin{pmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} * & * & \times & \times \\ \circledast & * & \times & \times \\ & & \times & \times \\ & & & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \times & * & * & \times \\ \times & * & * & \times \\ & \circledast & * & \times \\ & & & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \times & \times & * & * \\ \times & \times & * & * \\ & \times & * & * \\ & & \circledast & * \end{pmatrix}$$

Again, this illustration depicts the case $n_{\mathbb{F}} = 7$, $n_{\mathbb{A}} = 3$ ($n_{\mathbb{A}}^{\text{new}} = 4$) and $n_{\mathbb{Z}} = 4$ ($n_{\mathbb{Z}}^{\text{new}} = 3$) where besides matrix R also the vector $t_Z^{\text{new}\prime}$ is shown at the top for clarity. The chosen order of the Givens plane rotations implies that the upper triangular form of matrix R is only slightly destroyed: only one additional subdiagonal element is introduced in each column of R_{int}, which denotes the resulting intermediate Cholesky factor. In order to restore the upper triangular form another sequence of Givens plane rotations is applied to R_{int}:

$$\begin{pmatrix} \times & \times & \times & - \\ \times & \times & \times & - \\ & \times & \times & - \\ & & \times & - \end{pmatrix} \rightsquigarrow \begin{pmatrix} * & * & * \\ \circ & * & * \\ & \times & \times \\ & & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \times & \times & \times \\ \cdot & * & * \\ & \circ & * \\ & & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \times & \times & \times \\ \cdot & \times & \times \\ & \cdot & * \\ & & \circ \end{pmatrix}$$

Algebraically these transformations of R can be expressed as

$$\begin{aligned} R_{\text{int}} &\stackrel{\text{def}}{=} H^{\frac{1}{2}} Z_{\mathbb{F}}^{\text{new}} && \text{(4.3.20a)} \\ &= H^{\frac{1}{2}} Z_{\mathbb{F}} \cdot O^{2,1}(\varphi_1) \cdot \ldots \cdot O^{n_{\mathbb{Z}}, n_{\mathbb{Z}}-1}(\varphi_{n_{\mathbb{Z}}-1}) \cdot P\,, \\ R_{\text{new}} &\stackrel{\text{def}}{=} O^{1,2}(\varphi_{n_{\mathbb{Z}}}) \cdot \ldots \cdot O^{n_{\mathbb{Z}}-1, n_{\mathbb{Z}}}(\varphi_{2(n_{\mathbb{Z}}-1)}) \cdot R_{\text{int}}\,, && \text{(4.3.20b)} \end{aligned}$$

where P is a projection matrix which removes the rightmost column. Furthermore, if we define $O \stackrel{\text{def}}{=} O^{1,2}(\varphi_{n_{\mathbb{Z}}}) \cdot \ldots \cdot O^{n_{\mathbb{Z}}-1, n_{\mathbb{Z}}}(\varphi_{2(n_{\mathbb{Z}}-1)})$, it is obvious that the second sequence of Givens plane rotations does not affect other matrices:

$$Z_{\mathbb{F}}^{\text{new}\prime} H Z_{\mathbb{F}}^{\text{new}} \;=\; R_{\text{int}}' R_{\text{int}} \;=\; R_{\text{new}}' \underbrace{O O'}_{=\text{Id}} R_{\text{new}} \;=\; R_{\text{new}}' R_{\text{new}}\,. \qquad \text{(4.3.21)}$$

Adding a Bound to Working Set

When adding a bound to the working set we can assume that the variable to be fixed corresponds to the last column of the matrix C by applying an appropriate permutation. Thus, the column number of C and $Z_{\mathbb{F}}$ as well as the dimension of $Q_{\mathbb{F}}$ are decreased by one; the dimension of T does not change. Addition of a bound on the last free variable

[3] This is actually only true under the asumption that $C_{\mathbb{F}}^{\text{new}}$ has full row rank. Section 4.5.1 describes how this can be maintained.

appends the (transposed) $n_{\mathbb{F}}$-th coordinate vector $\left(e'_{n_{\mathbb{F}}}\ \mathbb{0}\right) \in \mathbb{R}^n$, $e'_{n_{\mathbb{F}}} \in \mathbb{R}^{n_{\mathbb{F}}}$, at the top of the augmented active constraints matrix:

$$\begin{pmatrix} e'_{n_{\mathbb{F}}} & \mathbb{0} \\ \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \\ C_{\mathbb{F}} & C_{\mathbb{X}} \end{pmatrix} \begin{pmatrix} Q_{\mathbb{F}} & \mathbb{0} \\ \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \end{pmatrix} = \begin{pmatrix} t_Z^{\mathrm{new}\prime} & t_Y^{\mathrm{new}\prime} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \\ \mathbb{0} & T & C_{\mathbb{X}} \end{pmatrix}. \tag{4.3.22}$$

The updated TQ factorisation is obtained by reducing the topmost row of the right hand side matrix of Eq. (4.3.22) to the $n_{\mathbb{F}}$-th coordinate vector via a sequence of Givens plane rotations:

$$\begin{pmatrix} Q_{\mathbb{F}}^{\mathrm{new}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 1 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \end{pmatrix} \overset{\mathrm{def}}{=} \begin{pmatrix} Q_{\mathbb{F}} & \mathbb{0} \\ \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \end{pmatrix} \cdot O^{2,1}(\varphi_1) \cdot \ldots \cdot O^{n_{\mathbb{F}},n_{\mathbb{F}}-1}(\varphi_{n_{\mathbb{F}}-1}), \tag{4.3.23a}$$

$$\begin{pmatrix} \mathbb{0} & \mathbb{0} & 1 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \\ \mathbb{0} & T_{\mathrm{new}} & C_{n_{\mathbb{F}}} & C_{\mathbb{X}} \end{pmatrix} \overset{\mathrm{def}}{=} \begin{pmatrix} t_Z^{\mathrm{new}\prime} & t_Y^{\mathrm{new}\prime} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \\ \mathbb{0} & T & C_{\mathbb{X}} \end{pmatrix} \cdot O^{2,1}(\varphi_1) \cdot \ldots \cdot O^{n_{\mathbb{F}},n_{\mathbb{F}}-1}(\varphi_{n_{\mathbb{F}}-1}), \tag{4.3.23b}$$

where $C_{n_{\mathbb{F}}}$ denotes the column of C which corresponds to the newly fixed variable.

The first $(n_{\mathrm{Z}}-1)$ Givens plane rotations $O^{2,1}(\varphi_1)\cdot\ldots\cdot O^{n_{\mathrm{Z}},n_{\mathrm{Z}}-1}(\varphi_{n_{\mathrm{Z}}-1})$, $n_{\mathrm{Z}} \leq n_{\mathbb{F}}$, alter the columns of $Q_{\mathbb{F}}$ (i.e. $Z_{\mathbb{F}}$) in the same way as described above for the case where a constraint is added to the working set. Therefore, another sequence of Givens plane rotations has to be applied in order to restore the upper triangular form of R_{int}, too.
The last $(n_{\mathbb{F}} - n_{\mathrm{Z}})$ Givens plane rotations $O^{n_{\mathrm{Z}}+1,n_{\mathrm{Z}}}(\varphi_{n_{\mathrm{Z}}}) \cdot \ldots \cdot O^{n_{\mathbb{F}},n_{\mathbb{F}}-1}(\varphi_{n_{\mathbb{F}}-1})$ have the effect of filling in elements above the reverse diagonal of matrix T, thereby shifting it one position to the left and transforming it into T_{new}. We picture this process for $n_{\mathbb{F}} = 4$ ($n_{\mathbb{F}}^{\mathrm{new}} = 3$), $n_{\mathrm{Z}} = 1$ ($n_{\mathrm{Z}}^{\mathrm{new}} = 0$) and $n_{\mathbb{A}} = 3$; the topmost row of the right hand side matrix of Eq. (4.3.22) is shown at the top:

$$\begin{pmatrix} \times & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \circ & * & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \cdot & \circ & * & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} \cdot & \cdot & \circ & * \end{pmatrix}$$

$$\begin{pmatrix} & & & \times \\ & & \times & \times \\ & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & & \times \\ & & \times & \times \\ \circledast & * & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & & \times \\ & \circledast & * & \times \\ \times & * & * & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & \circledast & * \\ & \times & * & * \\ \times & \times & * & * \end{pmatrix}$$

Removing a Constraint from Working Set

We consider the situation where the i-th, $1 \leq i \leq n_{\mathbb{A}}$, of the currently active constraints shall be removed from the working set. Then the row number of C and the dimension of T are decreased by one; the column number of $Z_{\mathbb{F}}$ and the dimension of R increase by one. First, the i-th row is removed from both $C_{\mathbb{F}}$ and T leading to the matrices $C_{\mathbb{F}}^{\mathrm{new}} \in \mathbb{R}^{(n_{\mathbb{A}}-1)\times n_{\mathbb{F}}}$ and $T_{\mathrm{int}} \in \mathbb{R}^{(n_{\mathbb{A}}-1)\times n_{\mathbb{A}}}$ satisfying

$$C_{\mathbb{F}}^{\mathrm{new}} Q_{\mathbb{F}} = \begin{pmatrix} \mathbb{0} & T_{\mathrm{int}} \end{pmatrix}. \tag{4.3.24}$$

Next, T_{int} is transformed to reverse upper triangular form which is achieved via Givens plane rotations applied to the columns 1 through i:

$$\begin{pmatrix} \mathbb{0} & T_{\text{new}} \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} \mathbb{0} & T_{\text{int}} \end{pmatrix} \cdot O^{n_Z+i,n_Z+i-1}(\varphi_{i-1}) \cdot \ldots \cdot O^{n_Z+2,n_Z+1}(\varphi_1)\,, \tag{4.3.25a}$$

$$Q_{\mathbb{F}}^{\text{new}} \overset{\text{def}}{=} O^{n_Z+i,n_Z+i-1}(\varphi_{i-1}) \cdot \ldots \cdot O^{n_Z+2,n_Z+1}(\varphi_1)\,. \tag{4.3.25b}$$

We illustrate the transformation of T for the case $n_{\mathbb{A}} = 4$ ($n_{\mathbb{A}}^{\text{new}} = 3$) and $i = 3$:

$$\begin{pmatrix} & & & \times \\ & & \times & \times \\ - & - & - & - \\ \times & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & & \times \\ & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & & \times \\ & \bigcirc & * & \times \\ \times & * & * & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & & \times \\ & \cdot & \times & \times \\ \bigcirc & * & \times & \times \end{pmatrix}$$

Equation (4.3.25b) shows that $Z_{\mathbb{F}}$ within $Q_{\mathbb{F}}$ is not altered by the mentioned Givens plane rotations. Thus, $Z_{\mathbb{F}}^{\text{new}}$ is identical to $Z_{\mathbb{F}}$ except for the additional rightmost column $z_{\mathbb{F}}^{\text{new}} \in \mathbb{R}^{n_{\mathbb{F}}}$ which is a linear combination of columns 1 through i of $Y_{\mathbb{F}}$. This fact provides an efficient possibility to calculate the new Cholesky factor R_{new} from R (with $r_{\text{new}} \in \mathbb{R}^{n_Z}$, $\varrho_{\text{new}} \in \mathbb{R}_{>0}$):

$$Z_{\mathbb{F}}^{\text{new}\prime} H_{\mathbb{F}}^{\text{new}} Z_{\mathbb{F}}^{\text{new}} = R_{\text{new}}' R_{\text{new}} \tag{4.3.26a}$$

$$\iff \begin{pmatrix} Z_{\mathbb{F}}' \\ z_{\mathbb{F}}^{\text{new}\prime} \end{pmatrix} H_{\mathbb{F}} \begin{pmatrix} Z_{\mathbb{F}} & z_{\mathbb{F}}^{\text{new}} \end{pmatrix} = \begin{pmatrix} R' & \mathbb{0} \\ r_{\text{new}}' & \varrho_{\text{new}} \end{pmatrix} \begin{pmatrix} R & r_{\text{new}} \\ \mathbb{0} & \varrho_{\text{new}} \end{pmatrix} \tag{4.3.26b}$$

$$\iff \begin{pmatrix} Z_{\mathbb{F}}' H_{\mathbb{F}} Z_{\mathbb{F}} & Z_{\mathbb{F}}' H_{\mathbb{F}} z_{\mathbb{F}}^{\text{new}} \\ z_{\mathbb{F}}^{\text{new}\prime} H_{\mathbb{F}} Z_{\mathbb{F}} & z_{\mathbb{F}}^{\text{new}\prime} H_{\mathbb{F}} z_{\mathbb{F}}^{\text{new}} \end{pmatrix} = \begin{pmatrix} R'R & R' r_{\text{new}} \\ r_{\text{new}}' R & r_{\text{new}}' r_{\text{new}} + \varrho_{\text{new}}^2 \end{pmatrix} \tag{4.3.26c}$$

$$\iff r_{\text{new}} = \left(R'\right)^{-1} Z_{\mathbb{F}}' H_{\mathbb{F}} z_{\mathbb{F}}^{\text{new}} \quad \wedge \quad \varrho_{\text{new}} = \sqrt{r_{\text{new}}' r_{\text{new}} - z_{\mathbb{F}}^{\text{new}\prime} H_{\mathbb{F}} z_{\mathbb{F}}^{\text{new}}}\,. \tag{4.3.26d}$$

Note that $H_{\mathbb{F}}^{\text{new}} = H_{\mathbb{F}}$ and that the radicand within Eq. (4.3.26d) is positive as long as $Z_{\mathbb{F}}^{\text{new}\prime} H_{\mathbb{F}}^{\text{new}} Z_{\mathbb{F}}^{\text{new}} \in \mathcal{S}_{\succ 0}$. This both necessary and sufficient criterion can actually be used to check positive definiteness of the projected Hessian matrix during the runtime. Moreover, it is worth mentioning that calculation of the new Cholesky factor R_{new} via (4.3.26) is only possible if $z_{\mathbb{F}}^{\text{new}}$ is appended as the rightmost column of $Z_{\mathbb{F}}^{\text{new}}$. This fact motivates the usage of a TQ decomposition because $z_{\mathbb{F}}^{\text{new}}$ would be added as the leftmost column if we were using the usual QR decomposition instead.

Removing a Bound from Working Set

Removing a bound from the working set means to free a previously fixed variable. Therefore, the column number of C and $\mathbb{Z}_{\mathbb{F}}$ as well as the dimension of $Q_{\mathbb{F}}$ and R are increased by one; the dimension of T is unaltered. Applying a suitable permutation, we can assume without loss of generality that the $(n_{\mathbb{F}}+1)$-th variable, i.e. the first fixed one, is to be freed from its bound. Then the leftmost column of $C_{\mathbb{X}}$ becomes the rightmost column $c_{\mathbb{F}}^{\text{new}} \in \mathbb{R}^{n_{\mathbb{A}}}$ of $C_{\mathbb{F}}^{\text{new}}$:

$$\begin{pmatrix} \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}^{\text{new}}} \\ C_{\mathbb{F}} & c_{\mathbb{F}}^{\text{new}} & C_{\mathbb{X}}^{\text{new}} \end{pmatrix} \begin{pmatrix} Q_{\mathbb{F}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & 1 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}^{\text{new}}} \end{pmatrix} = \begin{pmatrix} \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}^{\text{new}}} \\ \mathbb{0} & T & c_{\mathbb{F}}^{\text{new}} & C_{\mathbb{X}}^{\text{new}} \end{pmatrix}, \tag{4.3.27}$$

where $C_{\mathbb{X}}^{\text{new}} \in \mathbb{R}^{n_{\mathbb{A}} \times (n_{\mathbb{X}}-1)}$ denotes matrix $C_{\mathbb{X}}$ without column $c_{\mathbb{F}}^{\text{new}}$.
Thus, a sequence of Givens plane rotations is used in order to reduce $(T\ c_{\mathbb{F}}^{\text{new}})$ to reverse lower triangular form (illustrated for $n_{\mathbb{A}} = 3$):

$$\begin{pmatrix} & & \times & \times \\ & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & \circ & * \\ & \times & * & * \\ \times & \times & * & * \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & \cdot & \times \\ & \circ & * & \times \\ \times & * & * & \times \end{pmatrix} \rightsquigarrow \begin{pmatrix} & & \cdot & \times \\ & \cdot & \times & \times \\ \circ & * & \times & \times \end{pmatrix}$$

Algebraically, the effects on T and $Q_{\mathbb{F}}$ can be expressed as

$$\begin{pmatrix} \mathbb{0} & T_{\text{new}} \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} \mathbb{0} & T_{\text{int}} & c_{\mathbb{F}}^{\text{new}} \end{pmatrix} \cdot O^{n_{\mathbb{F}}+1,n_{\mathbb{F}}}(\varphi_1) \cdot \ldots \cdot O^{n_{\mathbb{Z}}+2,n_{\mathbb{Z}}+1}(\varphi_{n_{\mathbb{F}}-n_{\mathbb{Z}}}), \tag{4.3.28a}$$

$$Q_{\mathbb{F}}^{\text{new}} \overset{\text{def}}{=} \begin{pmatrix} Q_{\mathbb{F}} & \mathbb{0} \\ \mathbb{0} & 1 \end{pmatrix} \cdot O^{n_{\mathbb{F}}+1,n_{\mathbb{F}}}(\varphi_1) \cdot \ldots \cdot O^{n_{\mathbb{Z}}+2,n_{\mathbb{Z}}+1}(\varphi_{n_{\mathbb{F}}-n_{\mathbb{Z}}}). \tag{4.3.28b}$$

This sequence of Givens plane rotations does not affect the old null space basis matrix $Z_{\mathbb{F}}$ but the new rightmost column of $Z_{\mathbb{F}}^{\text{new}}$:

$$Z_{\mathbb{F}}^{\text{new}} \overset{\text{def}}{=} \begin{pmatrix} Z_{\mathbb{F}} & z_{\mathbb{F}}^{\text{new}} \\ \mathbb{0} & \zeta_{\mathbb{F}}^{\text{new}} \end{pmatrix}, \quad \zeta_{\mathbb{F}}^{\text{new}} \in \mathbb{R} \setminus \{0\}\ . \tag{4.3.29}$$

This change of $\mathbb{Z}_{\mathbb{F}}$ also causes $H_{\mathbb{F}}$ to be modified. Like in the case where a constraint is removed from the working set, no fresh Cholesky decomposition must be performed but the following efficient update scheme can be applied instead:

$$Z_{\mathbb{F}}^{\text{new}\prime} H_{\mathbb{F}}^{\text{new}} Z_{\mathbb{F}}^{\text{new}} = R_{\text{new}}' R_{\text{new}} \tag{4.3.30a}$$

$$\iff \begin{pmatrix} Z_{\mathbb{F}}' & \mathbb{0} \\ z_{\mathbb{F}}^{\text{new}\prime} & \zeta_{\mathbb{F}}^{\text{new}} \end{pmatrix} \begin{pmatrix} H_{\mathbb{F}} & h_{\mathbb{F}}^{\text{new}} \\ h_{\mathbb{F}}^{\text{new}\prime} & \eta_{\text{new}} \end{pmatrix} \begin{pmatrix} Z_{\mathbb{F}} & z_{\mathbb{F}}^{\text{new}} \\ \mathbb{0} & \zeta_{\mathbb{F}}^{\text{new}} \end{pmatrix} = \begin{pmatrix} R' & \mathbb{0} \\ r_{\text{new}}' & \varrho_{\text{new}} \end{pmatrix} \begin{pmatrix} R & r_{\text{new}} \\ \mathbb{0} & \varrho_{\text{new}} \end{pmatrix} \tag{4.3.30b}$$

$$\iff r_{\text{new}} = (R')^{-1} Z_{\mathbb{F}}' (H_{\mathbb{F}} z_{\mathbb{F}}^{\text{new}} + \zeta_{\mathbb{F}}^{\text{new}} h_{\mathbb{F}}^{\text{new}}) \tag{4.3.30c}$$

$$\wedge \quad \varrho_{\text{new}} = \sqrt{z_{\mathbb{F}}^{\text{new}\prime} (H_{\mathbb{F}} z_{\mathbb{F}}^{\text{new}} + 2\zeta_{\mathbb{F}}^{\text{new}} h_{\mathbb{F}}^{\text{new}}) + \eta_{\mathbb{F}}^{\text{new}} (\zeta_{\mathbb{F}}^{\text{new}})^2 - r_{\text{new}}' r_{\text{new}}}. \tag{4.3.30d}$$

Again, the radicand within Eq. (4.3.30d) is positive provided that $Z_{\mathbb{F}}^{\text{new}\prime} H_{\mathbb{F}}^{\text{new}} Z_{\mathbb{F}}^{\text{new}} \in \mathcal{S}_{\succ 0}$.

4.4 Initialisation

In order to initialise our online active set strategy an optimal solution pair of the initial QP and a corresponding working set $\mathbb{A}$ must be available. So the question naturally arises of how to obtain this information. One possibility would be to solve the initial QP by means of a standard active set QP solver. But this would be rather inconvenient since all the effort needed to implement and setup such a solver would be necessary just for the solution of the very first QP. Instead, our online active set strategy allows for an easy workaround: one simply has to set up a QP whose solution is known. A straightforward idea is to "solve" the following QP:

$$\min_x \quad \tfrac{1}{2} x' H x \tag{4.4.1a}$$

$$\text{s.t.} \quad -b \leq\ x\ \leq b, \tag{4.4.1b}$$

$$-b \leq Gx \leq b, \tag{4.4.1c}$$

where the gradient is set to zero and $b \geq \mathbb{0}$ is arbitrary.

Lemma 4.1 (initialisation): *If $b \geq \mathbb{0}$ then $(\mathbb{0}, \mathbb{0})$ is a primal-dual solution pair of the quadratic program* (4.4.1) *with corresponding working set $\mathbb{A} = \emptyset$.* ○

Proof: If $\mathbb{A}$ is assumed to be empty the KKT conditions of Theorem 2.5 have the following form:

$$\begin{aligned} Hx^{\text{opt}} &= \mathbb{0}\,, \\ y^{\text{opt}} &= \mathbb{0}\,, \\ -b \quad \leq \quad Gx^{\text{opt}} &\leq \quad b\,. \end{aligned}$$

It is obvious that they are satisfied by the choice $\left(x^{\text{opt}}, y^{\text{opt}}\right) \stackrel{\text{def}}{=} (\mathbb{0}, \mathbb{0})$. □

Therefore we can start from $(\mathbb{0}, \mathbb{0})$ and use our usual homotopy to go towards the solution of the initial QP.

This strategy also works for equality constraints: Let us assume that our initial QP comprises the constraint

$$\gamma \leq G_i x \leq \gamma\,, \quad \gamma \in \mathbb{R}\,, \quad i \in \{1, \ldots, m\}\,. \tag{4.4.2}$$

If this equality constraint is relaxed to the inequality constraint

$$-\beta \leq G_i x \leq \beta\,, \quad \beta \in \mathbb{R}_{\geq 0} \tag{4.4.3}$$

$\pm\beta$ will be both shifted towards γ. As soon as one of the constraint's bounds becomes active, and this must happen by the time they coincide, the constraint will not be considered when determing the maximum dual stepsize $\tau_{\max}^{\text{dual}}$ anymore and thus will stay active for all following iterations.
However, if there are $n_{\text{EC}} \leq \min\{n, m\}$ equality constraints this procedure leads to n_{EC} unnecessary working set changes since all equality constraints will finally become active. In order to avoid this, it is possible to start at $(\mathbb{0}, \mathbb{0})$ and include the indices of all equality constraints into the initial working set $\mathbb{A}$(and the corresponding part of b to zero). Similar to Lemma 4.1, it can be shown that $(\mathbb{0}, \mathbb{0})$ is still a primal-dual solution. Of course, in this case the TQ factorisation $C_{\mathbb{F}} = (\mathbb{0}\ T)\, Q'_{\mathbb{F}}$ as well as the Cholesky decomposition has to be calculated before starting the initial homotopy.

4.5 Degeneracy Handling

4.5.1 Linear Dependence of Constraints

Our algorithm requires that the KKT matrix in Eq. (4.3.7) is nonsingular. Because of the assumed positive definiteness of H this property holds if and only if the augmented active constraints matrix

$$\begin{pmatrix} \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} \\ C_{\mathbb{F}} & C_{\mathbb{X}} \end{pmatrix}$$

has full row rank (see Lemma 2.2). Since deletion of a row cannot lead to rank deficiency, linear independence only needs to be ensured if a row is added to the augmented active constraints matrix, i.e. if a bound or a constraint is added to the working set.

In order to clarify the idea, handling of linear dependence is described for QPs where bounds are treated as ordinary constraints, first. Refinements of this approach tailored to our problem formulation and the way we solve the KKT system will be presented afterwards. In the case that constraint $j \notin \mathbb{A}$ shall be added to the working set [11] proposed the solution of the following auxiliary system as a test if G'_j and the rows of $G_{\mathbb{A}}$ $(= C)$ are linearly independent:

$$\begin{pmatrix} H & G'_{\mathbb{A}} \\ G_{\mathbb{A}} & \mathbb{0} \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} G'_j \\ \mathbb{0} \end{pmatrix}, \tag{4.5.1}$$

where $p \in \mathbb{R}^n$ and $q \in \mathbb{R}^{n_{\mathbb{A}}}$.

Lemma 4.2 (linear independence check): *Provided that $G_{\mathbb{A}}$ has full row rank, G'_j and the rows of $G_{\mathbb{A}}$ are linearly independent if and only if Eq.* (4.5.1) *has a solution with $p \neq \mathbb{0}$.* ○

Proof: Since $G_{\mathbb{A}}$ is assumed to have full row rank, linear dependence of G'_j and the columns of $G_{\mathbb{A}}$ is equivalent to

$$\exists\, \hat{q} \in \mathbb{R}^{n_{\mathbb{A}}} :\; G'_j + \sum_{i \in \mathbb{A}} \hat{q}_i G'_i = \mathbb{0}\,. \tag{4.5.2}$$

Thus, if G'_j and the columns of $G_{\mathbb{A}}$ are linearly dependent $(p, q) \stackrel{\text{def}}{=} (\mathbb{0}, -\hat{q})$ is obviously a solution of (4.5.1). According to Lemma 2.2 this solution is unique which implies that Eq. (4.5.1) has no solution with $p \neq \mathbb{0}$.
On the other hand, if G'_j and the columns of $G_{\mathbb{A}}$ are linearly independent (and thus $G'_j \neq \mathbb{0}$) there exists no $\hat{q} \in \mathbb{R}^{n_{\mathbb{A}}}$ that satisfies Eq. (4.5.2). Therefore Eq. (4.5.1) has no solution with $p = \mathbb{0}$. But since Lemma 2.2 guarantees the existence of a solution there must be a solution of Eq. (4.5.1) with $p \neq \mathbb{0}$. □

So, if $p \neq \mathbb{0}$ we can conclude that the active constraint matrix keeps full row rank after the addition of constraint j to the working set. Otherwise, the components of vector q and the current dual vector $y_{\mathbb{A}}^{\text{opt}}$ can be used to determine a currently active constraint which must be removed before adding constraint j to the working set: In this case

$$\exists!\, q \in \mathbb{R}^{n_{\mathbb{A}}} :\; G'_j \;=\; \sum_{i \in \mathbb{A}} q_i G'_i \tag{4.5.3}$$

holds, where the uniqueness of q follows from the full row rank of $G_{\mathbb{A}}$, and it depends on the components of q how we proceed. If $q \leq \mathbb{0}$ all following QPs on the current homotopy path are infeasible as the boundary of the set $\mathcal{P}$ of admissible initial values is reached (this will be shown in Section 4.5.2). Instead, we assume that at least one component of q is positive. Then the following result is valid (taken from [11]):

Theorem 4.1 (ensuring linear independence of the active constraints): *Let $G_{\mathbb{A}}$ be the current active constraints matrix with full row rank and G'_j, $j \notin \mathbb{A}$, the constraint to be added to the working set $\mathbb{A}$. Moreover, assume that there exist a vector $q \in \mathbb{R}^{n_{\mathbb{A}}}$ as in Eq.* (4.5.3) *with at least one positive component and let $\big(\tilde{x}^{\text{opt}}(\tau_1), \tilde{y}^{\text{opt}}(\tau_1)\big)$ denote the optimal primal-dual solution pair at the current point $\tau_1 \in \mathbb{R}_{\geq 0}$ on the homotopy path.*

Then the matrix

$$G_{\mathbb{A}_{\text{new}}}, \quad \mathbb{A}_{\text{new}} \overset{\text{def}}{=} (\mathbb{A} \cup \{j\}) \setminus \{k\} \tag{4.5.4}$$

with

$$k \overset{\text{def}}{=} \underset{i \in \mathbb{A}}{\arg\min} \left\{ \frac{\tilde{y}_i^{\text{opt}}(\tau_1)}{q_i} \;\middle|\; q_i > 0 \right\} \tag{4.5.5}$$

also has full row rank. ○

Proof: Because $\left(\tilde{x}^{\text{opt}}(\tau_1), \tilde{y}^{\text{opt}}(\tau_1)\right)$ is a primal-dual optimal solution the KKT condition (4.1.5a) holds, i.e.

$$H\tilde{x}^{\text{opt}}(\tau_1) + \tilde{g}(\tau_1) \;=\; \sum_{i \in \mathbb{A}} G_i' \tilde{y}_i^{\text{opt}}(\tau_1)\,. \tag{4.5.6}$$

By multiplying Eq. (4.5.3) with an arbitrary $\lambda \in \mathbb{R}_{\geq 0}$ and subtracting the result from Eq. (4.5.6) we yield

$$H\tilde{x}^{\text{opt}}(\tau_1) + \tilde{g}(\tau_1) \;=\; \lambda G_j' + \sum_{i \in \mathbb{A}} G_i' \left(\tilde{y}_i^{\text{opt}}(\tau_1) - \lambda q_i\right). \tag{4.5.7}$$

Thus λ and the coefficients $\left(\tilde{y}_i^{\text{opt}}(\tau_1) - \lambda q_i\right)$ are also a valid dual solution vector which satisfies the KKT conditions (4.1.5) as long as all coefficients remain nonnegative. The largest value of λ for which this condition is satisfied is given by

$$\lambda_{\max} \overset{\text{def}}{=} \min_{i \in \mathbb{A}} \left\{ \frac{\tilde{y}_i^{\text{opt}}(\tau_1)}{q_i} \;\middle|\; q_i > 0 \right\} \in \mathbb{R}_{\geq 0}\,. \tag{4.5.8}$$

Note that this minimum is determined over a nonempty set according to our assumptions. Let k denote the constraint for which the minimum is attained, then $\tilde{y}_k^{\text{opt}}(\tau_1)$ is reduced to zero and constraint k can thus be removed from the working set. Since $q_k > 0$ the constraint vector G_j' is linearly independent from the G_i', $i \in \mathbb{A} \setminus \{k\}$, and therefore matrix $G_{\mathbb{A}_{\text{new}}}$, $\mathbb{A}_{\text{new}} \overset{\text{def}}{=} (\mathbb{A} \cup \{j\}) \setminus \{k\}$ has full row rank. □

This result provides a computationally convenient way for choosing a linearly independent subset of active constraints, if necessary. But it does not guarantee that this choice allows to make further progress along the homotopy path because it might be that constraint k immediately becomes active again. In order to prove that this cannot happen under certain conditions we need the following definition from [83]:

Definition 4.4 (ties): *The quadratic program* (4.1.3) *has*

- primal ties *at* $\tau_0 \in [0,1]$ *if* $\tau_{\max}^{\text{prim}} < \tau_{\max}^{\text{dual}}$ *and the minimum* (4.1.10a) *is obtained for at least two distinct indices;*
- dual ties *at* $\tau_0 \in [0,1]$ *if* $\tau_{\max}^{\text{dual}} < \tau_{\max}^{\text{prim}}$ *and the minimum* (4.1.10b) *is obtained for at least two distinct indices;*
- primal-dual ties *at* $\tau_0 \in [0,1]$ *if* $\tau_{\max}^{\text{dual}} = \tau_{\max}^{\text{prim}}$*;*
- ties *at* $\tau_0 \in [0,1]$ *if it has primal, dual or primal-dual ties.* ○

If ties occur there are different possibilities how to choose the new working set which poses additional difficulties. Otherwise the new working set is uniquely determined and we can prove the following theorem (taken from [11]):

Theorem 4.2: *If the assumptions of Theorem 4.1 hold and if no ties occur at τ_1, then constraint k remains inactive within an interval $(\tau_1, \tau_2]$, $\tau_2 > \tau_1$, on the homotopy path.* ○

Proof: The current linear line segment $\tilde{x}(\tau_0) + \tau\Delta x(\tau_0)$ of the primal optimal solution homotopy, starting at some $\tau_0 \in [0, \tau_1]$ and ending at τ_1, was chosen such that

$$G_{\mathbb{A}}\left(\tilde{x}(\tau_0) + \tau\Delta x(\tau_0)\right) = \tilde{b}_{\mathbb{A}}(\tau) \quad \forall\, \tau \in [0,1]\,, \tag{4.5.9a}$$

$$G'_j\left(\tilde{x}(\tau_0) + \tau\Delta x(\tau_0)\right) < \tilde{b}_j(\tau) \quad \forall\, \tau \in (\tau_1, 1] \tag{4.5.9b}$$

hold. Thus, by multiplying Eq. (4.5.3) with $\tilde{x}(\tau_0) + \tau\Delta x(\tau_0)$, one obtains the following equation

$$\sum_{i\in\mathbb{A}} q_i\tilde{b}_i(\tau) < \tilde{b}_j(\tau) \quad \forall\, \tau \in (\tau_1, 1]\,. \tag{4.5.10}$$

Within the next step of Algorithm 4.1, the new linear line segment of $\tilde{x}(\tau_1) + \tau\Delta x(\tau_1)$, starting at τ_1 and ending at some $\tau_2 \in [\tau_1, 1]$, is chosen such that

$$G'_i\left(\tilde{x}(\tau_1) + \tau\Delta x(\tau_1)\right) = \tilde{b}_i(\tau) \quad \forall\, i \in \mathbb{A}_{\text{new}} \stackrel{\text{def}}{=} (\mathbb{A} \cup \{j\}) \setminus \{k\} \tag{4.5.11}$$

holds in $[0, 1]$. By applying Eq. (4.5.3) again, we yield

$$\tilde{b}_j(\tau) = \sum_{i\in\mathbb{A}\setminus\{k\}} q_i\tilde{b}_i(\tau) + q_k G'_k\left(\tilde{x}(\tau_1) + \tau\Delta x(\tau_1)\right) \quad \forall\, \tau \in [0,1]\,. \tag{4.5.12}$$

Finally, by combining Eq. (4.5.10) and Eq. (4.5.12) we obtain

$$q_k\tilde{b}_k(\tau) < q_k G'_k\left(\tilde{x}(\tau_1) + \tau\Delta x(\tau_1)\right) \quad \forall\, \tau \in (\tau_1, 1] \tag{4.5.13}$$

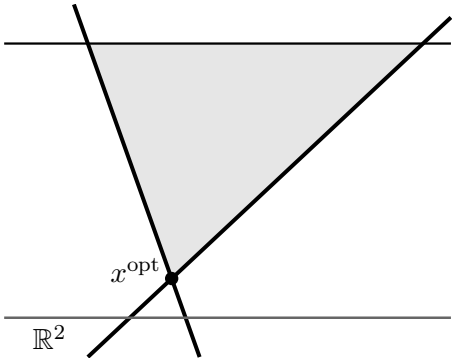
and, since $q_k > 0$, also

$$\tilde{b}_k(\tau) < G'_k\left(\tilde{x}(\tau_1) + \tau\Delta x(\tau_1)\right) \quad \forall\, \tau \in (\tau_1, 1] \tag{4.5.14}$$

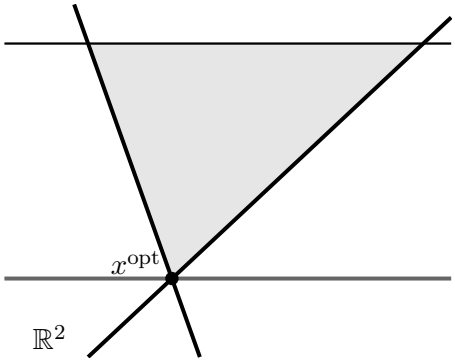
which proves that constraint k remains (strictly) inactive within the next step of Algorithm 4.1 from τ_1 to τ_2. If no ties occur at τ_1 only constraint j becomes active at τ_1 and $\tau_2 > \tau_1$ is valid. □

An approach for resolving ties is presented in [83]. Therein the solution of an auxiliary (non-parametric) quadratic program is proposed, which seems to be inadequate for the online context. Thus, our implementation does not cover the situation when ties are present—and no difficulties have been observed so far.

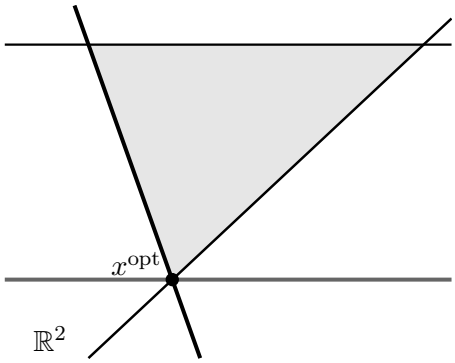
Figure 4.3 illustrates an example in which linear dependence of the active constraints occurs: the constraints are shifted while following the homotopy path (for simplicity, only one constraint is thought to be parameterised) which causes degeneracy at a certain homotopy parameter τ_1. Then Theorem 4.1 can be utilised in order to resolve this situation, i.e. to find an active constraint which can be removed from the working set. Afterwards, further progress along the homotopy path can be made.



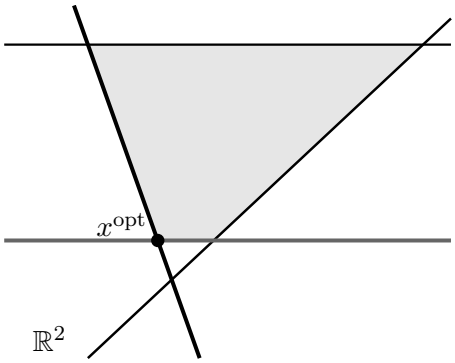
(a) Two active constraints.



(b) A third, parameterised constraint becomes active which is linearly dependent from the other ones.



(c) A formerly active constraint is removed from the working set (cf. Theorem 4.1).



(d) The parameterised (and active) constraint is shifted further.

Figure 4.3: Example of linear dependence of active constraints (bold) in parametric programming (dark-grey: parameterised constraint, grey: feasible set).

Implementation of Linear Dependence Handling

In order to ensure linear independence of the active constraints and to detect possible infeasibility, the modified KKT system (4.5.1) has to be solved. According to Eq. (4.3.10) the implemented variant of this KKT system reads

$$\begin{pmatrix} R'R & Z'_{\mathbb{F}}H_{\mathbb{F}}Y_{\mathbb{F}} & Z'_{\mathbb{F}}H_{\mathrm{M}} & \mathbb{0} & \mathbb{0} \\ Y'_{\mathbb{F}}H_{\mathbb{F}}Z_{\mathbb{F}} & Y'_{\mathbb{F}}H_{\mathbb{F}}Y_{\mathbb{F}} & Y'_{\mathbb{F}}H_{\mathrm{M}} & \mathbb{0} & T' \\ H'_{\mathrm{M}}Z_{\mathbb{F}} & H'_{\mathrm{M}}Y_{\mathbb{F}} & H_{\mathbb{X}} & \mathrm{Id}_{n_{\mathbb{X}}} & C'_{\mathbb{X}} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_{n_{\mathbb{X}}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & T & C_{\mathbb{X}} & \mathbb{0} & \mathbb{0} \end{pmatrix} \begin{pmatrix} Z'_{\mathbb{F}}p_{\mathbb{F}} \\ Y'_{\mathbb{F}}p_{\mathbb{F}} \\ p_{\mathbb{X}} \\ q_{\mathbb{X}} \\ q_{\mathbb{A}} \end{pmatrix} = \begin{pmatrix} Z'_{\mathbb{F}}(G'_j)_{\mathbb{F}} \\ Y'_{\mathbb{F}}(G'_j)_{\mathbb{F}} \\ (G'_j)_{\mathbb{X}} \\ \mathbb{0} \\ \mathbb{0} \end{pmatrix}, \quad (4.5.15)$$

where $p \in \mathbb{R}^n$, $q \in \mathbb{R}^{n_\mathbb{A}+n_\mathbb{X}}$ and $G'_j \in \mathbb{R}^n$ were split into two parts corresponding to the free and fixed variables or the active constraints, respectively. Because of its special structure the computational effort for its solution is much lower than a normal primal-dual step determination: first, we can exploit the equivalence

$$\exists! \, q \in \mathbb{R}^{n_\mathbb{A}} : \; (G'_j)_\mathbb{F} = \sum_{i \in \mathbb{A}} q_i (C_\mathbb{F})'_i \iff Z'_\mathbb{F}(G'_j)_\mathbb{F} = \mathbb{0} \tag{4.5.16}$$

which holds since the constraint to be added to the working set is linearly dependent with the active constraints if and only if it lies completely in the range space of $C_\mathbb{F}$ and is thus orthogonal to all basis vectors of the null space of $C_\mathbb{F}$. So, if $Z'_\mathbb{F}(G'_j)_\mathbb{F} \neq \mathbb{0}$ we can stop the calculation as no linear dependence occurs. Otherwise, we proceed where the information $Z'_\mathbb{F}(G'_j)_\mathbb{F} = \mathbb{0}$ further simplifies the solution. Since $Z'_\mathbb{F} p_\mathbb{F}$, $Y'_\mathbb{F} p_\mathbb{F}$ and $p_\mathbb{X}$ become zero in this case, we finally end up with the following formulae for q:

$$q_\mathbb{A} = (T')^{-1} Y'_\mathbb{F} (G'_j)_\mathbb{F} \,, \tag{4.5.17a}$$

$$q_\mathbb{X} = (G'_j)_\mathbb{X} - C'_\mathbb{X} q_\mathbb{A} \,. \tag{4.5.17b}$$

Compared to the calculation of the primal-dual step direction via Eqs. (4.3.11), the cost for a linear dependence check is almost negligible. Especially if a bound is added to the working set because then $(G'_j)_\mathbb{F}$ equals a unity vector and $(G'_j)_\mathbb{X}$ is zero. However, note that in the case of degeneracy further computations are necessary in order to perform the additional change of the working set.

4.5.2 Infeasibility

The proposed online active set strategy produces a sequence of iterates which are primal and dual feasible for consecutive (intermediate) quadratic programs. Thus, infeasibility can only occur if a bound or constraint is added while following the homotopy path. In this case the augmented active constraints matrix has to be prevented from becoming rank deficient anyway and we mentioned in Section 4.5.1 that possible infeasibility can be detected simultaneously, as follows.
Recall the situation[4] when a constraint $j \notin \mathbb{A}$ shall be added to the working set $\mathbb{A}$. If $G_\mathbb{A}$ has full row rank, linear dependence of G'_j and the columns of $G_\mathbb{A}$ is equivalent to

$$\exists! \, q \in \mathbb{R}^{n_\mathbb{A}} : \; G'_j = \sum_{i \in \mathbb{A}} q_i G'_i \,. \tag{4.5.18}$$

Theorem 4.1 shows that we can resolve linear dependence if the vector q in Eq. (4.5.18) has at least one positive component. If this in not the case infeasibility is encountered (cf. [11]):

Theorem 4.3 (infeasibility detection): *Let $G_\mathbb{A}$ be the current active constraints matrix with full row rank and G'_j, $j \notin \mathbb{A}$, the constraint to be added to the working set $\mathbb{A}$. Assume that there exists a vector $q \in \mathbb{R}^{n_\mathbb{A}}$ as in Eq.* (4.5.18) *which has no positive component. Moreover, let $\big(\tilde{x}^{\mathrm{opt}}(\tau_1), \tilde{y}^{\mathrm{opt}}(\tau_1)\big)$ denote the optimal primal-dual solution pair at the current point $\tau_1 \in \mathbb{R}_{\geq 0}$ on the homotopy path and assume that no ties occur at τ_1.*
Then all parametric quadratic programs on the homotopy path with $\tau > \tau_1$ are infeasible. ○

[4] Again, for clarity, we restrict the presentation to the case where bounds and constraints are not distinguished.

Proof: Suppose that for some $\tau > \tau_1$ an arbitrary vector $x \in \mathbb{R}^n$ satisfies the constraints

$$G_i' x \geq \tilde{b}_i(\tau) \quad \forall\, i \in \mathbb{A}\,. \tag{4.5.19}$$

Multiplying each such inequality by $q_i \leq 0$, adding them together and using Eq. (4.5.18) leads to

$$G_j' x \;\leq\; \sum_{i \in \mathbb{A}} q_i \tilde{b}_i(\tau)\,. \tag{4.5.20}$$

But on the other hand, as in the proof of Theorem 4.2 (cp. Eq. (4.5.10)), we can derive

$$\sum_{i \in \mathbb{A}} q_i \tilde{b}_i(\tau) \;<\; \tilde{b}_j(\tau) \quad \forall\, \tau \in (\tau_1, 1] \tag{4.5.21}$$

which implies

$$G_j' x \;<\; \tilde{b}_j(\tau) \quad \forall\, \tau \in (\tau_1, 1]\,. \tag{4.5.22}$$

Since x was arbitrary, constraint j will be violated for all $\tau > \tau_1$ as long as all constraints indexed by $\mathbb{A}$ remain fulfilled. Therefore, there exists no point satisfying all constraints indexed by $\mathbb{A} \cup \{j\}$ no matter how the primal step direction is chosen. Since Theorem 2.8 guarantees the existence of a continuous continuation of $x^{\mathrm{opt}}(\tau)$ all QPs on the homotopy path are infeasible for $(\tau_1, \tau_1 + \varepsilon)$ and some $\varepsilon > 0$. Finally, the convexity of $\mathcal{P}$ (cp. Theorem 2.6) proves that all QPs on the homotopy path are infeasible for all $\tau > \tau_1$. □

If the situation of Theorem 4.3 occurs, the boundary of the set of feasible parameters $\mathcal{P}$ is reached and we know that the current QP is infeasible:

Theorem 4.4 (infeasibility of the current QP): *Let $\mathrm{QP}(w_0)$ be the feasible, recently solved quadratic program and $\mathrm{QP}(w_0^{\mathrm{new}})$ the one to be solved next (both strictly convex and no ties occur along the homotopy path between them). Then $\mathrm{QP}(w_0^{\mathrm{new}})$ is* infeasible *if and only if there exists a $\tau_1 \in [0, 1)$—along the homotopy from $\mathrm{QP}(w_0)$ to $\mathrm{QP}(w_0^{\mathrm{new}})$—to which Theorem 4.3 applies.* ○

Proof: All primal-dual pairs $\big(x^{\mathrm{opt}}(\tau), y^{\mathrm{opt}}(\tau)\big)$, $\tau \in [0, 1]$, along the homotopy path are optimal and $x^{\mathrm{opt}}(\tau)$ primal feasible, hence. If there is no $\tau_1 < 1$ to which Theorem 4.3 applies it is possible to follow the homotopy until the optimal solution of $\mathrm{QP}(w_0^{\mathrm{new}})$, implying its feasibility.

The converse direction follows directly from Theorem 4.3 as $\mathrm{QP}(w_0^{\mathrm{new}})$ denotes the QP on the homotopy path at $\tau = 1$. □

If infeasibility of the current quadratic program to be solved is detected via Theorem 4.4 our implementation of the online active set strategy just stops the homotopy and waits for the next QP which may be feasible again. In doing so, convexity of $\mathcal{P}$ ensures that a homotopy from the currently solved intermediate QP to the new one exists (see Figure 4.4).

Provided that the MPC problem is well-posed, infeasibility should be a rare exception and mainly due to *measurement errors* of the current process state w_0^{new}. One interpretation of our infeasibility strategy is that it "trusts" the current process state as long as the resulting QP remains feasible and uses a linear interpolation between w_0^{new} and the old process state w_0 otherwise. This strategy seems adequate for practical setups, where uncertainties are inherently present, even if more elaborated schemes may be conceivable.

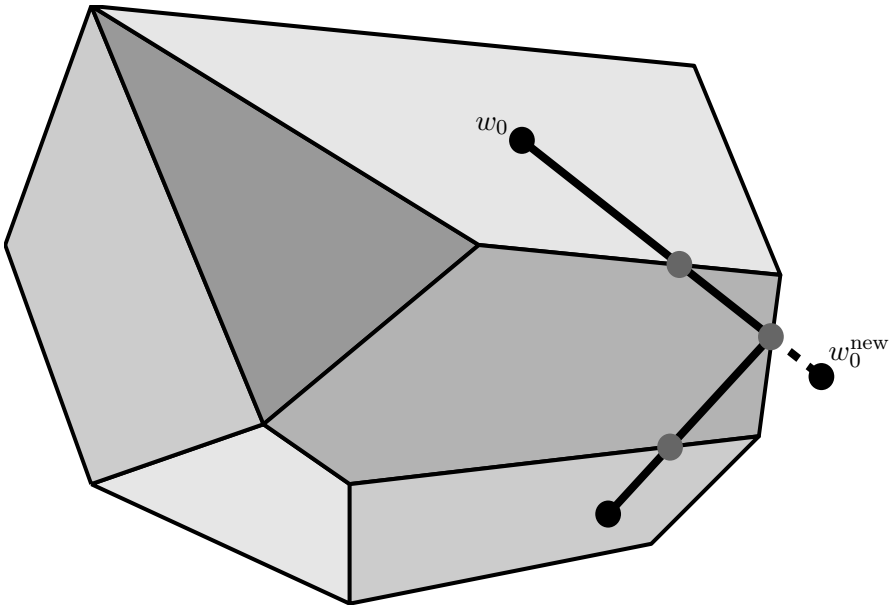


Figure 4.4: Infeasibility handling of the proposed online active set strategy.

4.6 Computational Complexity

4.6.1 Runtime Complexity

We already know from Section 3.1 that the effort for one iteration of a primal active set method is $\mathcal{O}(n^2)$ if *matrix updates* are used. In this section we want to investigate the runtime complexity of the proposed online active set strategy in more detail. Since theoretical values for the number of required iterations for finding the solution are not available—only an (almost trivial) exponential worst-case bound is known—we restrict the presentation to the complexity of one single iteration.

Algorithm 4.1 starts with calculating the vectors Δw_0, Δg and Δb via Eqs. (4.1.1) which obviously requires $\mathcal{O}(n)$ floating-point operations[5].
Afterwards, the primal-dual step directions Δx^{opt} and Δy^{opt} have to be determined. This is done by using Eqs. (4.3.11) while exploiting common subexpressions therein. Clearly, as $n_{\mathbb{F}}, n_{\mathbb{X}} \leq n$ and $n_{\mathbb{A}}, n_{\mathbb{I}} \leq n$, this calculation requires $\mathcal{O}(n^2)$ floating-points operations; the exact value is given in Table 4.1.
Third, the maximum homotopy step length $\tau_{\max}$ has to be obtained from Eqs. (4.1.10). This makes the calculation of the matrix-vector product $G'_{\mathbb{I}} \Delta x^{\text{opt}}$ and therefore $n n_{\mathbb{I}}$ floating point operations necessary[6]; besides some negligible $\mathcal{O}(n)$ operations.

[5] Within this section a floating-point operation is defined as one multiplication/division *together* with an addition. Thus, calculating the dot product $a'b$ of two vectors a, $b \in \mathbb{R}^n$ requires n floating-point operations, for example.

[6] In the very first iteration also $G'_{\mathbb{I}} x^{\text{opt}}$ has to be calculated, which is zero if the initialisation homotopy is used.

Steps (4), (5) and (7) only involve a fixed number of some vector operations and thus have $\mathcal{O}(n)$ complexity.
Finally, step (6) involves one of the four possible matrix updates (i.e. adding/removing of a bound/constraint to/from the working set). Their computational effort can easily be derived from their detailed description in Section 4.3.3 (see also [39]) and is summarised in Table 4.1, assuming that a Givens plane rotation can be performed by means of three floating-point operations (cf. page 49). Note that also the effort for calculating the product $Z'_{\mathbb{F}}(G'_j)_{\mathbb{F}}$, as decribed in Section 4.5.1, is included into the complexity of adding a bound/constraint. If this product equals zero linear independence must be ensured: via Eqs. (4.5.17) and some $\mathcal{O}(n)$ operations a bound or a constraint is determined which has to be removed from the working set.

Table 4.1: Runtime complexity of the online active set strategy (general case).

Task:	Complexity:
Determination of step direction	$5n^2 - 2nn_{\mathbb{A}} - 8nn_{\mathbb{X}} + 2n_{\mathbb{A}}^2 + 4n_{\mathbb{A}}n_{\mathbb{X}} + 4n_{\mathbb{X}}^2 + \mathcal{O}(n)$
Determination of step length	$nn_{\mathbb{I}} + \mathcal{O}(n)$
Removing a bound from working set	$\frac{5}{2}n^2 + nn_{\mathbb{A}} - 5nn_{\mathbb{X}} + 2n_{\mathbb{A}}^2 - n_{\mathbb{A}}n_{\mathbb{X}} + \frac{5}{2}n_{\mathbb{X}}^2 + \mathcal{O}(n)$
Removing a constraint from working set[7]	$\frac{5}{2}n^2 - \frac{1}{2}nn_{\mathbb{A}} - 5nn_{\mathbb{X}} + \frac{7}{8}n_{\mathbb{A}}^2 + \frac{1}{2}n_{\mathbb{A}}n_{\mathbb{X}} + \frac{5}{2}n_{\mathbb{X}}^2 + \mathcal{O}(n)$
Adding a bound to working set	$5n^2 - 4nn_{\mathbb{A}} - 10nn_{\mathbb{X}} + \frac{3}{2}n_{\mathbb{A}}^2 + 4n_{\mathbb{A}}n_{\mathbb{X}} + 5n_{\mathbb{X}}^2 + \mathcal{O}(n)$
Adding a constraint to working set	$5n^2 - 4nn_{\mathbb{A}} - 10nn_{\mathbb{X}} + 4n_{\mathbb{A}}n_{\mathbb{X}} + 5n_{\mathbb{X}}^2 + \mathcal{O}(n)$
Ensuring linear independence	$nn_{\mathbb{A}} + \frac{1}{2}n_{\mathbb{A}}^2 + \mathcal{O}(n)$
Remaining calculations	$\mathcal{O}(n)$

As summarised in Table 4.1, the computational effort of all steps of the online active set strategy depends not only on the number of variables but also on how many variables are fixed ($n_{\mathbb{F}}$) and how many constraints are active ($n_{\mathbb{A}}$). One complete iteration consists of determination of the step direction, determination of the step length, one change of the working set and the remaining calculations. In order to simplify the analysis, we define the *average* effort for one working set change as

$$\begin{aligned} &\frac{n_{\mathbb{X}}}{2n} \cdot \text{“removing a bound”} + \frac{n_{\mathbb{A}}}{2n} \cdot \text{“removing a constraint”} \\ &\frac{n - n_{\mathbb{X}}}{2n} \cdot \text{“adding a bound”} + \frac{n - n_{\mathbb{A}}}{2n} \cdot \text{“adding a constraint”} , \end{aligned} \tag{4.6.1}$$

since it seems reasonable to assume that it is more likely that a bound is to be removed from the working set if more variables are fixed and so on.
Furthermore, we can consider the case when linear independence occurs. Then also linear independence has to be ensured by removing a bound or a constraint from the working set. The *average* effort for performing this additional working set change is chosen as

$$\frac{n_{\mathbb{X}}}{n_{\mathbb{X}} + n_{\mathbb{A}}} \cdot \text{“removing a bound”} + \frac{n_{\mathbb{A}}}{n_{\mathbb{X}} + n_{\mathbb{A}}} \cdot \text{“removing a constraint”} , \tag{4.6.2}$$

provided that $n_{\mathbb{X}} + n_{\mathbb{A}} > 0$.

As a last simplification, we assume that the number of constraints equals the number of variables, i.e. $m = n$, and express both the number of fixed variables and the number of active constraints as a fraction of some arbitrary but fixed $n \in \mathbb{N}$:

$$n_{\mathbb{X}} \overset{\text{def}}{=} n\alpha_{\mathbb{X}}\,, \quad \alpha_{\mathbb{X}} \in [0,1]\,, \tag{4.6.3a}$$

$$n_{\mathbb{A}} \overset{\text{def}}{=} n\alpha_{\mathbb{A}}\,, \quad \alpha_{\mathbb{A}} \in [0, 1-\alpha_{\mathbb{X}}]\,. \tag{4.6.3b}$$

Table 4.2 shows the runtime complexity of the online active set strategy for different values of $\alpha_{\mathbb{X}}$ and $\alpha_{\mathbb{A}}$. Figure 4.6.1 illustrates the runtime complexity of one complete iteration (no linear dependence occurs) of the online active set strategy with respect to the number of fixed variables and active constraints as defined in Eqs. (4.6.3).
We can see that the most computational effort per iteration is needed if no variables are fixed and no constraints are active, which normally is the case if the system to be controlled is near a steady-state. If the number of fixed variables or active constraints increases the runtime complexity decreases significantly. This effect is particularly striking if the number of free variables becomes small which also justifies the distinction between bounds and constraints.
Another expected observation is that computational effort increases if linear dependence occurs. Therefore, it is reasonable to take the effort of one complete iteration in which linear dependece occurs and no variables are fixed and no constraints are active, i.e. $13.5n^2+\mathcal{O}(n)$ floating-point operations, as an *uppper bound* for the computational burden of one iteration (even if linear dependence cannot occur in this situation). Although this bound need not to be strict because of the averaging process of Eqs. (4.6.1) and (4.6.2) and the assumption $m = n$, it should be a sufficiently accurate guess for practical purposes if n is "large". Especially if $m \gg n$ one can construct situations where the computational effort might be higher, but it is important to note that the effort per iteration grows *quadratically* in the number of variables as long as $m \in \mathcal{O}(n)$.

Table 4.2: Runtime complexity of the online active set strategy modulo $\mathcal{O}(n)$ for several special cases.

Task:	Complexity:					
	$n_{\mathbb{X}} = 0$, $n_{\mathbb{A}} = 0$	$n_{\mathbb{X}} = \frac{n}{3}$, $n_{\mathbb{A}} = 0$	$n_{\mathbb{X}} = 0$, $n_{\mathbb{A}} = \frac{n}{3}$	$n_{\mathbb{X}} = \frac{n}{3}$, $n_{\mathbb{A}} = \frac{n}{3}$	$n_{\mathbb{X}} = n$, $n_{\mathbb{A}} = 0$	$n_{\mathbb{X}} = 0$ $n_{\mathbb{A}} = n$
Determination of step direction	$5.0n^2$	$2.8n^2$	$4.6n^2$	$2.8n^2$	$1.0n^2$	$5.0n^2$
Determination of step length	$1.0n^2$	$1.0n^2$	$0.7n^2$	$0.7n^2$	$1.0n^2$	$0.0n^2$
Removing a bound	$2.5n^2$	$1.1n^2$	$3.1n^2$	$1.6n^2$	$0.0n^2$	$5.5n^2$
Removing a constraint[7]	$2.5n^2$	$1.1n^2$	$2.4n^2$	$1.1n^2$	$0.0n^2$	$2.9n^2$
Adding a bound	$5.0n^2$	$2.2n^2$	$3.8n^2$	$1.5n^2$	$0.0n^2$	$2.5n^2$
Adding a constraint	$5.0n^2$	$2.2n^2$	$3.7n^2$	$1.3n^2$	$0.0n^2$	$1.0n^2$
Ensuring linear independence	$0.0n^2$	$0.0n^2$	$0.4n^2$	$0.4n^2$	$0.0n^2$	$1.5n^2$
One complete iteration (no linear dependence occurs)	$11.0n^2$	$5.8n^2$	$8.8n^2$	$4.8n^2$	$2.0n^2$	$7.7n^2$
One complete iteration (linear dependence occurs)	$[13.5n^2]$	$7.1n^2$	$11.8n^2$	$6.6n^2$	$2.0n^2$	$11.9n^2$

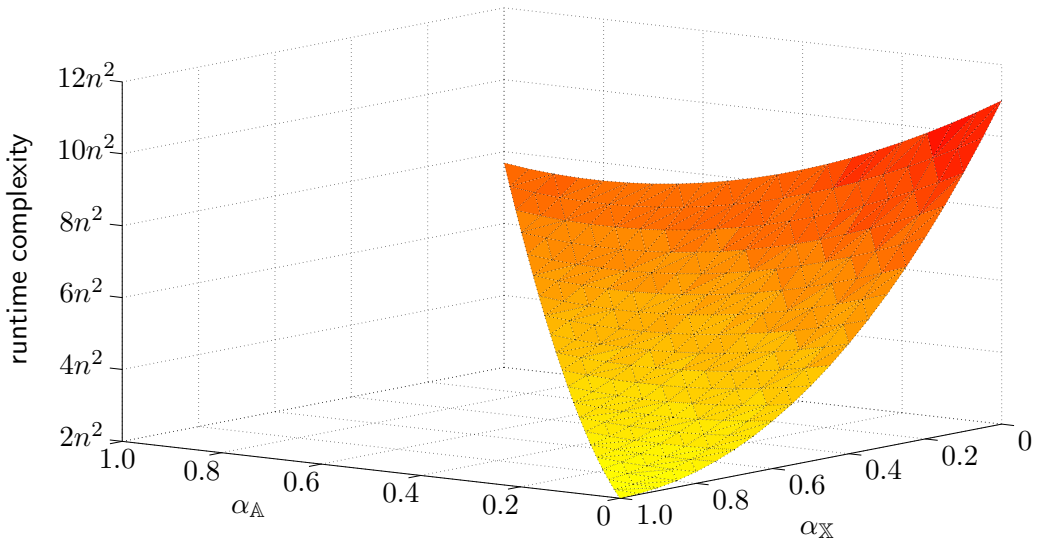


Figure 4.5: Runtime complexity of one complete iteration (no linear dependence occurs) of the online active set strategy with respect to the number of fixed variables and active constraints.

Refinements for Determing the Step Direction

As mentioned in Section 4.3.2, the computational effort for calculating the primal-dual step direction can be reduced if the currently active bounds $b_{\mathbb{X}}$ or constraints' bounds $b_{\mathbb{A}}$ (see Eq. (4.3.2)) are independent from w_0. We omit the resulting equivalents to Eqs. 4.3.11 and just summarise their runtime complexities in Table 4.3. If both active bounds and constraints do not depend on w_0 savings between 20 % and 100 % are theoretically possible (compared with the standard approach for determing the step direction).

Table 4.3: Runtime complexity for calculating the primal-dual step direction of the online active set strategy.

Task:	Complexity:
Determination of step direction	$5n^2 - 2nn_{\mathbb{A}} - 8nn_{\mathbb{X}} + 2n_{\mathbb{A}}^2 + 4n_{\mathbb{A}}n_{\mathbb{X}} + 4n_{\mathbb{X}}^2 + \mathcal{O}(n)$
Determination of step direction (bounds independent)	$5n^2 - 2nn_{\mathbb{A}} - 9nn_{\mathbb{X}} + 2n_{\mathbb{A}}^2 + 3n_{\mathbb{A}}n_{\mathbb{X}} + 4n_{\mathbb{X}}^2 + \mathcal{O}(n)$
Determination of step direction (bounds and constraints independent)	$4n^2 - 3nn_{\mathbb{A}} - 7nn_{\mathbb{X}} + \frac{3}{2}n_{\mathbb{A}}^2 + 4n_{\mathbb{A}}n_{\mathbb{X}} + 3n_{\mathbb{X}}^2 + \mathcal{O}(n)$

[7] The computational effort depends on which constraint is removed. For simplicity, it is assumed that the $\frac{n_{\mathbb{A}}}{2}$th row is removed from $G_{\mathbb{A}}$.

4.6.2 Memory Requirements

The proposed online active set strategy was implemented under the assumption that all matrices are *dense*, i.e. that most entries are non-zero. This is justified if the matrices of the open-loop optimal control problem are dense or a long prediction horizon $n_\mathrm{p} \gg 1$ is used (leading to dense entries $A^j B$, $0 \leq j \leq n_\mathrm{p} - 1$, in Eq. (2.2.20c)). Thus, all matrices H, A, T, Q and R are stored completely in two-dimensional arrays. For each matrix the maximal possibly required memory is allocated and, for simplicity, no advantage of the symmetry of H and the triangular shape of T and R is taken. Table 4.4 lists all memory requirements of our implementation of the online active set strategy and shows that the storage complexity is $\mathcal{O}(n^2)$, provided that the number of constraints grows linearly in the number of variables.

Table 4.4: Memory requirements of our implementation of the online active set strategy.

Data:	H	A	T	Q	R	others	total
Memory:	n^2	nm	n^2	n^2	n^2	$\mathcal{O}(n)$	$4n^2 + nm + \mathcal{O}(n)$

4.7 Further Refinements and Extensions

In this section we use formulation (2.3.15) instead of (4.3.1) for notational convenience.

4.7.1 Step Length Determination

Most of the runtime for determing the primal-dual step length is spent for calculating the maximal *primal step length* via Eq. (4.1.10a). This calculation even takes a significant part of the whole computational effort for one iteration if the number of constraints becomes large (compared with the number of optimisation variables). Therefore, we present an idea of how the determination of the maximal primal step length can be simplified.

We assume without loss of generality that every (nontrivial) constraint has Euclidean length one, which can easily be achieved by normalising every constraint, i.e.

$$G_i' x \;\geq\; \tilde{b}_i(\tau) \quad\Longleftrightarrow\quad \frac{G_i'}{\|G_i'\|_2} x \;\geq\; \frac{\tilde{b}_i(\tau)}{\|G_i'\|_2} \quad \forall\, i \in \{1, \ldots, m\}\,. \tag{4.7.1}$$

At every primal solution along the homotopy path, $\tau \in [0, 1]$, and for every constraint we define a *feasibility measure*:

$$\varepsilon_i(\tau) \;\stackrel{\mathrm{def}}{=}\; G_i' x^{\mathrm{opt}}(\tau) - \tilde{b}_i(\tau) \;\geq\; 0 \quad \forall\, i \in \{1, \ldots, m\}\,. \tag{4.7.2}$$

Then the following holds:

Lemma 4.3 (feasibility measure): *Let a normalised constraint $G_i' x \geq \tilde{b}_i(\tau)$, $1 \leq i \leq m$, with corresponding feasibility measure as defined in Eq. (4.7.2) be given. Let this constraint be inactive at some fixed $\tau_1 \in [0, 1]$ along the homotopy path, i.e. $\varepsilon_i(\tau_1) > 0$, and $\|\Delta x(\tau_1)\|_2 + |\Delta b| < \varepsilon_i(\tau_1)$. Then the constraint remains inactive for all $\tau \in [\tau_1, 1]$.* ○

Proof: The triangle and Cauchy-Schwarz's inequality imply:

$$\begin{aligned}
& G_i' \left(x^{\mathrm{opt}}(\tau_1) + \tau \Delta x(\tau_1)\right) - \left(b_i(0) + \tau \Delta b_i\right) \\
= \; & \varepsilon_i(\tau_1) + \tau \left(G_i' \Delta x(\tau_1) - \Delta b_i\right) \\
\geq \; & \varepsilon_i(\tau_1) - \tau \left|G_i' \Delta x(\tau_1) - \Delta b_i\right| \\
\geq \; & \varepsilon_i(\tau_1) - \tau \left(\left|G_i' \Delta x(\tau_1)\right| + |\Delta b_i|\right) \\
\geq \; & \varepsilon_i(\tau_1) - \tau \left(\left\|G_i'\right\|_2 \left\|\Delta x(\tau_1)\right\|_2 + |\Delta b_i|\right) \\
= \; & \varepsilon_i(\tau_1) - \tau \left(\left\|\Delta x(\tau_1)\right\|_2 + |\Delta b_i|\right) \\
\geq \; & \varepsilon_i(\tau_1) - \left(\left\|\Delta x(\tau_1)\right\|_2 + |\Delta b_i|\right) \\
> \; & \varepsilon_i(\tau_1) - \varepsilon_i(\tau_1) \; = \; 0 \,,
\end{aligned}$$

which shows that the constraint remains inactive for all $\tau \in [\tau_1, 1]$. □

This lemma shows that an inactive constraint whose feasibility measure is greater than the Euclidean norm of the current primal step direction plus the absolute value of the constraint vector step direction cannot become a blocking constraint. Hence, storing the feasibility measure of the inactive constraints may partly avoid the calculation of the product $G_{\mathbb{I}}' \Delta x(\tau_1)$ in Algorithm 4.1. Since calculating the feasibility measures $\varepsilon_i(\tau_1)$, $i \in \mathbb{I}(\tau_1)$, exactly after each homotopy step would outweigh the possible benefit, only cheaply available *lower bounds* $\underline{\varepsilon}_i \leq \varepsilon_i(\tau) \; \forall \, \tau \in [0, 1]$ are held:

(1) for $\tau_1 = 0$ define

$$\underline{\varepsilon}_i \stackrel{\mathrm{def}}{=} \varepsilon_i(0) \quad \forall \, i \in \mathbb{I}(0) \,, \tag{4.7.3}$$

(2) when determing the maximum primal homotopy step length $\tau_{\max}$ consider only inactive constraints $1 \leq i \leq m$ with

$$\|\Delta x(\tau_1)\|_2 + |\Delta b_i| \; \geq \; \underline{\varepsilon}_i \,, \tag{4.7.4}$$

(3) afterwards update $\underline{\varepsilon}_i$ as follows:

$$\underline{\varepsilon}_i \stackrel{\mathrm{def}}{=} \begin{cases} \varepsilon_i(\tau_1 + \tau_{\max}) & \|\Delta x(\tau_1)\|_2 + |\Delta b_i| \geq \underline{\varepsilon}_i \,, \\ \underline{\varepsilon}_i - (\tau_1 + \tau_{\max}) \left(\left\|\Delta x(\tau_1)\right\|_2 + |\Delta b_i|\right) & \text{else} \,. \end{cases} \tag{4.7.5a}$$

Steps (2) and (3) are repeated until the solution of current QP ($\tau_1 = 1$) is found, and also afterwards for solving the following QPs. Note that step (3) requires only $\mathcal{O}(m)$ additional floating-point operations as all necessary quantities are already calculated in the second step. Therefore considerable computational savings can be expected if the quadratic program comprises many constraints that are "far" from becoming active. In our first test example (see Chapter 5) we observed computational savings up to 10 %.

4.7.2 Extension to Sequential Quadratic Programming

Now we briefly present a possibility to extend the proposed online active set strategy to *nonlinear MPC*. As mentioned in Section 2.1, in this case a nonlinear program (NLP) instead

of a QP has to be solved. This can be done efficiently via *sequential quadratic programming (SQP) methods* (see, e.g., [65] for a detailed description). Therein a sequence of QPs is solved at each sampling instant which differ not only in the gradient and the constraint vector, but also in the (positive definite) Hessian matrix (approximation) and the constraint matrix.

Let us assume that we have solved one of these QPs:

$$\mathrm{QP}: \qquad \min_{x\in\mathbb{R}^n} \quad \tfrac{1}{2}x'Hx + x'g \tag{4.7.6a}$$

$$\text{s.t.} \quad Gx \;\geq\; b\,, \tag{4.7.6b}$$

with optimal primal-dual solution pair $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ and correponding optimal working set $\mathbb{A}$ and now want to solve the next one:

$$\mathrm{QP}^{\mathrm{new}}: \qquad \min_{x\in\mathbb{R}^n} \quad \tfrac{1}{2}x'H^{\mathrm{new}}x + x'g^{\mathrm{new}} \tag{4.7.7a}$$

$$\text{s.t.} \quad G^{\mathrm{new}}x \;\geq\; b^{\mathrm{new}}\,. \tag{4.7.7b}$$

By subtracting the KKT optimality conditions (2.3.13) of both QPs it is easy to see that $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$, together with the *same optimal working set* $\mathbb{A}$[8], is also the optimal solution of the transformed QP:

$$\overrightarrow{\mathrm{QP}}: \qquad \min_{x\in\mathbb{R}^n} \quad \tfrac{1}{2}x'H^{\mathrm{new}}x + x'\vec{g} \tag{4.7.8a}$$

$$\text{s.t.} \quad G^{\mathrm{new}}x \;\geq\; \vec{b}, \tag{4.7.8b}$$

with

$$\vec{g} \;\stackrel{\mathrm{def}}{=}\; g - \left(H^{\mathrm{new}} - H\right)x^{\mathrm{opt}} + \left(G^{\mathrm{new}} - G\right)y^{\mathrm{opt}}\,, \tag{4.7.9a}$$

$$\vec{b} \;\stackrel{\mathrm{def}}{=}\; b + \left(G^{\mathrm{new}} - G\right)x^{\mathrm{opt}}\,. \tag{4.7.9b}$$

Thus, it is possible to start from the optimal solution $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ of $\overrightarrow{\mathrm{QP}}$ and start a homotopy towards the solution of $\mathrm{QP}^{\mathrm{new}}$. In doing so the following steps have to be performed:

1. Calculate matrix factorisations of new Hessian matrix H^{new} and new constraint matrix G^{new} for optimal working set $\mathbb{A}$;
2. Calculate transformed gradient vector $\vec{g}$ and transformed constraint vector $\vec{b}$ via Eqs. (4.7.9);
3. Peform a homotopy from $\overrightarrow{\mathrm{QP}}$ to $\mathrm{QP}^{\mathrm{new}}$ (i.e. from $\vec{g}$ to g^{new} and from $\vec{b}$ to b^{new}, respectively) starting from the last optimal solution $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$.

This approach makes it possible to warm start also the QPs within a SQP algorithm and even allows to interrupt solving the optimal control problem *during* one SQP iteration. Implementing this extension of our online active set strategy will be an issue for future work.

[8] Provided that $G^{\mathrm{new}}_{\mathbb{A}}$ has full row rank.

Chapter 5

Numerical Tests: Chain of Spring Connected Masses

Now we want to analyse the performance of the proposed online active set strategy by solving two different problems: the first one is a challenging benchmark problem—comprising 240 variables and 1191 bounds/constraints—where a chain of spring connected masses is regulated back into its steady-state after a strong excitation. Second, see Chapter 6, we aim at controlling a *real-world Diesel engine* available for experiments at the Institute for Design and Control of Mechatronical Systems in Linz, Austria. The results are also compared with those of a standard active set QP solver and the explicit (offline) approach.

5.1 Model Description and Problem Formulation

Our first test example is a variant of a recently published benchmark problem [86], [87]. Since it was deeply analysed in [86] we outline only its main characteristics.

We consider a chain consisting of nine balls which are connected by eight Hookian springs in between and two further Hookian springs at each end. Each ball i, $1 \leq i \leq 9$, is thought to be concentrated in a single point $x^i \in \mathbb{R}^3$ with mass $m \in \mathbb{R}_{>0}$ (in kg). All springs are identical having spring constant $d \in \mathbb{R}_{>0}$ (in $\mathrm{N/m}$) and rest length $L \in \mathbb{R}_{>0}$ (in m). One end of the chain is fixed at a certain point $x_0 \in \mathbb{R}^3$, whereas the free end of the spring at the other end of the chain is freely movable (its position is denoted by $x^{10} \in \mathbb{R}^3$). The whole chain of spring connected masses is situated in a homogeneous gravitational field decribed by its acceleration vector $g \in \mathbb{R}^3$ (in $\mathrm{m/s^2}$).

Without loss of generality, we let $x^0 \stackrel{\text{def}}{=} \mathbb{0}$ and obtain for all times $t \in \mathbb{T} \stackrel{\text{def}}{=} [0, \infty)$ the following (second-order) ODE system from Newton's laws of motion:

$$\ddot{x}^i(t) \;=\; \frac{F^{i,i+1}(t) - F^{i-1,i}(t)}{m} + g \qquad \forall\, i \in \{1, \ldots, 9\}\,, \tag{5.1.1a}$$

$$\text{where} \quad F^{i,i+1}(t) \;\stackrel{\text{def}}{=}\; d\left(1 - \frac{L}{\|x^{i+1}(t) - x^i(t)\|_2}\right)\left(x^{i+1}(t) - x^i(t)\right) \tag{5.1.1b}$$

denotes the force acting on the ith mass due to the spring between the ith and the $(i+1)$th mass (pointing from x^i to x^{i+1}). Via standard techniques, this system can be reformulated

as a first-order, i.e. involving only first time derivatives, ODE system by introducing the velocity vectors $\dot{x}^i(t) \in \mathbb{R}^3$, $1 \leq i \leq 9$, of the masses as additional differential variables. The chain is controlled by manipulating the three velocity components of the free end at point x^{10}, leading to three additional differential equations

$$\dot{x}^{10}(t) \;=\; u(t)\,, \tag{5.1.1c}$$

where $u : \mathbb{T} \rightarrow \mathbb{R}^3$ denotes the process inputs as described in Section 2.1. By defining

$$x(t) \stackrel{\text{def}}{=} \left(x^1(t)', \dot{x}^1(t)', \ldots, x^9(t)', \dot{x}^9(t)', x^{10}(t)'\right)' \in \mathbb{R}^{57} \tag{5.1.2}$$

system (5.1.1) becomes a *nonlinear* model of the form:

$$\dot{x}(t) \;=\; f\left(x(t), u(t)\right) \quad \forall\, t \in \mathbb{T}\,. \tag{5.1.3}$$

In order to obtain a *linear process model* we linearise system (5.1.1) at a steady-state. It can be shown that all velocities of the masses and the controllable end of the chain $\dot{x}^i(t)$, $1 \leq i \leq 10$, must be zero at a steady-state. Thus, if we fix the position of the free end of the chain, i.e. $x^{10}(t) \stackrel{\text{def}}{=} x_{\text{end}} \in \mathbb{R}^3$ for all $t \in \mathbb{T}$, the unique stable steady-state $(\hat{x}, \mathbb{0}) \in \mathbb{R}^{60}$ satisfying

$$\mathbb{0} \;=\; f\left(\hat{x}, \mathbb{0}\right) \tag{5.1.4}$$

is easily obtained. Afterwards, the system matrices of the linear process model (Eqs. (2.2.1)) are definied as

$$A \stackrel{\text{def}}{=} \frac{\partial f\left(\hat{x}, \mathbb{0}\right)}{\partial x(t)} \quad \text{and} \quad B \stackrel{\text{def}}{=} \frac{\partial f\left(\hat{x}, \mathbb{0}\right)}{\partial u(t)} \tag{5.1.5}$$

as well as

$$C \stackrel{\text{def}}{=} \mathrm{Id}_{57}\,. \tag{5.1.6}$$

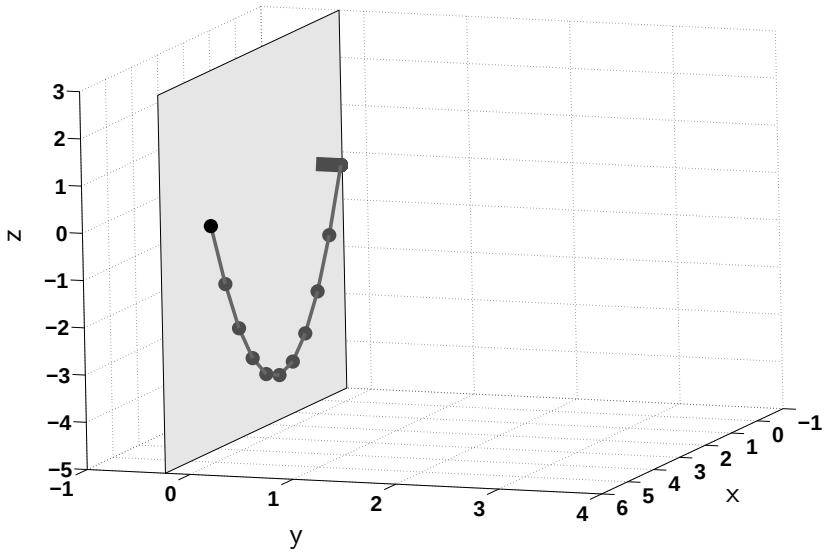


Figure 5.1: Chain of spring connected masses at its steady-state for $x_{\text{end}} = (5, 0, 0)$. (The controllable free end of the chain is symbolised by a black ball.)

The (quadratic) objective function is chosen such that deviations from the steady-state $(\hat{x}, \mathbb{0})$ are penalised:

$$\min_{x(t),\,u(t)} \frac{1}{2} \int_{t_0}^{t_0+t_\mathrm{p}} \big(x(t)-\hat{x}\big)' \underbrace{\begin{pmatrix} \mathbb{0} & & & & & \\ & \beta\cdot\mathrm{Id}_3 & & & & \\ & & \ddots & & & \\ & & & \mathbb{0} & & \\ & & & & \beta\cdot\mathrm{Id}_3 & \\ & & & & & \alpha\cdot\mathrm{Id}_3 \end{pmatrix}}_{\underset{\text{def}}{=} Q} \big(x(t)-\hat{x}\big) + u(t)' \underbrace{\begin{pmatrix} \gamma & & \\ & \gamma & \\ & & \gamma \end{pmatrix}}_{\underset{\text{def}}{=} R} u(t)\,dt\,, \tag{5.1.7}$$

with α, β, $\gamma \in \mathbb{R}_{>0}$. This choice implies $Q \in \mathcal{S}^{57}_{\succeq 0}$ and $R \in \mathcal{S}^{3}_{\succ 0}$, a terminal penalty weight matrix is not used (i.e. $P \overset{\text{def}}{=} \mathbb{0} \in \mathcal{S}^{57}_{\succeq 0}$).

Finally, we impose bounds on the process inputs

$$-1 \leq u_i(t) \leq 1 \quad \forall\, i \in \{1,2,3\} \tag{5.1.8}$$

and thus yielding the benchmark example from [86]. Additionally, we place a vertical wall (parallel to the second coordinate axis) near to the chain at steady-state $(\hat{x}, \mathbb{0})$; and we choose x_end such that the chain at this steady-state is hanging parallel to this wall (see Figure 5.1). Then we introduce lower bounds on the second component of the position of all balls, i.e. $\xi_\mathrm{wall} \leq x_2^i$ for all $1 \leq i \leq 9$, in order to prevent the chain from hitting the wall while it is controlled. In the notation of Definition 2.3, these constraints together with the bounds (5.1.8) read

$$\underbrace{\begin{pmatrix} \xi_\mathrm{wall} \\ \vdots \\ \xi_\mathrm{wall} \\ -\mathbb{1} \\ -\mathbb{1} \end{pmatrix}}_{\underset{\text{def}}{=} l} \leq \underbrace{\begin{pmatrix} e_2' & & \mathbb{0} \\ & \ddots & \vdots \\ & e_2' & \mathbb{0} \\ \mathbb{0} & & \mathbb{0} \\ \mathbb{0} & & \mathbb{0} \end{pmatrix}}_{\underset{\text{def}}{=} M} x(t) + \underbrace{\begin{pmatrix} \mathbb{0} \\ \mathbb{0} \\ \mathbb{0} \\ \mathrm{Id}_3 \\ -\mathrm{Id}_3 \end{pmatrix}}_{\underset{\text{def}}{=} N} u(t)\,, \tag{5.1.9}$$

with the second coordinate vector $e_2 \in \mathbb{R}^3$.

The continuous-time open-loop optimal control problem (5.1.1), (5.1.7), (5.1.9) is discretised into a finite optimisation problem, see Section 2.2, by dividing the prediction horizon of length $t_\mathrm{p} \overset{\text{def}}{=} 16\,\mathrm{s}$ into $n_\mathrm{p} \overset{\text{def}}{=} 80$ equidistant control intervals. The dimensions of the resulting parametric quadratic program (after the condensing procedure described in Section 2.2.3) are given in Table 5.2. Some numerical properties of this parametric quadratic program are summarised in Table 5.3; the used numerical values of all the above mentioned model constants are listed in Table 5.1.

Table 5.1: Numerical constants for the chain example.

Constant:	m	d	L	g	x_end	α	β	γ	ξ_wall
Value:	0.03	1	0.0333	$(0, 0, -9.81)$	$(5, 0, 0)$	50	2	0.02	-0.2

Table 5.2: Problem dimensions (after condensing) of the chain example.

Quantity:	Dimension:
Dimension of initial value vector	57
Number of variables	240
Number of bounds	480
Number of constraints	711

Table 5.3: Matrix properties of the chain example.

Property:	Value:
Condition number of Hessian matrix H	$1.01 \cdot 10^4$
Maximum eigenvalue of Hessian matrix H	$5.26 \cdot 10^0$
Minimum eigenvalue of Hessian matrix H	$5.20 \cdot 10^{-4}$
Number of nonzero elements of Hessian matrix H	57600 $(100.0\,\%)$
Condition number of constraint matrix G	$9.57 \cdot 10^3$
Numerical rank[1] of constraint matrix G	79
Number of nonzero elements of constraint matrix G	84368 $(49.6\,\%)$

5.2 Numerical Results

We simulate in a closed-loop manner integrating the *nonlinear* ODE system with high accuracy in order to obtain the movements of the chain. Since we control the chain using a linear model, feedback control is mandatory even in this nominal setup (i.e. without any noise or measurement errors). Starting at the steady-state corresponding to $x_{\text{end}} = (5, 0, 0)$, a strong perturbation is exerted to the chain by moving the free end with a constant velocity $(-1.5, 1.0, 1.0)\ \text{m/s}$ for 3 seconds. Then the MPC controller takes over and tries to return the chain into its original steady-state while not hitting against the wall. (Note that during the initial pertubation phase the optimiser is already running but the calculated optimal control action is not given back to the chain.) This scenario is simulated on the time horizon $[0, 20]\,\text{s}$ using a constant sampling time of $\delta \stackrel{\text{def}}{=} 0.2\,\text{s}$, i.e. $\sigma \stackrel{\text{def}}{=} 1$ in Eq. (2.2.9). It was tested with four different methods: first, we solve every QP exactly using three alternative methods:

- `qpsol` with cold start, i.e. initialisation with an empty working set and the origin as an intial guess for the solution,
- `qpsol` with warm start, i.e. the solver is initialised with the solution and corresponding working set of the previous QP (but without providing any matrix factorisations),
- online active set strategy as presented in Chapter 4 where we follow every homotopy path until the exact solution is reached.

[1] Number of (normalised) singular values greater than 10^{-15}; see [46] for a discussion on determing the rank of a matrix numerically.

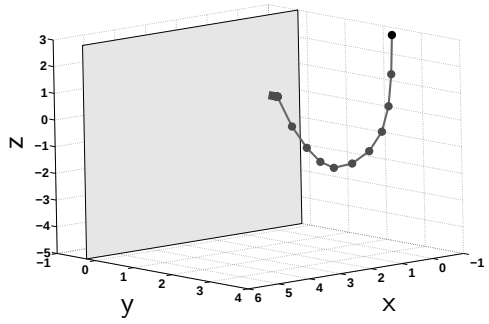
Second, we allow for an inexact QP solution by using the

- online active set strategy and limiting the maximum number of working set changes (as described in Section 4.2) to 10.

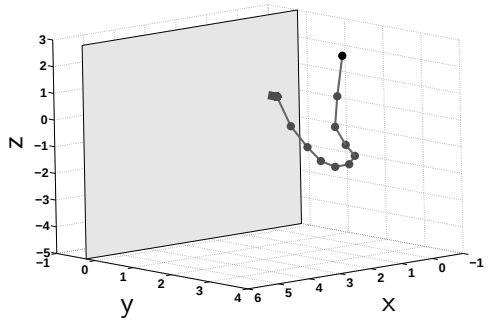
`qpsol` is a very common primal active set QP solver based on the null space method (see Section 3.1.1). It is written for QPs with *dense* matrices and solves an auxiliary LP for finding an initial feasible point during phase I. A description of the FORTRAN implementation is given in [62].

Figure 5.2 illustrates the optimally controlled chain at four particular time instants. The number of bounds and constraints' bounds active at the solution of each QP as well as the Euclidean norm of the QP solution vector are depicted in Figure 5.3 for the case of exact QP solution.

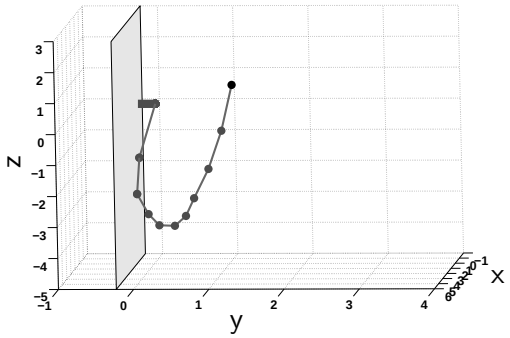
The number of QP iterations, i.e. the number of working set recalculations in the case of the online active set strategy, and runtimes[2] per sampling instant are reported in Table 5.4 and illustrated in Figures 5.4 and 5.5, respectively.



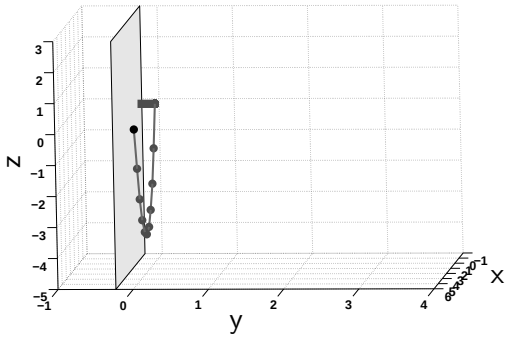
(a) At beginning of control phase ($t = 3\,\mathrm{s}$)



(b) Early moment in control phase ($t = 4\,\mathrm{s}$)

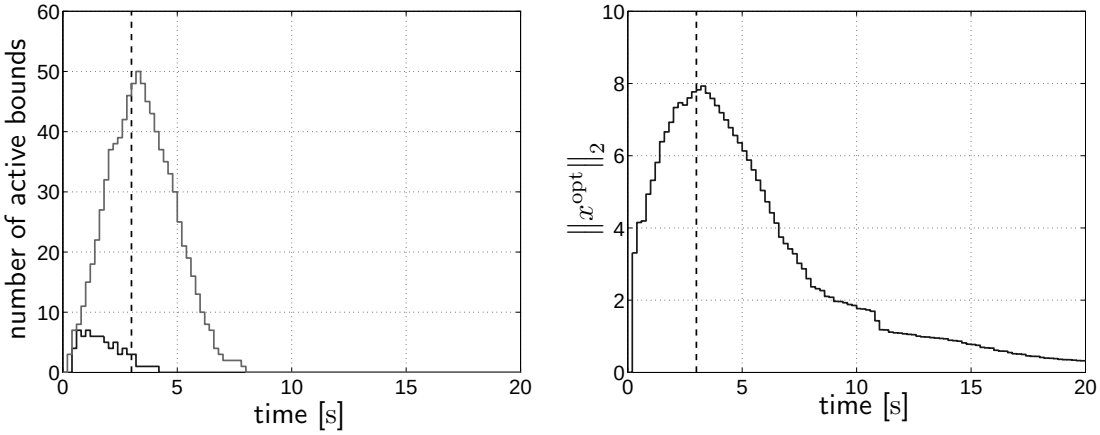


(c) Softly touching the wall ($t = 5.4\,\mathrm{s}$)



(d) Almost at steady-state ($t = 20\,\mathrm{s}$)

Figure 5.2: Optimally controlled closed-loop trajectory of the chain with exact QP solution.



(a) Number of active bounds (grey) and active constraints' bounds (black).

(b) Euclidean norm of the QP solution vector.

Figure 5.3: Properties of the exact QP solution for the optimally closed-loop controlled chain.

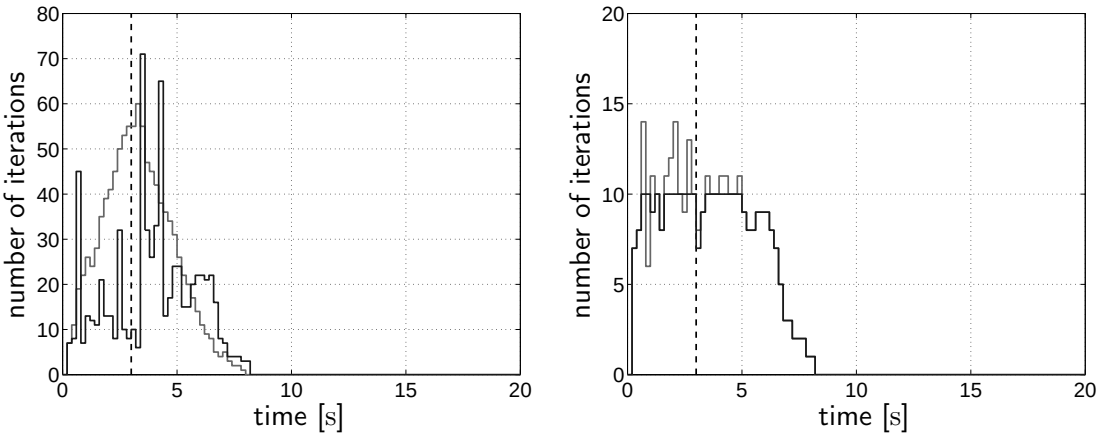
Table 5.4: Comparison of standard QP solver and online active set strategies with respect to runtimes and number of iterations.

Method:	Maximum runtime [ms]	Average runtime [ms]	Maximum no. of iterations	Average no. of iterations
`qpsol` (cold start)	1006.8	223.5	60	10.4
`qpsol` (warm start)	969.6	140.9	71	7.1
online active set strategy (fully converged)	74.8	18.5	14	3.3
online active set strategy (at most 10 iterations)	51.6	16.8	10	3.1

The solution, and thus also the optimal objective function value, are identical when using `qpsol` or the fully converged online active set strategy. Moreover, all QPs are feasible and so the optimal solution is feasible in these cases, too. However, note that tiny infeasibilities of the "real" chain with respect to constraint violations may occur between two sampling instances because the model is not exact. A qualitatively different form of infeasibilities can occur if the real-time variant of the online active set strategy is used: if the homotopy towards the new QP solution is stopped prematurely the solution of the *intermediate* QP might be suboptimal and infeasible with respect to the current QP that one wants to solve.

[2] All simulations were performed on an Intel Pentium 4 processor with 2.53 GHz (single core), 512 kB L2 cache and 1 GB main memory using gcc 3.3.4 with compiler flag -O3. The runtimes are obtained from multiple measurements with the linux-specific function `gettimeofday()` and should be accurate within some hundred microseconds.

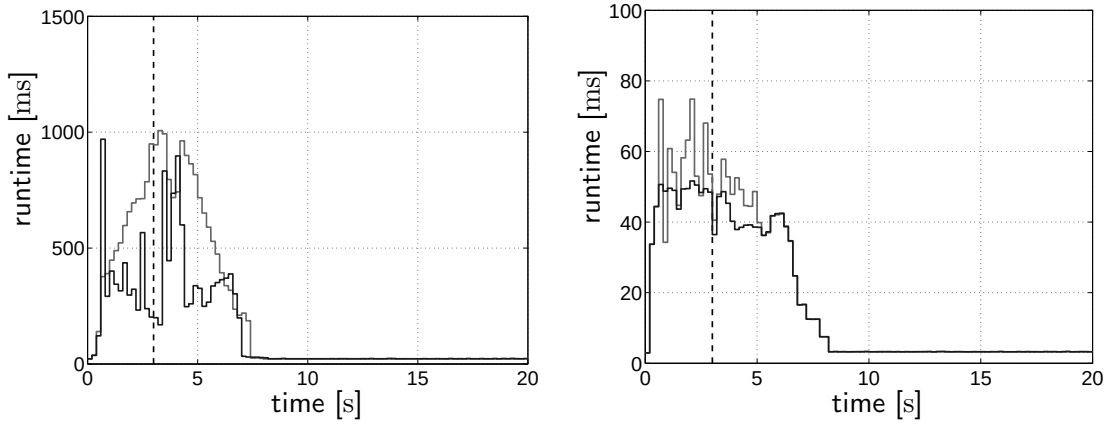
In the chain test scenario these possible infeasibilities are restricted to constraint violations because all bounds are equally fixed for all sampling times and their fulfilment is thus not affected by the current position along the homotopy path. Table 5.5 compares the MPC objective function over the whole simulation horizon $[0, 20]\,\mathrm{s}$ as well as the maximal "real" constraint violation of the solutions of the exact online active set strategy (or `qpsol`) and the inexact one.



(a) Standard QP solver (grey: cold start, black: warm start).

(b) Online active set strategy (grey: fully converged, black: real-time variant performing at most 10 iterations).

Figure 5.4: Number of iterations per sampling instant for chain example.



(a) Standard QP solver (grey: cold start, black: warm start).

(b) Online active set strategy (grey: fully converged, black: real-time variant performing at most 10 iterations).

Figure 5.5: Runtimes per sampling instant for chain example.

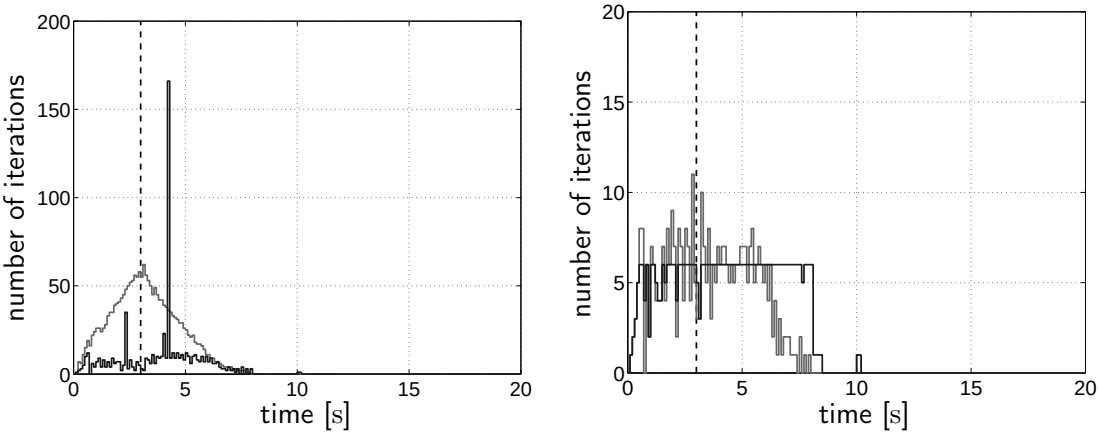
Table 5.5: Optimal MPC objective function value and maximum "real" infeasibility (constraint violation).

Method:	Optimal objective function value	Maximum "real" constraint violations
Exact QP solution	1747.07	0.0019
Inexact QP solution using the online active set strategy (at most 10 iterations)	1746.72	0.0056

Decreasing the Sampling Time to $\delta = 0.1\,\mathrm{s}$

As the runtimes of the online active set strategy are well below $0.2\,\mathrm{s}$, we can reduce the sampling time to $\delta = 0.1\,\mathrm{s}$, i.e. $\sigma \overset{\text{def}}{=} 2$ in Eq. (2.2.9), in order to react faster to inaccuracies due to the mentioned model-plant mismatch (note that the discretisation of the optimal control problem is *not* changed). We also simulate this slightly different setup using `qpsol`, even if this solver is not able to solve the occuring optimal control problems within this shorter time period.

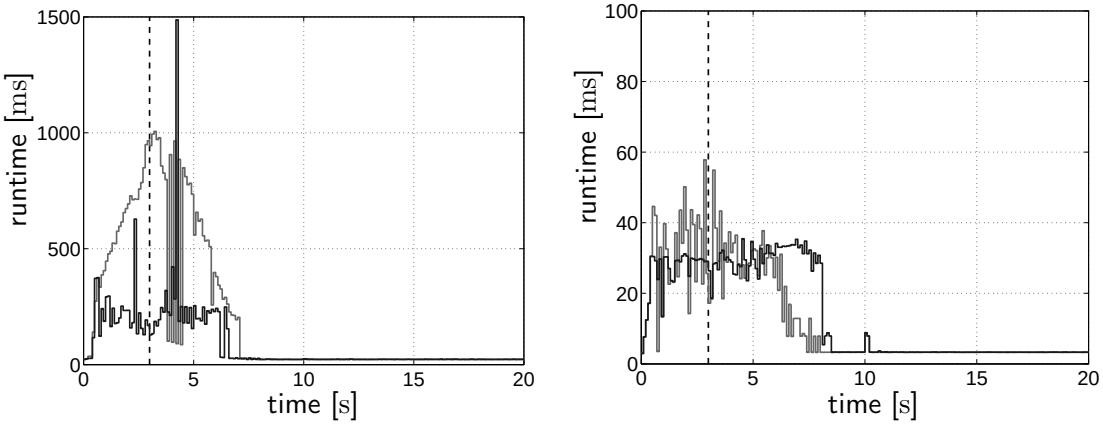
We do not illustrate the optimised trajectories and the properties of the QP solutions since they are very similar to that depicted in the Figures 5.2 and 5.3. The number of QP iterations and runtimes per sampling instant are summarised in Table 5.6 and illustrated in Figures 5.6 and 5.7, respectively. Again, the MPC objective function over the whole simulation horizon $[0, 20]\,\mathrm{s}$ (divided by two) as well as the maximal "real" constraint violation of the solutions of the exact online active set strategy (or `qpsol`) and the inexact one are reported in Table 5.7.



(a) Standard QP solver (grey: cold start, black: warm start).

(b) Online active set strategy (grey: fully converged, black: real-time variant performing at most 6 iterations).

Figure 5.6: Number of iterations per sampling instant for chain example ($\delta = 0.1\,\mathrm{s}$).



(a) Standard QP solver (grey: cold start, black: warm start).

(b) Online active set strategy (grey: fully converged, black: real-time variant performing at most 6 iterations).

Figure 5.7: Runtimes per sampling instant for chain example ($\delta = 0.1\,\mathrm{s}$).

Table 5.6: Comparison of standard QP solver and online active set strategies with respect to runtimes and number of iterations ($\delta = 0.1\,\mathrm{s}$).

Method:	Maximum runtime [ms]	Average runtime [ms]	Maximum no. of iterations	Average no. of iterations
`qpsol` (cold start)	1005.9	204.1	62	10.1
`qpsol` (warm start)	1487.1	89.2	166	3.4
online active set strategy (fully converged)	57.8	12.2	11	1.9
online active set strategy (at most 6 iterations)	35.5	13.7	6	2.3

Table 5.7: Optimal MPC objective function value and maximum "real" infeasibility (i.e. constraint violation) for $\delta = 0.1\,\mathrm{s}$.

Method:	Optimal objective function value	Maximum "real" constraint violations
Exact QP solution	1658.25	0.0041
Inexact QP solution using the online active set strategy (at most 6 iterations)	1686.26	0.0108

5.3 Summary of the Results

The most obvious observation is that the runtimes of the fully converged online active set strategy are significantly—more than an order of magnitude— shorter than that of `qpsol`, even if `qpsol` is performing warm starts. This is true for both the average and the more crucial maximum runtime. Thus, `qpsol` is far from being able to control the chain within the given sampling times, whereas the proposed online active set strategy meets the real-time requirements with ease[3]. Apparently, this results from a smaller number of QP iterations (the effort for one iteration of `qpsol` and the online active set strategy is comparable), but this fact cannot fully explain the enormous difference.
Some other things are also important: first, the primal solution of the preceding QP often is not a feasible initial value for the next QP, making a phase I necessary. Within the initial seven seconds of the simulation with $\delta = 0.2\,\mathrm{s}$ ($\delta = 0.1\,\mathrm{s}$), up to 13 (6) phase I LP iterations[4] were necessary if the warm start feature of `qpsol` is used. Instead, a cold start requires a phase I quite rarely (at most one LP iteration) since the origin is often a primal feasible point[5]. Second, our online active set strategy can use both matrix factorisations from the previous QP, whereas `qpsol` has to calculate them from scratch even if an initial guess for the active set is provided via the warm start feature. Finally, the runtimes of `qpsol` may suffer from some overhead because it also handles indefinite QPs. But even if a special positive definite QP variant of `qpsol` which is also able to maintain matrix factorisations would have been used, a considerable speedup of the proposed online active set method can be expected: a factor of 3-7 compared with cold starting and 2-4 compared with warm starting seems to be realistic according to the data given in Tables 5.4 and 5.6.

Besides the comparison with `qpsol`, the results of the online active set strategy (and its real-time variant) are interesting for themselves: first, reducing the sampling times also reduces both the maximum and the average number of required active set changes per sampling instant. This is a useful property from an application point of view because shorter sampling times normally result in a improved controller performance. Second, a proper restriction of the number of working set changes using the real-time variant leads to a further decrease of the maximum runtime (the average runtime is only slightly affected because the working set changes are more or less postponed to later sampling instants) without becoming much suboptimal or infeasible. For $\delta = 0.2\,\mathrm{s}$ the optimal objective function value of the real-time variant is even a little bit better due to a slight increase of infeasibilities; for $\delta = 0.1\,\mathrm{s}$ the "real" infeasibilities remain very small and only 1.7 % loss of optimality in the objective function value is oberserved. Of course, a theoretical performance guarantee cannot be given so far.

Finally, we remark that this test problem with a state-space dimension of 57 and far more than $3^{240} \approx 10^{114}$ possible active sets is by no means tractable with the explicit approach (as presented in Section 2.3.2).

[3] The reported runtimes do not include the effort for calculating the current gradient vector $g(w_0)$ and constraint vector $b(w_0)$ since it is almost negligible compared with the remaining online computations.

[4] Using $\delta = 0.1\,\mathrm{s}$, warm started `qpsol` performs 70 LP iterations and afterwards 166 QP iterations at $t = 4.2\,\mathrm{s}$. Since this simulation phase is quite crucial, this outlier could result in a heavy crash into the wall.

[5] Unfortunately, besides the number of LP iterations, `qpsol` provides no possibility to obtain the runtime required for phase I.

Chapter 6

Numerical Tests: Real-World Diesel Engine

6.1 Model Description and Problem Formulation

In this second test example we aim at controlling a real-world direct injection turbo charged Diesel engine on a dynamical testbench at the Institute for Design and Control of Mechatronical Systems of the Johannes Kepler University in Linz (Austria), see Figure 6.1.

In order to minimise the *emissions* we control the so-called *airpath* of the Diesel engine, which is depicted in Figure 6.2: fresh air streams through the compressor into the intake manifold inside the engine. From there it flows into the cylinders where the fuel is burnt for producing the engine torque. Afterwards, the exhaust gases (especially NO_x and soot) stream into the exhaust manifold from where they can flow in two directions: one part of them drives a *variable geometry turbocharger VGT* which spins up the compressor by means of a common shaft, and thus strongly influences the pressure in the intake manifold; the other part flows through the *exhaust gas recirculation (EGR) valve* and mixes with the fresh air. This already burnt gas acts as an inert gas during combustion which lowers the peak temperature and hence reduces the NO_x emissions. In modern Diesel engines both the opening of the EGR valve as well as the angle of the inlet guide vanes of the VGT can be controlled.

Modelling of the combustion process naturally leads to *partial differential equations*, where temporal as well as spatial derivatives are present and each explosion needs to be simulated—a nearly impossible task for today's computing capacity. Another possibility is the usage of so-called *mean value models* (without any spatial effects) leading to *nonlinear ODE systems*. A mean value model for Diesel engines can be found in [52], a similar one for gasoline engines is developed in Appendix C.
In order to employ our online active set strategy we need a linear process model, which could be derived by linearising the nonlinear ODE system from a mean value model at a certain point. Instead, we follow the ideas presented in [66], [67] and directly use *linear identification* techniques (see [58] for an introduction). To this end a discrete-time linear state-space model (2.2.10c)-(2.2.10d) is obtained from real measurements by fitting the input to the output data (via a least-squares-like *prediction error approach*).

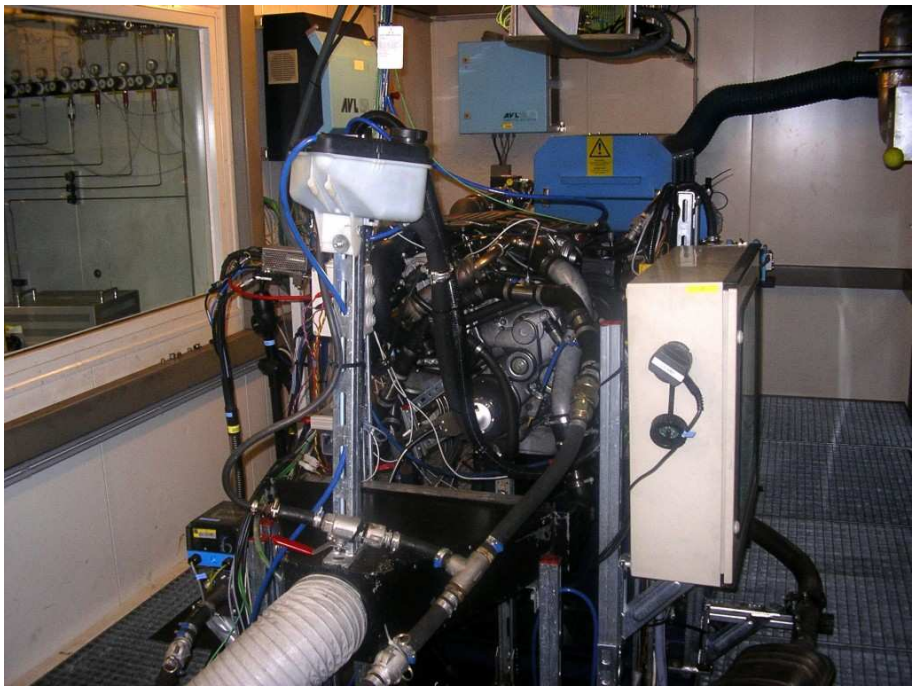


Figure 6.1: Diesel engine testbench at the University in Linz.

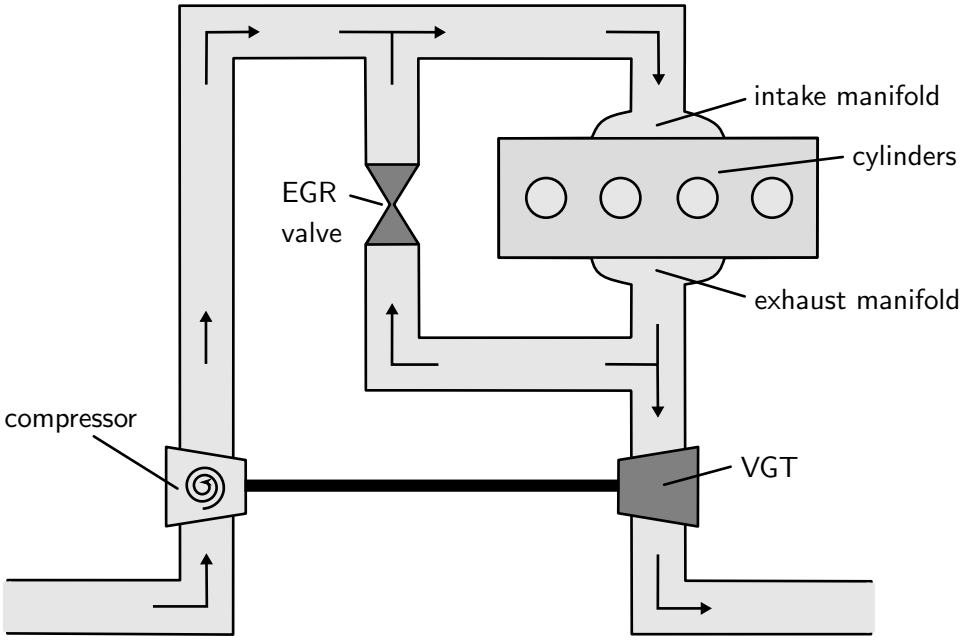


Figure 6.2: Schematic diagram of the Diesel engine airpath (inspired by [52])

Since the Diesel engine's dynamics are highly nonlinear it is not possible to derive a single linear model for the whole operating range (i.e. engine speed from 800 to 4500 rpm and fuel injection between 0 and 50 $\mathrm{mg/stroke}$). Therefore the operating range is empirically divided into twelve small operating areas and a linear process model is identified for each of them. The subsequent validation of all models with real engine data showed that the prediction quality of most of the models for the Diesel engine in Linz is good.

Instead of minimising the emissions directly, two process outputs—namely the *mass air flow (MAF)* through the compressor and the *manifold absolute pressure (MAP)* inside the intake manifold—are regulated to certain setpoints. These setpoints depend on the current operating point and are optimised (offline) with respect to emissions, fuel consumption and torque.

Thus, for each of the twelve operating areas we obtain an identified model of the following form:

$$x_{k+1} = A^{\mathrm{id}} x_k + E^{\mathrm{id}} x_k^{\mathrm{p}} + B^{\mathrm{id}} u_k \qquad \forall\, k \in \mathbb{N} \cup \{0\}\ , \tag{6.1.1a}$$

$$y_k = C^{\mathrm{id}} x_k \qquad \forall\, k \in \mathbb{N} \cup \{0\}\ , \tag{6.1.1b}$$

where $A^{\mathrm{id}} \in \mathbb{R}^{2\times 2}$, $B^{\mathrm{id}} \in \mathbb{R}^{2\times 2}$, $C^{\mathrm{id}} \in \mathbb{R}^{2\times 2}$. The inputs $u_k \in \mathbb{R}^2$ describe the position of the EGR and the VGT (normalised to lie between 0 and 100), the outputs $y_k \in \mathbb{R}^2$ contain the values of MAF and MAP. Moreover, the system states depend (via the matrix $E^{\mathrm{id}} \in \mathbb{R}^{2\times 2}$) on the current *engine speed* and the amount of *injected fuel*. They are treated as known parameters which are fixed over the whole prediction horizon; for each time step we summarise them in the vector $x_k^{\mathrm{p}} \in \mathbb{R}^2$.

Furthermore, the mismatch

$$x_k^{\mathrm{e}} \overset{\mathrm{def}}{=} y_k^{\mathrm{meas}} - y_k \quad \forall\, k \in \mathbb{N} \cup \{0\} \tag{6.1.2}$$

between the measured and the predicted outputs is estimated via a linear *Kalman filter* (see [66] for details) and is also assumed to be constant over the whole prediction horizon.

These modifications lead to the following augmented linear process model:

$$\begin{pmatrix} x_{k+1} \\ x_{k+1}^{\mathrm{p}} \\ x_{k+1}^{\mathrm{e}} \end{pmatrix} = \begin{pmatrix} A^{\mathrm{id}} & E^{\mathrm{id}} & \mathbb{0} \\ \mathbb{0} & \mathrm{Id}_2 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_2 \end{pmatrix} \begin{pmatrix} x_k \\ x_k^{\mathrm{p}} \\ x_k^{\mathrm{e}} \end{pmatrix} + \begin{pmatrix} B^{\mathrm{id}} \\ \mathbb{0} \\ \mathbb{0} \end{pmatrix} u_k \quad \forall\, k \in \mathbb{N} \cup \{0\}\ , \tag{6.1.3a}$$

$$y_k = \begin{pmatrix} C^{\mathrm{id}} & \mathbb{0} & \mathrm{Id}_2 \end{pmatrix} \begin{pmatrix} x_k \\ x_k^{\mathrm{p}} \\ x_k^{\mathrm{e}} \end{pmatrix} \qquad \forall\, k \in \mathbb{N} \cup \{0\}\ . \tag{6.1.3b}$$

Finally, two further augmentations of the state space are necessary: first, we introduce the desired setpoint, or reference, values of MAF and MAP as additional parameters, say $x_k^{\mathrm{r}} \in \mathbb{R}^2$ ($= y_{\mathrm{ref}}$ in Eq. (2.2.6)), as they are constant for one optimisation problem but may vary from one QP to the next. Second, we do not want to control EGR and VGT directly but their rates of change $\Delta u_k \in \mathbb{R}^2$ ($u_k = u_{k-1} + \Delta u_k$), instead. Thus, we end up with

an ODE system consisting of ten states:

$$\begin{pmatrix} x_{k+1} \\ x^{\mathrm{p}}_{k+1} \\ x^{\mathrm{e}}_{k+1} \\ x^{\mathrm{r}}_{k+1} \\ u_k \end{pmatrix} = \begin{pmatrix} A^{\mathrm{id}} & E^{\mathrm{id}} & \mathbb{0} & \mathbb{0} & B^{\mathrm{id}} \\ \mathbb{0} & \mathrm{Id}_2 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathrm{Id}_2 & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathrm{Id}_2 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathrm{Id}_2 \end{pmatrix} \begin{pmatrix} x_k \\ x^{\mathrm{p}}_k \\ x^{\mathrm{e}}_k \\ x^{\mathrm{r}}_k \\ u_{k-1} \end{pmatrix} + \begin{pmatrix} B^{\mathrm{id}} \\ \mathbb{0} \\ \mathbb{0} \\ \mathbb{0} \\ \mathrm{Id}_2 \end{pmatrix} \Delta u_k \,, \tag{6.1.4a}$$

$$y_k = \begin{pmatrix} C^{\mathrm{id}} & \mathbb{0} & \mathrm{Id}_2 & \mathbb{0} & \mathbb{0} \end{pmatrix} \begin{pmatrix} x_k \\ x^{\mathrm{p}}_k \\ x^{\mathrm{e}}_k \\ x^{\mathrm{r}}_k \\ u_k \end{pmatrix} \qquad \forall k \in \mathbb{N} \cup \{0\} \,. \tag{6.1.4b}$$

After this transformation it is possible to introduce bounds on the values as well as on the rate of change of EGR and VGT[1]:

$$\begin{pmatrix} -10 \\ -5 \end{pmatrix} \leq \Delta u_k \leq \begin{pmatrix} 3.3 \\ 5 \end{pmatrix} \qquad \forall\, k \in \mathbb{N} \cup \{0\} \,, \tag{6.1.5a}$$

$$\begin{pmatrix} 0 \\ 10 \end{pmatrix} \leq u_k \leq \begin{pmatrix} 100 \\ 70 \end{pmatrix} \qquad \forall\, k \in \mathbb{N} \cup \{0\} \,. \tag{6.1.5b}$$

The lower/upper bounds on the rate of the EGR valve have different absolute values because it has to work against a spring for opening.
The objective function[2] is chosen as:

$$\min_{\substack{x^\star_{k_0},\ldots,x^\star_{k_0+n_{\mathrm{p}}},\\ y_{k_0},\ldots,y_{k_0+n_{\mathrm{p}}},\\ u_{k_0},\ldots,u_{k_0+n_{\mathrm{p}}}\\ \Delta u_{k_0},\ldots,\Delta u_{k_0+n_{\mathrm{p}}-1}}} \frac{1}{2} \sum_{k=k_0}^{k_0+n_{\mathrm{p}}-1} (y_k - y_{\mathrm{ref}})' \underbrace{\begin{pmatrix} 2 & \\ & 2 \end{pmatrix}}_{\substack{=\\ \mathrm{def}\\ Q}} (y_k - y_{\mathrm{ref}}) + \Delta u_k{}' \underbrace{\begin{pmatrix} 1 & \\ & 1 \end{pmatrix}}_{\substack{=\\ \mathrm{def}\\ R}} \Delta u_k \, dt \,, \tag{6.1.6}$$

where $x^\star_i$ denotes x_i, x^{p}_i, x^{e}_i or x^{r}_i for all $k_0 \leq i \leq k_0 + n_{\mathrm{p}}$. The prediction horizon of $t_{\mathrm{p}} \overset{\mathrm{def}}{=} 4\,\mathrm{s}$ length is divided into $n_{\mathrm{p}} \overset{\mathrm{def}}{=} 9$ equidistant control intervals, each of 50 ms length, and a tenth one with length 3.55 s.
The dimensions of the resulting parametric quadratic program (after the condensing procedure described in Section 2.2.3) for the fifth operating area are given in Table 6.1. Some numerical properties of this parametric quadratic program are summarised in Table 6.2.

Table 6.1: Problem dimensions (after condensing) of the Diesel engine example.

Quantity:	Dimension:
Dimension of initial value vector	10
Number of variables	20
Number of bounds	40
Number of constraints	40

[1] The given numerical values are valid for the fifth operating area (engine speed: 2100-2500 rpm, injected fuel: 0-30 mg/stroke).

[2] When comparing the input and output weights R and Q, note that the inputs are almost two orders of magnitude smaller than the outputs.

Table 6.2: Matrix properties of the Diesel engine example, fifth operating area.

Property:	Value:
Condition number of Hessian matrix H	$4.64 \cdot 10^4$
Maximum eigenvalue of Hessian matrix H	$1.00 \cdot 10^0$
Minimum eigenvalue of Hessian matrix H	$2.16 \cdot 10^{-5}$
Number of nonzero elements of Hessian matrix H	400 (100.0 %)
Condition number of constraint matrix G	$1.32 \cdot 10^1$
Numerical rank of constraint matrix G	20
Number of nonzero elements of constraint matrix G	110 (27.5 %)

6.2 Numerical Results

We perform closed-loop simulations using the linear model of the fifth operating area (engine speed: 2100-2500 rpm, injected fuel: 0-30 $\mathrm{mg/stroke}$). The engine speed as well as the amount of injected fuel is kept constant—at 2300 rpm and 15 $\mathrm{mg/stroke}$, respectively—and the controller shall track two step changes of the setpoints for MAF and MAP. The Diesel engine is simulated by integrating the linear model and adding (uniformly distributed) *white noise*[3] to the measured (i.e. simulated) MAF and MAP values; a linear Kalman filter is used to estimate the true values. Moreover, white noise is also added to the values of speed and injected fuel as they have to be measured in practice. Finally, the sampling time is chosen to be $\delta = 50\,\mathrm{ms}$. This setup corresponds to that described in [66] and was implemented in a Matlab/Simulink environment [59] (see Figure 6.3).

As in the chain benchmark problem (cp. Chapter 5), the simulations were conducted by using:

- `qpsol` with cold and warm starts,
- online active set strategy with exact QP solution and with the number of working set changes limited to 10 and 5, respectively.

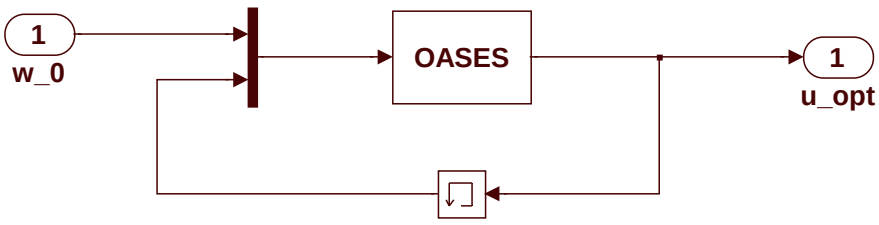


Figure 6.3: Implementation of the online active set strategy (`OASES`) compiled into a Matlab/Simulink block.

[3] We used the same noise sequence for all simulations by starting the random number generator with a fixed seed value.

Moreover, the *explicit approach* (as described in Section 2.3.2) was employed. In doing so, we encountered difficulties due to exponential complexity of the required precalculation: the Matlab Hybrid Toolbox [6] failed to precalculate an explicit controller for control horizon lenghts greater than two (although this most likely resulted from an internal error); for $n_\mathrm{p} = 5$ it stopped after several minutes and more than 15000 regions found with the message "unexpected degeneracy condition". Since a control horizon of length $n_\mathrm{p} = 10$ could lead to about $2.6 \cdot 10^{17}$ critical regions[4], 15000 should be a strongly underestimating lower bound on their actual number. So, if we make the conservative assumption that every region is described by 10 inequalities, even 15000 critical regions would require $15000 \cdot 10^2 \cdot 8\,\mathrm{byte} \approx 12\,\mathrm{Mbyte}$ of memory (about $150\,\mathrm{Mbyte}$ for all 12 linear models!). And if a linear search through all regions is performed online (as implemented in the Hybrid Toolbox), half of them need to be checked on average which requires about one million floating-point operations. On a Pentium IV processor this may take some hundred microseconds, a value that is easily achieved using our online active set strategy, as we will see soon.
Therefore, we compare the results of the online computation (using a control horizon of 10 intervals) with an explicit controller based on only one control interval. This controller comprises 25 critical regions and was used in [66], [67] to perform real-world closed-loop experiments on the above-mentioned Diesel engine.

We simulated on the time horizon $[0, 30]\,\mathrm{s}$ with a constant sampling time of $\delta \stackrel{\mathrm{def}}{=} 50\,\mathrm{ms}$, starting from a steady-state. The reference values used for MAF and MAP are depicted in Figure 6.4, together with the optimised outputs. The optimised inputs are shown in Figure 6.5. Since the output trajectories as well as the inputs are nearly identical for all online QP solutions (i.e. also for the inexact QP solution using the real-time variant of the online active set strategy), only the values for exact online QP solution and that of the explicit approach (with one control interval) are compared. The number of bounds and constraints' bounds active at the solution of each QP as well as the Euclidean norm of the QP solution vector are depicted in Figure 6.6 for the case of exact online QP solution.

The number of QP iterations, i.e. the number of working set recalculations in case of the online active set strategy, and runtimes[5] per sampling instant are illustrated in Figures 6.7 and 6.8, respectively. The maximum number of iterations, the maximum runtime and the MPC objective function evaluated over the whole simulation horizon are summarised in Table 6.3. In case of the real-time variant (limited to five working set changes) of the online active set strategy the value of the EGR opening becomes infeasible at one sampling instant (-1.8 at $10.1\,\mathrm{s}$) and is therefore clipped to 0.

Finally, we want to mention that both matrix factorisations remained very accurate during the whole simulation: their maximum deviation from their exact counterparts lay below

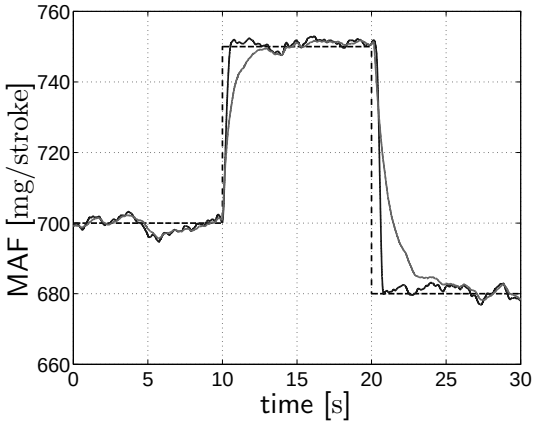
[4] It this case the maximum number of different optimal active set/critical regions can be calculated via

$$\sum_{j=0}^{2n_\mathrm{p}} \sum_{k=0}^{j} 2^k \binom{2n_\mathrm{p}}{k} \cdot 2^{j-k} \binom{2n_\mathrm{p}}{j-k},$$

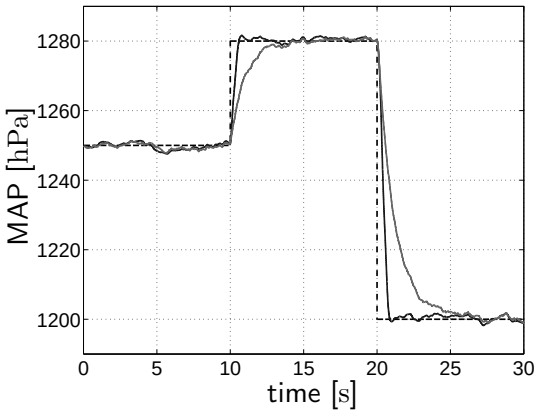
using a simple combinatorial argument.

[5] All simulations were performed on an Intel Pentium 4 processor with 2.53 GHz (single core), 512 kB L2 cache and 1 GB main memory using gcc 3.3.4 with compiler flag -O3. The runtimes are obtained from a series of measurements with the linux-specific function `gettimeofday()` and should be accurate in the order of 10-50 microseconds.

machine precision. Furthermore, as expected for this small-scale example, computational overhead for the alternative step length determination (as described in Section 4.7.1) outweighed the benefit.

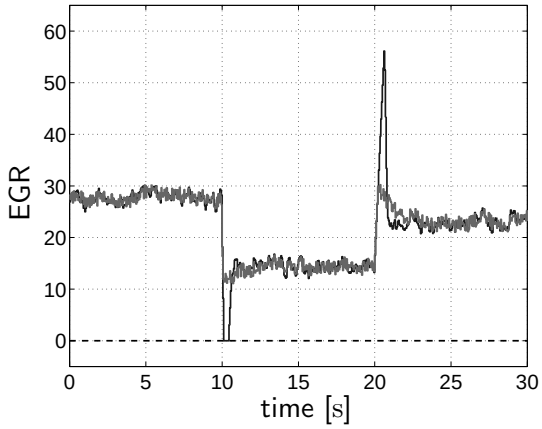


(a) Optimised MAF values (grey: explicit approach, black: exact online QP solution, dashed: reference value).

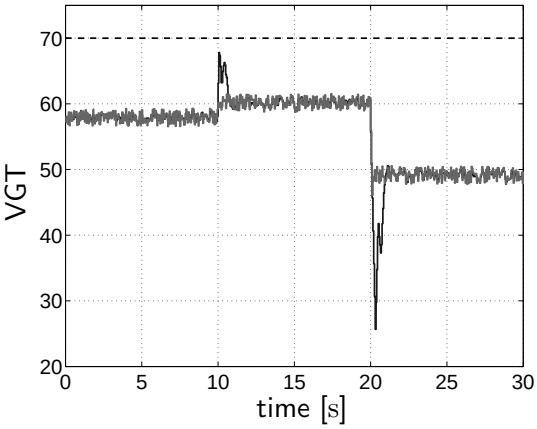


(b) Optimised MAP values (grey: explicit approach, black: exact online QP solution, dashed: reference value).

Figure 6.4: Optimised outputs for Diesel engine example.

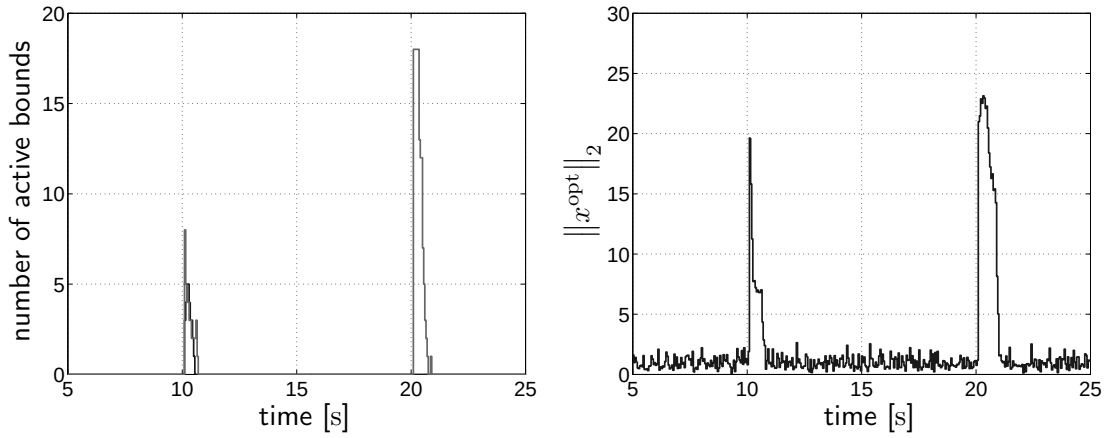


(a) Optimised EGR values (grey: explicit approach, black: exact online QP solution, dashed: lower bound).



(b) Optimised VGT values (grey: explicit approach, black: exact online QP solution, dashed: upper bound).

Figure 6.5: Optimal controls for Diesel engine example.



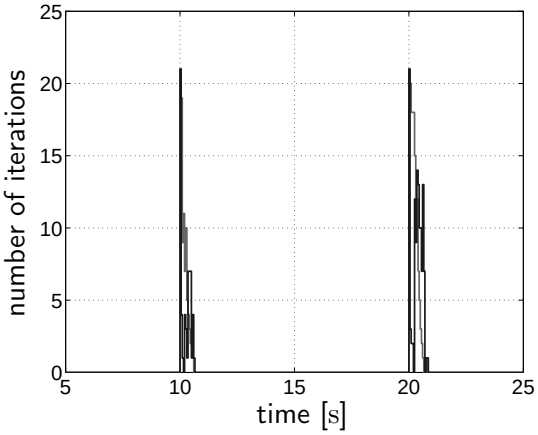
(a) Number of active bounds (grey) and active constraints' bounds (black).

(b) Euclidean norm of online QP solution vector.

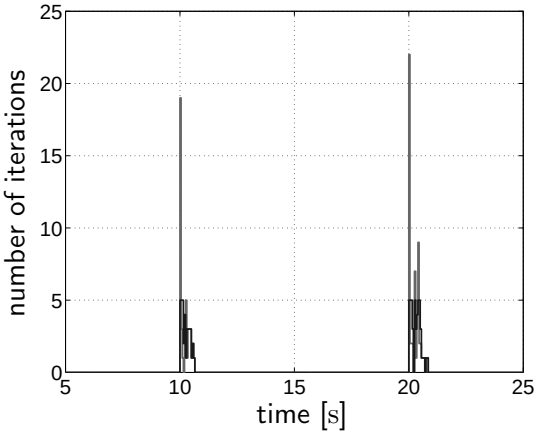
Figure 6.6: Properties of the exact online QP solution for the optimally closed-loop controlled Diesel engine.

Table 6.3: Comparison of an standard online QP solver, the online active set strategies as well as the explicit approach with respect to runtimes, number of iterations and MPC objective function value.

Method:	Maximum runtime [ms]	Maximum no. of iterations	Optimal objective function value
`qpsol` (cold start)	3.03	21	4851.7
`qpsol` (warm start)	2.67	21	4851.7
online active set strategy (fully converged)	0.41	22	4851.7
online active set strategy (at most 10 iterations)	0.22	10	4851.8
online active set strategy (at most 5 iterations)	0.13	5	4851.2
explicit approach	< 0.01	–	6497.3

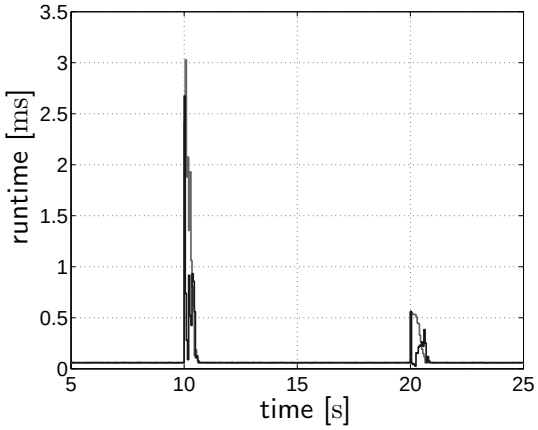


(a) Standard online QP solver (grey: cold start, black: warm start).

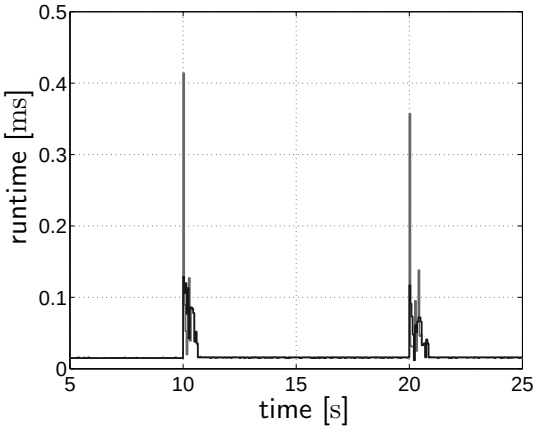


(b) Online active set strategy (grey: fully converged, black: real-time variant performing at most 5 iterations).

Figure 6.7: Number of iterations per sampling instant for Diesel engine example.



(a) Standard online QP solver (grey: cold start, black: warm start).



(b) Online active set strategy (grey: fully converged, black: real-time variant performing at most 5 iterations).

Figure 6.8: Runtimes per sampling instant for Diesel engine example.

6.3 Summary of the Simulation Results

The most important observation from a practical point of view is that reference tracking performance is considerably improved by using many control intervals. The period required for reaching a new MAF/MAP setpoint after a step change is greatly reduced, from about three to below one second, as can be seen in Figure 6.4. Since not only absolute bounds but also limits on the rate of change of the manipulated variables are considered within

the optimisation problem, it should be possible to directly realise these improvements in practice. The necessary optimal control problem formulation with an increased number of degrees of freedom calls for a fast online QP solver, instead of an explicit approach, as argued above.

Comparing the results of `qpsol` and our online active set strategy shows that the number of iterations for exact QP solution are quite similar. This might be due to the fact that the constraints' bounds exhibit a very special structure—EGR and VGT are artificial states, introduced in order to deal with their (discretised) derivatives. This probably leads to a special geometry of the partition of the set of feasible parameters and thus to similar steps of the conventional primal method and the proposed online active set strategy. The significantly higher runtime of `qpsol` at the first setpoint change is not yet fully understood. This effect only occurs if many constraints become active. It also persists when the dimension of the QP is varied. At the second setpoint change, when only bounds become active, an equal number of iterations also leads to comparable runtimes.
Nevertheless, this example clarifies the advantages of the real-time variant of the online active set strategy: almost without becoming suboptimal or infeasible, it was possible to reduce the number of working set changes by a factor of four (compared with exact QP solution)! This result justifies the conjecture that it might not be necessary to solve every QP exactly if the initial state is disturbed by measurement noise. Reducing the computational runtime in this way makes online QP solution definitely viable for this kind of problem, even if cheap (and hence slow) controller CPUs are used.

6.4 Real-World Experiments

The simulation results presented so far encourage our aim to perform closed-loop *real-world* experiments at the testbench in Linz. Preliminary tests, using a simplified implementation of our online active set strategy which could handle bounds on the inputs only, were already performed in spring 2006. For this purpose, the C++ source code was integrated into a Matlab/Simulink controller and implemented on the rapid prototyping hardware system *dSPACE* [28], which directly controls the engine. The dSPACE hardware is about five to ten times slower than a common Pentium IV processor; thus, when looking at the runtimes in Section 6.2, one should increase them in mind by one order of magnitude (which means at most $4\,\mathrm{ms}$ for the online active set strategy).

Another question is how to switch the controller between different models for different operating areas. On the one hand, it is possibile to let several QP solvers be running at the same time; on the other hand, if these switches do not occur too frequently, a cold start in the new operating area seems feasible. A third possibility is to apply the extension of our online active set strategy to problems with varying QP matrices (as described in Section 4.7.2). This might make sense due to the expectation that the active set will be similar across neighbouring operating areas. The most appropriate approach for this application would be to allow the QP matrices to change in every iteration, which directly leads to *nonlinear MPC*.

Chapter 7

Conclusions and Outlook

In this Diplom thesis, we presented the main concepts of model predictive control and showed that the resulting optimal control problems can be formulated as quadratic programs, provided that the objective function is quadratic and the ODE model as well as the constraints are linear. It was shown that these quadratic programs depend linearly on the current state of the controlled process; the special structure of these *parametric* quadratic programs was analysed and some of their important properties were presented. We also outlined several existing methods for solving these quadratic programs, namely active set methods and the so-called explicit approach.

After these theoretical preparations a new online active set strategy for the fast solution of (parametric) quadratic programming problems arising in model predictive control was developed. This strategy builds on ideas from parametric optimisation and fully exploits the knowledge of the solution of the previous quadratic program making the assumption that the active set does not change much from one quadratic program to the next. Furthermore, we showed how this strategy can be modified to make it suitable for real-time applications. We addressed various important ingredients for an efficient implementation of our method and also described procedures for dealing with degenerated QPs. Complexity issues and a possbile extension of the proposed method to nonlinear model predictive control problems were discussed.

Finally, we investigated the performance of our $\mathrm{C}{+}{+}$ implementation of the online active set strategy with two test examples: a challenging medium-scale benchmark problem and a small-scale problem for controlling a real-world Diesel engine in a closed-loop manner. In these examples, our strategy turned out to be significantly faster than a standard active set QP solver (even if the conventional warm start technique is used) while overcoming the prohibitive limitations of the explicit approach to MPC optimisation.

Future work will go into three major directions: (i) improvements and performance tests of the current implementation, (ii) extensions of the online active set strategy to other problem classes, and (iii) its application to real-world control problems.

(i) First, some refinements of the current implementation from a theoretical as well as from a software engineerical point of view are still conceivable. For example, it might be possible to incorporate so-called *long steps* when an active constraints swaps within one sampling period from its upper to its lower bounds (or vice versa), which causes

two—unnecessary from hindsight—active set changes within our current algorithm. Also a theoretical bound on the suboptimality if the homotopy is stopped prematurely would be desirable. Furthermore, a more extensive benchmarking will show if our strategy is also superior to other QP solvers written with MPC applications in mind.

(ii) Second, we want to adapt the proposed online active set method in order to make it suitable for sequential quadratic programming for solving nonlinear model predictive control problems. The main ideas of this extension were already described in Chapter 4 and will be implemented soon. Moreover, extending the applicability of our method to (not strictly) convex quadratic or linear programs seems to be possible and useful.

(iii) Finally, the simulations of the Diesel engine presented in Chapter 6 will form the basis of *closed-loop real-world* experiments, scheduled for the end of the year 2006. Besides performance improvements like reduction of NO_x emissions or soot formation, these tests will hopefully give further insight into practical requirements for making model predictive control a viable control strategy for fast applications in the millisecond range.

Appendix A

Mathematical Basics

In order to ease the presentation some basic definitions and results are collected in this appendix, instead of giving them where they first occur. Since it is assumed that the reader is familiar with all concepts they are stated without further explanation.

Definition A.1 (convex set): *A set $\mathcal{X} \subseteq \mathbb{R}^n$ is called* convex *iff*

$$\tau x_1 + (1-\tau)x_2 \ \in \ \mathcal{X} \tag{A.1}$$

for all $x_1,\, x_2 \in \mathcal{X}$ and all $\tau \in [0,1] \subset \mathbb{R}$. ○

Definition A.2 (convex function): *A real-valued function $f\colon \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ is called* convex *iff $\mathcal{D}$ is a convex set and*

$$f\left(\tau x_1 + (1-\tau)x_2\right) \ \leq \ \tau f\left(x_1\right) + (1-\tau) f\left(x_2\right) \tag{A.2}$$

for all $x_1,\, x_2 \in \mathcal{D}$ and all $\tau \in [0,1] \subset \mathbb{R}$. ○

Definition A.3 (polyhedron): *A set $\mathcal{X} \subseteq \mathbb{R}^n$ is called* polyhedron *iff there exist a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$ such that*

$$\mathcal{X} \ = \ \left\{ x \in \mathbb{R}^n \ \middle|\ Ax \leq b \right\}. \tag{A.3}$$

○

Definition A.4 (range space and null space of a matrix): *Let a matrix $A \in \mathbb{R}^{m\times n}$ be given.*

(i) Its range space *(or* image*)* $\operatorname{im} A$ *is the vector space spanned by the columns of A, i.e.*

$$\operatorname{im} A \ \stackrel{\text{def}}{=} \ \left\{ Ax \ \middle|\ x \in \mathbb{R}^n \right\} \subseteq \mathbb{R}^m. \tag{A.4}$$

(ii) Its null space *(or* kernel*)* $\ker A$ *is defined as*

$$\ker A \ \stackrel{\text{def}}{=} \ \left\{ x \in \mathbb{R}^n \ \middle|\ Ax = \mathbb{0} \right\}. \tag{A.5}$$

○

Theorem A.1 (Cholesky decomposition): *For every matrix $A \in \mathcal{S}^n_{\succ 0}$ there exists a unique upper triangular matrix $R \in \mathbb{R}^{n\times n}$ with positive diagonal entries such that*

$$A \;=\; R'R\,. \tag{A.6}$$

Matrix R, or its transposed $L \stackrel{\text{def}}{=} R'$, is called Cholesky factor *of A.* ○

Proof: Can be found in [46, p. 143]. □

Theorem A.2 (QR factorisation): *Let a matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ be given. Then the following holds:*

(i) There exist an orthonormal matrix $V \in \mathbb{R}^{m\times m}$ and an upper triangular matrix $U \in \mathbb{R}^{n\times n}$ such that

$$A \;=\; V \begin{pmatrix} U \\ \mathbb{0} \end{pmatrix}. \tag{A.7}$$

(ii) If A has full row rank there exist an orthonormal matrix $V \in \mathbb{R}^{m\times n}$ and an upper triangular matrix $U \in \mathbb{R}^{n\times n}$ with positive diagonal entries such that

$$A \;=\; VU\,. \tag{A.8}$$

This factorisation is unique. ○

Proof: Can be found in [46, p. 223–230]. □

Definition A.5 (condition number of a matrix): *For every matrix $A \in \mathbb{R}^{m\times n}$, $A \neq \mathbb{0}$, the* condition number $\operatorname{cond} A$ *is defined as*

$$\operatorname{cond} A \;\stackrel{\text{def}}{=}\; \left\|A^\dagger\right\|_2 \left\|A\right\|_2\,. \tag{A.9}$$

Therein $A^\dagger$ denotes the so-called pseudoinverse *of A which coincide with A^{-1} if the matrix A is invertible (see [41, p. 170–172]).* ○

Definition A.6 (big-O notation): *For every scalar function $f\colon \mathbb{N} \to \mathbb{N}$ we define*

$$\mathcal{O}(f) \;\stackrel{\text{def}}{=}\; \left\{g\colon \mathbb{N} \to \mathbb{N} \;\middle|\; \exists\, \alpha,\, \beta,\, n_0 \in \mathbb{N}:\; g(n) \leq \alpha f(n) + \beta \;\forall\, n \geq n_0\right\} \tag{A.10}$$

as the set of all integer functions which are asymptotically dominated by f. ○

Appendix B

Implementation Overview

Now we give a concise overview about the practical implementation of the proposed online active set strategy: the software module OASES. It is thought to be a guideline for actually setting up and solving sequences of strictly convex quadratic programs with OASES; theoretical issues and numerical results were addressed in the main part of this thesis.

B.1 Software Module OASES

The software module OASES is written in an object-oriented manner in C++ and comes along with the fully commented[1] files listed in Table B.1. Besides some standards libraries no further software packages are required. Core of the module is the QProblem class which is able to store, process and solve strictly quadratic programs using the online active set strategy; it makes use of several auxiliary classes.

Table B.1: Complete file list of the software module OASES.

File name:	Description:
OASES_QProblem.cpp/hpp/ipp	QProblem class for using the online active set strategy for strictly convex QPs
OASES_SubjectTo.cpp/hpp/ipp	QProblem_SubjectTo class for managing working sets of constraints or variables of a QProblem
OASES_Bounds.cpp/hpp/ipp	QProblem_Bounds class for managing working sets of variables of a QProblem
OASES_Constraints.cpp/hpp/ipp	QProblem_Bounds class for managing working sets of constraints of a QProblem
OASES_Indexlist.cpp/hpp/ipp	QProblem_Indexlist class for managing index lists of constraints or bounds within the QProblem_SubjectTo class
OASES_Utils.cpp/hpp	some utilities for working with the QProblem class
OASES_main.cpp	main function sample for testing the QProblem class

[1] All comments can be interpreted by the documentation system *doxygen* [80].

B.2 OASES in a Nutshell

The user interacts with the OASES module solely via the QProblem class. So, for setting up a quadratic program an instance of the QProblem class has to be created. This can be done by different constructors, e.g. the following

```
QProblem::QProblem( const double* H, const double* A, const double* g,
                    const double* lb, const double* ub,
                    const double* lbA, const double* ubA,
                    int nV, int nC );
```

which takes the (positive definite) Hessian matrix H, the constraint matrix A, the gradient vector g, the lower and upper bound vectors lb and ub, the lower and upper constraints' bound vectors lbA and ubA, the number of variables nV and the number of constraints nC of the quadratic program to be solved. All these data must be stored in arrays of type double (matrices stored row-wise in an one-dimensional array). A further constructor for QPs whitout constraints exists, as well as constructors for reading the data directly from ASCII files.

After setting up the first quadratic program it has to be initialised via the function:

```
int QProblem::init( int& nWSR, bool objFLAG, double& cputime );
```

It initialises all internal data structures and *solves* the quadratic program using the techniques described in Section 4.4. The argument nWSR specifies the maximum number of working set recalculations to be performed during the initial homotopy (on output in contains the number of working set recalculations actually performed). objFLAG indicates if also the optimal objective function value shall be calculated; cputime contains (on output) the CPU time required for the whole initialisation. The functions init() returns a status code which indicates if the initialisation was successful. Alternatively, the function solve() provides an interface for solving the quadratic program with a different solver (e.g. qpsol).

If not only a single quadratic program but a whole sequence of QPs shall be solved—as it is the usual situation for a MPC problem—the next QP can be solved using the function:

```
int QProblem::hotstart( const double* g_new,
                        const double* lb_new, const double* ub_new,
                        const double* lbA_new, const double* ubA_new,
                        int& nWSR, bool objFLAG, double& cputime );
```

The next QP is specified by passing its gradient vector g_new, its lower and upper bound vectors lb_new and ub_new as well as lower and upper constraints' bound vectors lbA_new and ubA_new. It is solved by means of the online active set strategy using at most nWSR working set recalculations. objFLAG indicates if also the optimal objective function value shall be calculated; cputime contains (on output) the CPU time required for nWSR steps along the homotopy path. The function hotstart() returns a status code which indicates, e.g., if the optimal solution of the next QP could be found within the given number of working set recalculations or if an error occured. Again, special (overloaded) variants for QPs whitout constraints or for reading the data of the next QP directly from ASCII files exist.

Besides this main functionality, several functions for obtaining status information are implemented. Among them

```
double* QProblem::getPrimalSolution ( )
double* QProblem::getDualSolution ( )
double  QProblem::getObjVal( )
```

for getting the primal-dual solution pair $\left(x^{\mathrm{opt}}, y^{\mathrm{opt}}\right)$ and the optimal objective function value or

```
bool QProblem::isInitialised( )
bool QProblem::isSolved( )
bool QProblem::isInfeasible( )
```

for asking if the current QP was initialised, solved or found to be infeasible. Moreover, several output functions are available.

We conclude by presenting a very simple example for illustrating the handling of the OASES module:

```
#include "OASES_QProblem.hpp"

int main( )
{
    // data of first QP
    double H[2*2] = { 1.0, 0.0, 0.0, 0.5 };
    double A[1*2] = { 1.0, 1.0 };
    double g[2]   = { 1.0, 1.0 };
    double lb[2]  = { 0.5, -2.0 };
    double ub[2]  = { 5.0,  2.0 };
    double lbA[1] = { -1.0 };
    double ubA[1] = { 2.0 };

    // data of second QP
    double g_new[2]   = { 1.0, 1.0 };
    double lb_new[2]  = { 0.0, -1.0 };
    double ub_new[2]  = { 5.0, -0.5 };
    double lbA_new[1] = { -2.0 };
    double ubA_new[1] = { 1.0 };

    // setting up first QP
    QProblem testExample( H,A,g,lb,ub,lbA,ubA, 2,1 );

    // solve first QP
    double cputime;
    int nWSR = 10;
    testExample.init( nWSR,true,cputime );

    // solve second QP
    nWSR = 10;
    testExample.hotstart( g_new,lb_new,ub_new,lbA_new,ubA_new, nWSR,true,cputime );

    return 0;
}
```

Appendix C

Fast Nonlinear Model Predictive Control of Gasoline Engines

As an example for NMPC applications we reprint a publication recently presented at the IEEE International Conference on Control Applications 2006 in Munich [31].

Not included in this online version (for copyright reasons)!

Bibliography

[1] A.A. Anda and H. Park. Fast plane rotations with dynamic scaling. *SIAM Journal on Matrix Analysis and Applications*, 15(1):162–174, 1994.

[2] M. Athans and P.L. Falb. *Optimal Control*. McGraw-Hill, New York, 1966.

[3] R.A. Bartlett and L.T. Biegler. QPSchur: A dual, active set, schur complement method for large-scale and structured convex quadratic programming algorithm. *Optimization and Engineering*, 7:5–32, 2006.

[4] R.A. Bartlett, L.T. Biegler, J. Backstrom, and V. Gopal. Quadratic programming algorithms for large-scale model predictive control. *Journal of Process Control*, 12:775–795, 2002.

[5] R.A. Bartlett, A. Wächter, and L.T. Biegler. Active set vs. interior point strategies for model predicitve control. In *Proceedings of the American Control Conference*, pages 4229–4233, Chicago, Il, 2000.

[6] A. Bemporad. *Hybrid Toolbox – User's Guide*, 2004.

[7] A. Bemporad and C. Filippi. Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming. *Journal of Optimization Theory and Applications*, 117(1):9–38, 2003.

[8] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[9] A. Bemporad, M. Morari, and N.L. Ricker. *Model Predictive Control Toolbox*, 2005.

[10] A.B. Berkelaar, K. Roos, and T. Terkaly. *Recent Advances in Sensitivity Analysis and Parametric Programming*, chapter 6: The Optimal Set and Optimal Partition Approach to Linear and Quadratic Programming. Kluwer Publishers, Dordrecht, 1997.

[11] M.J. Best. *Applied Mathematics and Parallel Computing*, chapter An Algorithm for the Solution of the Parametric Quadratic Programming Problem, pages 57–76. Physica-Verlag, Heidelberg, 1996.

[12] J.T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, Philadelphia, 2001.

[13] R.R. Bitmead, M. Gevers, and V. Wertz. *Adaptive optimal control: the thinking man's GPC*. Prentice Hall, Sydney, 1990.

[14] H.G. Bock, M. Diehl, D.B. Leineweber, and J.P. Schlöder. Efficient direct multiple shooting in nonlinear model predictive control. In F. Keil, W. Mackens, H. Voß, and J. Werther, editors, *Scientific Computing in Chemical Engineering II*, volume 2, pages 218–227, Berlin, 1999. Springer.

[15] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.

[16] N. L. Boland. A dual-active-set algorithm for positive semi-definite quadratic programming. *Mathematical Programming*, 78:1–27, 1997.

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

[18] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, London, 2004.

[19] A.M. Cervantes, S. Tonelli, A. Brandolin, J.A. Bandoni, and L.T. Biegler. Large-scale dynamic optimization for grade transitions in a low density polyethylene plant. *Computers and Chemical Engineering*, 26(2):227–237, 2002.

[20] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.

[21] J.W. Daniel, W.B. Gragg, L. Kaufman, and G.W. Steward. Reorthogonalization and stable algorithms for updating the gram-schmidt QR factorization. *Mathemathics of Computation*, 30(136):772–795, 1976.

[22] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[23] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Universität Heidelberg, 2001. http://www.ub.uni-heidelberg.de/archiv/1659/.

[24] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[25] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H.G. Bock, E.D. Gilles, and J.P. Schlöder. An efficient algorithm for nonlinear model predictive control of large-scale systems. Part I: Description of the method. *Automatisierungstechnik*, 50(12):557–567, 2002.

[26] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H.G. Bock, E.D. Gilles, and J.P. Schlöder. An efficient algorithm for nonlinear model predictive control of large-scale systems. Part II: Application to a distillation column. *Automatisierungstechnik*, 51(1):22–29, 2003.

[27] W.S. Dorn. Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18:155–162, 1960.

[28] dSPACE. Homepage. http://www.dspace.com, 2006.

[29] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control*. (submitted).

[30] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy for fast parametric quadratic programming in MPC applications. In *Proceedings of the IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, Grenoble*, 2006.

[31] H.J. Ferreau, G. Lorini, and M. Diehl. Fast nonlinear model predictive control of gasoline engines. In *Proceedings of the IEEE International Conference on Control Applications, Munich*, pages 2754–2759, 2006.

[32] A.V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming*. Academic Press, New York, 1983.

[33] R. Fletcher. A general quadratic programming algorithm. *J. Inst. Math. Appl.*, 7:76–91, 1971.

[34] R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, 2nd edition, 1987.

[35] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[36] P.E. Gill, G.H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, 1974.

[37] P.E. Gill, N.I.M. Gould, W. Murray, M.A. Saunders, and M.H. Wright. A weighted gram-schmidt method for convex quadratic programming. *Mathematical Programming*, 30:176–195, 1984.

[38] P.E. Gill and W. Murray. Numerically stable methods for quadratic programming. *Mathematical Programming*, 14:349–372, 1978.

[39] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Transactions on Mathematical Software*, 10(3):282–298, 1984.

[40] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. Inertia-controlling methods for general quadratic programming. *SIAM Review*, 33(1):1–36, 1991.

[41] P.E. Gill, W. Murray, and M.H. Wright. *Numerical Linear Algebra and Optimization*, volume 1. Addison-Wesley, New York, 1991.

[42] P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization*. Academic Press, London, 1999.

[43] W.J. Givens. Numerical computation of the characteristic values of a real symmetric matrix. Technical Report 1574, Oak Ridge National Laboratory, 1954.

[44] D. Goldfarb. Matrix factorizations in optimization of nonlinear functions subject to linear constraints. *Mathematical Programming*, 10:1–31, 1975.

[45] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33, 1983.

[46] G.H. Golub and C.F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[47] S. Hammarling. A note on modifications to the givens plane rotation. *J. Inst. Maths Applics*, 13:215–218, 1974.

[48] R.J. Hanson and T. Hopkins. Algorithm 830: Another visit with standard and modified Givens transformations and a remark on Algorithm 539. *ACM Transactions on Mathematical Software*, 30(1):86–94, 2004.

[49] M.R. Hestenes. *Calculus of variations and optimal control theory*. Wiley, New York, 1966.

[50] A.U. Idnani. *Numerically stable dual projection methods for solving positive definite quadratic programs*. PhD thesis, City College of New York, 1980.

[51] T.A. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Trans. Automatic Control*, 48:810–815, 2003.

[52] Merten Jung. *Mean-Value Modelling and Robust Control of the Airpath of a Turbocharged Diesel Engine*. PhD thesis, University of Cambridge, 2003.

[53] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

[54] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[55] 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. *Journal of Optimization Theory and Applications*, 57(2):265–293, 1988.

[56] V. Klee and G.J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, 1972.

[57] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, 1951. University of California Press.

[58] L. Ljung. *System identification*. Prentice Hall, Upper Saddle River, N.J., 1999.

[59] The MathWorks. Homepage. http://www.mathworks.com/, 2006.

[60] D. Q. Mayne and S. Rakovic. Optimal control of constrained piecewise affine discrete-time systems. *Computational Optimization and Applications*, 25:167–191, 2003.

[61] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 26(6):789–814, 2000.

[62] NAG. *Fortran Library Routine Document E04NAF*, 1991.

[63] NAG. *Fortran Library Routine Document E04NFF/E04NFA*, 1999.

[64] G. De Nicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear time varying systems. *IEEE Transactions on Automatic Control*, AC-43(7):1030–1036, 1998.

[65] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, Heidelberg, 1999.

[66] P. Ortner. MPC for a diesel engine airpath using an explicit approach for constraint systems. Master's thesis, Institut für Design und Regelung mechatronischer Systeme, Universität Linz, 2005.

[67] P. Ortner, P. Langthaler, J.V.G. Ortiz, and L. del Re. MPC for a diesel engine air path using an explicit approach for constraint systems. In *Proceedings of the IEEE International Conference on Control Applications, Munich*, pages 2760–2765, 2006.

[68] G. Pannocchia, J.B. Rawlings, and S.J. Wright. The partial enumeration method for model predictive control: Algorithm and examples. Technical Report 2006-01, Texas-Wisconsin Modeling and Control Consortium, 2006.

[69] S. Piche, B. Sayyar-Rodsari, D. Johnson, and M. Gerules. Nonlinear model predictive control using neural networks. *IEEE Control Systems Magazine*, 20:53–62, 2000.

[70] L.S. Pontryagin, V.G. Boltyanski, R.V. Gamkrelidze, and E.F. Miscenko. *The Mathematical Theory of Optimal Processes*. Wiley, Chichester, 1962.

[71] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Numerical Analysis, Dundee 1977*, volume 630 of *Lecture Notes in Mathematics*, Berlin, 1978. Springer.

[72] S.J. Qin and T.A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[73] C.V. Rao, S.J. Wright, and J.B. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.

[74] P.O.M. Scokaert and J.B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1169, 1998.

[75] J. Spjøtvold, E.C. Kerrigan, C.N. Jones, T.A. Johansen, and P. Tøndel. Conjectures on an algorithm for convex parametric quadratic programs. Technical report, Department of Engineering, University of Cambridge, 2004.

[76] M. Sznaier and M.J. Damborg. Suboptimal control of linear systems with state and control inequality constraints. In *Proceedings of the 26th IEEE conference on decision and control, Los Angeles*, pages 761–762, 1987.

[77] P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit mpc solutions. *Automatica*, 39:489–497, 2003.

[78] P. Tøndel, T.A. Johansen, and A. Bemporad. Computation and approximation of piecewise affine control laws via binary search trees. *Automatica*, 39:945–950, 2003.

[79] T.H. Tsang, D.M. Himmelblau, and T.F. Edgar. Optimal control via collocation and non-linear programming. *International Journal on Control*, 21:763–768, 1975.

[80] D. van Heesch. Doxygen homepage. http://www.doxygen.org.

[81] R. J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11:451–484, 1999.

[82] A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, 2002.

[83] X. Wang. Resolution of ties in parametric quadratic programming. Master's thesis, University of Waterloo, Ontario, Canada, 2004.

[84] A.G. Wills, D. Bates, A.J. Fleming, B. Ninness, and S.O.R. Moheimani. Application of MPC to an active structure using sampling rates up to 25kHz. 44th IEEE Conference on Decision and Control and European Control Conference ECC'05, Seville, 2005.

[85] R.B. Wilson. *A simplicial algorithm for concave programming*. PhD thesis, Harvard University, 1963.

[86] L. Wirsching. An SQP algorithm with inexact derivatives for a direct multiple shooting method for optimal control problems. Master's thesis, University of Heidelberg, 2006.

[87] L. Wirsching, H. G. Bock, and M. Diehl. Fast NMPC of a chain of masses connected by springs. In *Proceedings of the IEEE International Conference on Control Applications, Munich*, pages 591–596, 2006.

[88] P. Wolfe. The simplex method for quadratic programming. *Econometrica*, 27:382–398, 1959.

[89] P. Wolfe. A duality theorem for non-linear programming. *Quarterly of Applied Mathematics*, 19:239–244, 1961.

[90] W.M. Wonham. *Linear Multivariable Control: a Geometric Approach*. Springer, Heidelberg, 1979.

[91] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, Philadelphia, 1997.

[92] E. Zafiriou. Robust model predictive control of processes with hard constraints. *Computers & Chemical Engineering*, 14(4–5):359–371, 1990.

Index