

Lecture Summary

Model Predictive Control

Julius Schlapbach

October 1, 2022

Contents

Disclaimer	4
1 Introduction	5
1.1 Definition of Model Predictive Control	5
1.2 Concepts of MPC	5
1.3 Important Aspects of Model Predictive Control	7
2 System Theory Basics	8
2.1 Models of Dynamic Systems	8
2.2 Analysis of LTI Discrete-Time Systems	10
2.3 Analysis of Nonlinear Discrete-Time Systems	11
3 Unconstrained Linear Quadratic Optimal Control	14
3.1 Optimal Control	14
3.2 Receding Horizon	17
3.3 Infinite Horizon LQR	19
4 Introduction to Convex Optimization	20
4.1 Main Concepts	20
4.2 Convex Sets	21
4.3 Convex Functions	22
4.4 Level and Sublevel Sets	23
4.5 Examples of Convex Functions	23
4.6 Convex Optimization Problems	24
4.7 Duality	26
4.8 Optimality / KKT Conditions	28
4.9 Sensitivity Analysis	29
5 Constrained Finite Time Optimal Control	30
5.1 Constrained Linear Optimal Control	30
5.2 Constrained Optimal Control: Quadratic Cost	31
5.3 Constrained Optimal Control: 1- and ∞ -Norm Cost	33
5.4 Comparison Quadratic and Linear Cost Functions	35
5.5 Receding Horizon Control Notation	35
6 Invariance	36
6.1 Objectives of Constrained Control	36
6.2 Invariance	36
6.3 Control Invariance	37
6.4 Invariant Sets and MPC	38
6.5 Practical Computation of Invariant Sets	39
7 Feasibility and Stability	42
7.1 Key Points of MPC	42

7.2	Loss of Feasibility and Stability in MPC	43
7.3	Feasibility and Stability Guarantees in MPC	43
8	Practical Issues in MPC	45
8.1	Common Tasks for MPC	45
8.2	Reference Tracking	45
8.3	Enlarging the Feasible Set	49
9	Robust MPC	54
9.1	Uncertainty Models	54
9.2	Impact of Bounded Additive Noise	55
9.3	Robust Open-Loop MPC	59
9.4	Closed-Loop Predictions	59
9.5	Tube-MPC	60
9.6	Implementation of Tube-MPC	64
9.7	Robust Constraint Tightening MPC	65
9.8	Robustness of Nominal MPC	66
10	Implementation & Remarks	67
10.1	Explicit Solution	67
10.2	Iterative Optimization Methods	68
10.3	Constrained Minimization	69

Disclaimer

Beim nachfolgenden Dokument handelt es sich um eine möglichst vollständige Übersicht mit Erklärungen der Vorlesung ‘Model Predictive Control’, gehalten durch Prof. M. Zeilinger im FS22 und enthält unter Umständen auch Grafiken oder Ausschnitte aus den Vorlesungsunterlagen derselben. Das Dokument darf nur für die Nutzung im Rahmen des Studiums des entsprechenden Fachs an der ETH Zürich eingesetzt werden und nicht an Dritte ausserhalb der ETH Zürich weitergegeben werden ohne entsprechende Nachfrage.

Es kann keine Garantie für die Vollständigkeit oder Korrektheit des Inhaltes gegeben werden. Um Tippfehler, inhaltliche Fehler, etc. verbessern zu können, wäre ich für eine Meldung derselben an juliusc@student.ethz.ch dankbar.

The following document serves as an overview of the lecture ‘Model Predictive Control’, taught by Prof. M. Zeilinger in FS22 and might therefore also contain graphics or extracts from lecture notes. The use of this document is limited to the studies of the corresponding subject at ETH Zurich and may not be used by other parties outside of ETH Zurich without permission of the author.

I can give no guarantee regarding completeness or correctness of the content. In order to correct typos, mistakes with regards to content and further improve the summary, please report such issues to juliusc@student.ethz.ch if found.

1 Introduction

1.1 Definition of Model Predictive Control

In classical control systems, some reference that should be tracked and the feedback measurements from the plant are fed into the controller, which then computes an input signal to steer the plant. Model predictive control on the other hand replaces this controller with an optimization based algorithm to achieve better solutions and to be able to state objectives and constraints in a more intuitive manner (see figure 1.1).

Such a control model could for example be used to steer a car as fast as possible over a race track, while not hitting other cars, staying on the road, limit acceleration, etc. (constraints). The objective in that case would be to minimize time, while satisfying all stated constraints, resulting in the mentioned optimization problem. In order to adapt to new track conditions / obstacles, feedback has to be introduced and the optimization problem is solved again for a finite time horizon, starting at the current iteration. The resulting input sequence is however only applied for a single timestep (this receding horizon formulation introduced feedback; figure 1.1).

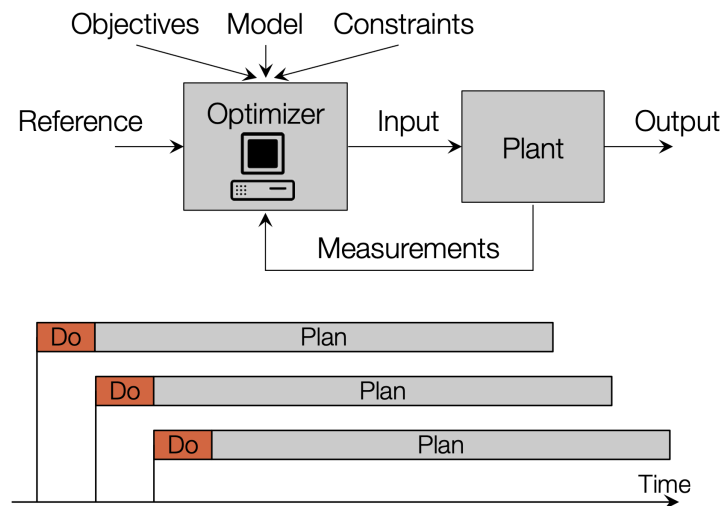


Figure 1.1: Control Diagram for Model Predictive Control (lecture slides)

1.2 Concepts of MPC

While MPC is very useful for planning purposes, it is important to note that it is usually applied on a higher level than classical control (where the controller itself is designed) and does not replace the latter on a lower level. In contrast to classical control, MPC does not result in a controller design, but an optimization problem to determine the optimal input $u(t)$ is repeatedly solved in real-time, allowing to address control constraints (limits) and process constraints (safety) very nicely and in *time domain*. Classical control methods are usually designed in *frequency domain* and address the issues of disturbance rejection (disturbance added to the output), noise insensitivity (noise added to output measurements) and model uncertainty. Figure 1.2 illustrates this.

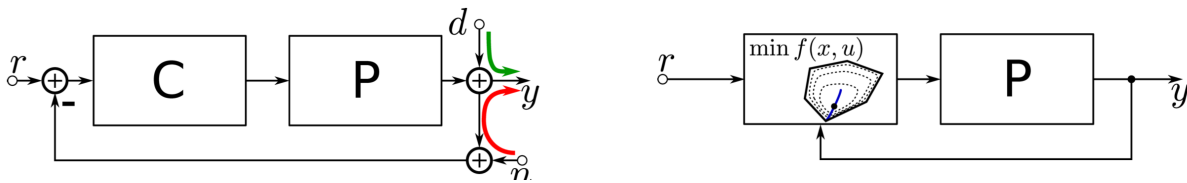


Figure 1.2: Comparison of Classical Control and MPC (lecture slides)

Examples of constraints that are usually addressed by model predictive control include:

- Physical constraints (e.g. actuator limits)
- Performance constraints (e.g. overshooting)
- Safety constraints (e.g. temperature or pressure limits)

Classical control methods are often able to satisfy such constraints approximately in an ad hoc manner, if the set points are sufficiently far away from the constraints. As optimal operating points are often close to constraints, classical control leads to a suboptimal plant operation. Model Predictive Control includes these constraints in the design process, ensuring that they are not violated and a set point optimizing the cost function can be used to achieve an optimal plant operation (improved overall performance). This is also illustrated in figure 1.3, comparing system trajectories for classical control and MPC.

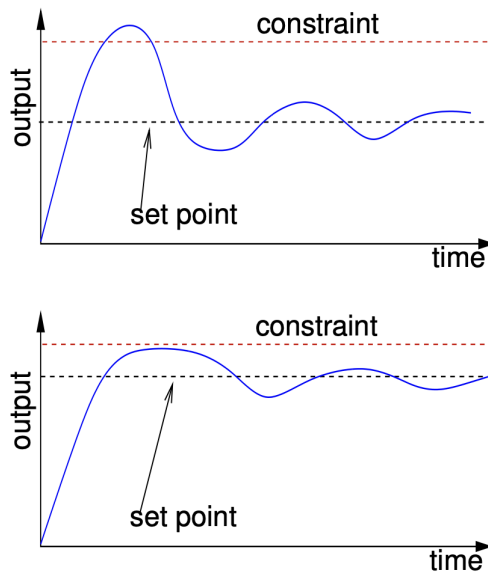


Figure 1.3: Performance of Classical vs. Predictive Control (lecture slides)

The mathematical formulation of model predictive control is:

$$\begin{aligned}
 U_k^*(x(k)) &:= \arg \min_{U_k} \sum_{i=0}^{N-1} I(x_{k+i}, u_{k+i}) && (1.1) \\
 \text{s.t. } &x_k = x(k) && \text{measurement} \\
 &x_{k+i+1} = Ax_{k+i} + Bu_{k+i} && \text{system model} \\
 &x_{k+i} \in \mathcal{X} && \text{state constraints} \\
 &u_{k+i} \in \mathcal{U} && \text{input constraints} \\
 &U_k = \{u_k, u_{k+1}, \dots, u_{k+N-1}\} && \text{optimization variables}
 \end{aligned}$$

At each sample time, the state $x(k)$ is measured or estimated, the optimal control sequence corresponding to the mathematical formulation above is computed $U_k^* = \{u_k^*, u_{k+1}^*, \dots, u_{k+N-1}^*\}$ and the only first control action u_k^* is implemented on the system. These steps are then repeated for each iteration and lead to the control sequence for the system.

Applications of MPC include (details can be found in the lecture slides):

- Autonomous Race Cars
- Path Following
- Autonomous Quadcopter Flight
- Energy Efficient Building Control
- Kite Power
- Automotive Systems

1.3 Important Aspects of Model Predictive Control

While model predictive control is a great alternative to classical solutions when it comes to high level controllers, where the handling of constraints as well as the performance of the controller are very important for optimality, there are some challenges:

- *Implementation*: The optimization problem of MPC has to be solved in real-time with the available resources and within a single sampling interval of the system.
- *Feasibility*: The optimization problem may not be feasible for all future time steps as some constraints (e.g. unexpected obstacles, etc.) might not be satisfiable.
- *Stability*: Closed-loop stability and convergence are not guaranteed when solving the MPC problem.
- *Robustness*: The solution to MPC for the closed-loop system might not be robust to uncertainties and/or disturbances.

2 System Theory Basics

In order to apply model predictive control to a system, a model of the system as well as a state estimator are required, before the optimal control and optimization problems can be set up, solved and implemented. This step is again followed by some verification steps to ensure the desired behavior. In this chapter, the most important basics of system theory are revisited, as they are important for upcoming MPC approaches.

2.1 Models of Dynamic Systems

For classical controller design, the description of the system using a transfer functions $G(s) = \frac{Y(s)}{U(s)}$ is an essential part. While this allows to use an abundance of tools for the analysis of the linear system closed-loop behavior, they usually do not extend to nonlinear systems (e.g. constrained systems) and can not be used in the time domain.

Nonlinear Time-Invariant Continuous-Time State Space Models

Nonlinear time-invariant continuous-time state space models consist of a state vector $x \in \mathbb{R}^n$, an input vector $u \in \mathbb{R}^m$, an output vector $y \in \mathbb{R}^p$ as well as the system dynamics $g(x, u)$ and output function $h(x, u)$:

$$\dot{x} = g(x, u) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n \quad (2.1)$$

$$y = h(x, u) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p \quad (2.2)$$

These systems are very general and even higher order ODEs can be brought into this form by augmenting the state vector with the derivatives $x_{j+1} = \dot{x}^{(j)}$ for $j = 0, \dots, n-1$. The system dynamics then contain a chain of integrators in addition to the nonlinear dynamics. Analysis and control synthesis of such systems is usually very hard, wherefore the systems are often simplified through linearization and similar methods.

LTI Continuous-Time State Space Models

LTI continuous-time systems can be written with a state space model, consisting of the system matrix $A^c \in \mathbb{R}^{n \times n}$, input matrix $B^c \in \mathbb{R}^{n \times m}$, output matrix $C \in \mathbb{R}^{p \times n}$ and throughput matrix $D \in \mathbb{R}^{p \times m}$ (the superscript c denotes the continuous nature of the system):

$$\dot{x} = A^c x + B^c u \quad (2.3)$$

$$y = Cx + Du \quad (2.4)$$

While there are a lot of tools for these systems, they are especially helpful as there exists a closed-form solution for the prediction of the state x when starting from some initial state $x_0 := x(t_0)$ (only precise in the noiseless case):

$$x(t) = e^{A^c(t-t_0)}x_0 + \int_{t_0}^t e^{A^c(t-\tau)}B^c u(\tau) d\tau \quad (2.5)$$

Most nonlinear systems can be linearized around their operating point to simplify the control design (constraints can be used to force the system to stay close to the operating point), using a first order Taylor expansion around \bar{x} (with $\partial f/\partial x^T$ being the Jacobian matrix):

$$f(x) \approx f(\bar{x}) + \left. \frac{\partial f}{\partial x^T} \right|_{x=\bar{x}} (x - \bar{x}) \quad (2.6)$$

Linearization around a stationary operating point x_s, u_s with $\dot{x}_s = g(x_s, u_s) = 0$ and $y_s = h(x_s, u_s)$ results in the following system dynamics (usually Δx , Δu and Δy are just abbreviated with x , u and y):

$$\dot{x} = g(x_s, u_s) + \left. \frac{\partial g}{\partial x^T} \right|_{\substack{x=x_s \\ u=u_s}} (x - x_s) + \left. \frac{\partial g}{\partial u^T} \right|_{\substack{x=x_s \\ u=u_s}} (u - u_s) \quad (2.7)$$

$$\Delta \dot{x} = A^c \Delta x + B^c \Delta u \quad (2.8)$$

$$y = h(x_s, u_s) + \left. \frac{\partial h}{\partial x^T} \right|_{\substack{x=x_s \\ u=u_s}} (x - x_s) + \left. \frac{\partial h}{\partial u^T} \right|_{\substack{x=x_s \\ u=u_s}} (u - u_s) \quad (2.9)$$

$$\Delta y = C \Delta x + D \Delta u \quad (2.10)$$

These linearizations simplify controller design and analysis, but they are only valid within a small neighborhood of the linearization point. Further away, the linearized system might diverge from the real one.

Time-Invariant Discrete-Time State Space Models

As discrete-time systems are only defined at discrete instances in time, they are described by difference equations instead of differential equations (with the same dimensions as before):

$$x(k+1) = g(x(k), u(k)) \quad (2.11)$$

$$y(k) = h(x(k), u(k)) \quad (2.12)$$

Discrete-time systems are often continuous-time systems that have been discretized with some sampling time T_s to take into account the finite available computation time. They are also better suited for MPC in general. Usually, it is assumed that the input is held constant up to the next sampling instance $u(t) = u(t_k)$, $t \in [t_k, t_{k+1})$.

Euler discretization

Euler discretization is based on the numerical Euler method and transforms a continuous time system of the following form into a discrete time model (below), using the approximation $\dot{x}^c(t) \approx \frac{x^c(t+T_s) - x^c(t)}{T_s}$.

$$\dot{x}^c(t) = g^c(x^c(t), u^c(t)) \quad (2.13)$$

$$y^c(t) = h^c(x^c(t), u^c(t)) \quad (2.14)$$

Discrete-time model using $x(k) := x^c(t_0 + kT_s)$ and $u(k) := u^c(t_0 + kT_s)$:

$$x(k+1) = x(k) + T_s g^c(x(k), u(k)) = g(x(k), u(k)) \quad (2.15)$$

$$y(k+1) = h^c(x(k), u(k)) = h(x(k), u(k)) \quad (2.16)$$

Under regularity assumptions, the state trajectories of the continuous-time and discrete-time systems look very similar for the same input and small enough T_s (Euler discretization approaches the exact discretization).

The discretization of continuous-time systems with a matrix description is even simpler:

$$\dot{x}(t) = A^c x(t) + B^c u(t) \quad (2.17)$$

$$y(t) = C^c x(t) + D^c u(t) \quad (2.18)$$

The corresponding discretized system is:

$$x(k+1) = (I + T_s A^c)x(k) + T_s B^c u(k) = Ax(k) + Bu(k) \quad (2.19)$$

$$y(k) = C^c x(k) + D^c u(k) = Cx(k) + Du(k) \quad (2.20)$$

Note that continuous-time stability does not imply discrete-time stability for systems that were discretized using the Euler discretization.

Exact discretization

Exact discretization is based on the explicit ODE solution given by equation 2.5, where the choice $t_0 = t_k$ and $t = t_{k+1} = t_k + T_s$ yield the representation (not available for general nonlinear systems → use Euler discretization):

$$x(t_{k+1}) = e^{A^c T_s} x(t_k) + \int_0^{T_s} e^{A^c(T_s - \tau')} B^c d\tau' u(t_k) \quad (2.21)$$

$$= Ax(t_k) + Bu(t_k) \quad (2.22)$$

Under assumption of a constant input signal $u(t)$ for the sampling intervals, the solution of the discretized system exactly coincides with the continuous-time system at sampling instances. Additionally, $B = (A^c)^{-1}(A - I)B^c$ holds if A^c is invertible and the discrete-time system is stable if the continuous-time system is stable.

Solution of Linear Discrete-Time Systems

The exact solution to a linear discrete-time system of the form $x(k+1) = Ax(k) + Bu(k)$ is a linear function of the initial state and the input sequence:

$$x(k+N) = A^N x(k) + \sum_{i=0}^{N-1} A^i Bu(k+N-1-i) \quad (2.23)$$

2.2 Analysis of LTI Discrete-Time Systems

Coordinate Transformations

For a discrete-time system of the following form, there are infinitely many choices for the state and the same input-output behavior, as the output is determined by the initial state and the inputs only:

$$x(k+1) = Ax(k) + Bu(k) \quad (2.24)$$

$$y(k) = Cx(k) + Du(k) \quad (2.25)$$

Some coordinate transformations T might simplify the system analysis (with unchanged input and output $u(k)$ and $y(k)$):

$$\tilde{x}(k+1) = TAT^{-1}\tilde{x}(k) + TBu(k) = \tilde{A}\tilde{x}(k) + \tilde{B}u(k) \quad (2.26)$$

$$y(k) = CT^{-1}\tilde{x}(k) + Du(k) = \tilde{C}\tilde{x}(k) + \tilde{D}u(k) \quad (2.27)$$

Stability

An autonomous LTI system with $x(k+1) = Ax(k)$ is globally asymptotically stable if:

$$\lim_{k \rightarrow \infty} x(k) = 0, \quad \forall x(0) \in \mathbb{R}^n \quad (2.28)$$

This is fulfilled if and only if the absolute value of all eigenvalues λ_i of A are strictly smaller than one $|\lambda_i| < 1$ (as the system works discrete-time).

Observability & Controllability

A system $x(k+1) = Ax(k) + Bu(k)$ is called *controllable* if for any pair for states $(x(0), x^*)$ there exists a finite time N and a control sequence $\{u(0), \dots, u(N-1)\}$ such that $x(N) = x^*$. Using the Cayley Hamilton theorem, A^k can be expressed as a linear combination of lower order matrix exponentials A^j as long as $k \geq n$, what yields the *controllability matrix*:

$$\mathcal{C} = [B \quad AB \quad \dots \quad A^{n-1}B] \quad (2.29)$$

The LTI system is then controllable if and only if $\text{rank}(\mathcal{C}) = n$ holds. Similarly, the system is *stabilizable* (weaker and implied by controllability) if all uncontrollable modes are stable so that the system can still be steered to the origin. The necessary and sufficient condition for this is:

$$\text{rank}([\lambda_j I - A \mid B]) = n, \quad \forall \lambda_j \in \Lambda_A^+ \Leftrightarrow |\lambda_j| > 1 \quad (2.30)$$

A system with zero input of the form $x(k+1) = Ax(k)$ and $y(k) = Cx(k)$ is said to be *observable* if there exists a finite N such that for every $x(0)$ the measurements $y(0), y(1), \dots, y(N-1)$ allow to uniquely determine the initial state $x(0)$. The necessary and sufficient condition for observability of the system (A, C) is then given by the observability matrix \mathcal{O} having full rank ($\text{rank}(\mathcal{O}) = n$):

$$\mathcal{O} = [C^T \quad (CA)^T \quad \dots \quad (CA^{n-1})^T]^T \quad (2.31)$$

If some modes are not observable but asymptotically stable, the system is said to be *detectable* (the modes that are not observable converge to the origin). The condition for this is given by:

$$\text{rank}([A^T - \lambda_j I \mid C^T]) = n, \quad \forall \lambda_j \in \Lambda_A^+ \quad (2.32)$$

2.3 Analysis of Nonlinear Discrete-Time Systems

Stability for Nonlinear Systems

Stability in the sense of Lyapunov (referred to as common stability in this lecture) can be defined for some nonlinear, time-invariant, discrete-time system of the following form, which possibly already describes the closed-loop system dynamics:

$$x(k+1) = g(x(k)) \quad (2.33)$$

The system is stable around an equilibrium point \bar{x} , fulfilling $g(\bar{x}) = \bar{x}$ if it stays in an arbitrarily small neighborhood of the equilibrium when disturbed slightly. For LTI systems this stability criterion can be further simplified (discussed later in this chapter).

Formally, a system is called *stable in the sense of Lyapunov* around some equilibrium point \bar{x} if there exists some $\delta(\epsilon)$ for each $\epsilon > 0$ such that the system stays within radius ϵ around \bar{x} when starting at maximum distance δ from \bar{x} (and unstable otherwise):

$$\|x(0) - \bar{x}\| < \delta(\epsilon) \rightarrow \|x(k) - \bar{x}\| < \epsilon, \quad \forall k \geq 0 \quad (2.34)$$

An equilibrium point is called *globally asymptotically stable* if it is stable and attractive, i.e. the system always converges to the equilibrium point over time independent of the initial condition $x(0)$:

$$\lim_{k \rightarrow \infty} \|x(k) - \bar{x}\| = 0, \quad \forall x(0) \in \mathbb{R}^n \quad (2.35)$$

In case this only holds for points within some neighborhood $R > 0$ around the equilibrium point \bar{x} , the system is said to be *locally asymptotically stable*:

$$\lim_{k \rightarrow \infty} \|x(k) - \bar{x}\| = 0, \quad \forall x(0) \in \{x(0) \mid \|x(0) - \bar{x}\| < R\} \quad (2.36)$$

Lyapunov Direct Method

Stability can be proven by constructing a Lyapunov function. However, it is important to mention that this condition is only sufficient, but not necessary. I.e. if some candidate function $V(x)$ turns out not to fulfill the criteria for a Lyapunov function, one cannot say that the system is not stable.

Such a *global Lyapunov function* for a system $x(k+1) = g(x(k))$ with equilibrium point $\bar{x} = 0$ is a function $V : \mathbb{R}^n \rightarrow \mathbb{R}$ that is continuous at the origin and finite for all $x \in \mathbb{R}^n$, satisfying the following conditions (usually the third conditions has to be proven for some nonlinear system):

1. $\|x\| \rightarrow \infty \implies V(x) \rightarrow \infty$ (radially unbounded)
2. $V(0) = 0$ and $V(x) > 0, \forall x \in \mathbb{R}^n \setminus \{0\}$
3. $V(g(x)) - V(x) \leq -\alpha(x), \forall x \in \mathbb{R}^n$ for some continuous positive definite $\alpha : \mathbb{R}^n \rightarrow \mathbb{R}$

The existence of such a Lyapunov function $V(x)$ implies global asymptotic stability for the system around $x = 0$ according to the Lyapunov theorem. If a function can be found such that $\alpha(x)$ is positive semidefinite (but not positive definite), the system is globally Lyapunov stable around $x = 0$ (i.e. it does not diverge). Often, quadratic functions as $V(x) = x^2$ are good initial guesses for Lyapunov functions.

Lyapunov Indirect Method

The indirect method states that when considering the linearization matrix $A = \frac{\partial g}{\partial x^T} \Big|_{x=0}$ around the equilibrium point $\bar{x} = 0$, the system is locally asymptotically stable if all eigenvalues of A have absolute value less than one ($|\lambda_i| < 1$). If there exists some eigenvalue with $|\lambda_i| > 1$, the equilibrium \bar{x} is unstable.

If there exist one or multiple eigenvalues with $|\lambda_i| = 1$ (but all remaining eigenvalues are stable), the indirect method does not yield any conclusion and a Lyapunov function (direct method) has to be created. Considering continuous-time systems instead of discrete ones, the condition for stability changes to $\text{Re}(\lambda_i) < 0$ for all eigenvalues for stability and $\text{Re}(\lambda_i) > 0$ for some eigenvalue for an unstable system.

Lyapunov Stability of LTI Discrete-Time Systems

For linear and discrete-time systems, the following notation is used (which might already represent a closed loop system with $A = A' + B'k$):

$$x(k+1) = Ax(k) \quad (2.37)$$

Choosing the candidate function $V(x) = x^T Px$ with some positive definite matrix $P > 0$ already fulfills the first two conditions of the Lyapunov direct method due to its quadratic nature and the third one is equivalent to:

$$V(Ax(k)) - V(x(k)) = x^T(k)A^T PAx(k) - x^T(k)Px(k) = x^T(A^T PA - P)x(k) \leq -\alpha(x(k)) \quad (2.38)$$

As the function $\alpha(x)$ has to be positive definite, it can be chosen to be $\alpha(x) = x^T Qx(k)$ with some positive definite matrix $Q > 0$. The third condition for global asymptotic stability given by the Lyapunov theorem and can now be reduced to the task of finding some $P > 0$, which solves the *discrete-time Lyapunov equation* for some given Q (constructive result for a Lyapunov function - they do not have to be guessed):

$$A^T PA - P = -Q, \quad Q > 0 \quad (2.39)$$

In connection with this equation 2.39, a theorem states that it has a unique solution $P > 0$ if and only if all eigenvalues of A lie inside the unit circle, what corresponds to stability for the system $x(k+1) = Ax(k)$. In the case of linear systems *stability is always global* and the condition for global Lyapunov stability is not only sufficient but also necessary.

The matrix P that solves the discrete-time Lyapunov equation 2.39 can also be used to determine the *infinite horizon cost-to-go* for an asymptotically stable system $x(k+1) = Ax(k)$ with a quadratic cost function determined by Q :

$$\Psi(x(0)) = \sum_{k=0}^{\infty} x(k)^T Qx(k) = \sum_{k=0}^{\infty} x(0)^T (A^k)^T Q A^k x(0) = x(0)^T Px(0) \quad (2.40)$$

3 Unconstrained Linear Quadratic Optimal Control

3.1 Optimal Control

Notation

As most problems in optimal and model predictive control contain both real state trajectories and predictions of the state, distinct notation is very important. In this course, the real state trajectory given by the correct system dynamics $x(k+1) = g(x(k))$ is denoted with the bracket notation $x(k)$, while the prediction of the same state based on the model is denoted by x_k . For the initial condition, the real state value and the prediction usually coincide $x(0) = x_0$.

Optimal Control Problem

The optimal control problem is to find an optimal input (in discrete-time an input sequence $U := [u_1^T, u_2^T, \dots]^T$) over a finite or infinite horizon, minimizing some given cost function and the systems initial state $x(0)$. The *cost* is usually composed of some stage cost $l(x_i, u_i)$ and a terminal cost $l_f(x_N)$ in the case of finite horizon problems of length N :

$$J(x_0, U) := l_f(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i) \quad (3.1)$$

The terminal cost in finite horizon problems with N steps is mainly required to compensate for the cost going forward from step N that is missed. The larger N , the smaller this effect is for a converging system. In addition to the minimization of the cost function, the state predictions $\{x_i\}_{i=0}^N$ must satisfy the model of the system dynamics:

$$x_{i+1} = g(x_i, u_i), \quad i = 0, \dots, N-1, \quad x(0) = x_0 \quad (3.2)$$

There might be some additional constraints on states and/or inputs:

$$h(x_i, u_i) \leq 0, \quad i = 0, \dots, N-1 \quad (3.3)$$

In finite horizon problems, there often is a constraint on the terminal state:

$$x_N \in \mathcal{X}_f \quad (3.4)$$

The general finite horizon optimal control problem for a discrete-time system is then given by (where $J^*(x(0))$ denotes the optimal cost and U the vector of stacked inputs):

$$\begin{aligned} J^*(x(0)) &:= \min_U J(x(0), U) & (3.5) \\ \text{s.t. } x_{i+1} &= g(x_i, u_i), \quad i = 0, \dots, N-1 \\ h(x_i, u_i) &\leq 0, \quad i = 0, \dots, N-1 \\ x_N &\in \mathcal{X}_f \\ x_0 &= x(0) \end{aligned}$$

Linear Quadratic Optimal Control

This section only considers linear discrete-time time-invariant systems of the following form:

$$x(k+1) = Ax(k) + Bu(k) \quad (3.6)$$

With the objective to minimize the following quadratic cost function through an optimal input sequence U (without input or state constraints) over some finite horizon of length N :

$$J(x_0, U) := x_N^T P x_N + \sum_{i=0}^{N-1} (x_i^T Q x_i + u_i^T R u_i) \quad (3.7)$$

The matrices P , Q and R are given as:

- *terminal weight* given by $P \geq 0$ with $P = P^T$ (positive semi-definite and symmetric matrix)
- *state weight* given by $Q \geq 0$ with $Q = Q^T$ (positive semi-definite and symmetric matrix)
- *input weight* given by $R > 0$ with $R = R^T$ (the input weight is usually chosen to be strictly positive definite and symmetric in order not to miss any input component)

The initial state $x(0) = x_0$ is assumed to be given as the current state, while x_1, \dots, x_N and u_0, \dots, u_{N-1} are predictions and optimization variables that have to satisfy the system dynamics.

Batch Approach to the Optimal Control Problem

The batch approach is based on expressing all future state predictions x_i in terms of the initial condition x_0 and the inputs u_0, \dots, u_{N-1} . Using the dynamics given by equation 3.6 yields the matrix form:

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} 0 & \dots & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \dots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix} \quad (3.8)$$

$$= S^x x(0) + S^u U \quad (3.9)$$

The matrices $\bar{Q} := \text{diag}(Q, \dots, Q, P)$ and $\bar{R} := \text{diag}(R, \dots, R)$ are used to reformulate stage and terminal cost in matrix form:

$$J(x(0), U) = X^T \bar{Q} X + U^T \bar{R} U \quad (3.10)$$

Using the state dynamics in matrix form given by equation 3.9 and substituting X in the finite horizon cost function in equation 3.10, results in a function $J(x(0), U)$ that is obviously positive definite and quadratic in U (dependency on x_i was eliminated). As such functions attain one unique optimum, the optimal input U^* can be found by setting the gradient of the cost function with respect to U to zero:

$$\nabla_U J(x(0), U) = 0 \quad (3.11)$$

The optimal control U for some given initial conditions is:

$$U^*(x(0)) = -((S^u)^T \bar{Q} S^u + \bar{R})^{-1} (S^u)^T \bar{Q} S^x x(0) \quad (3.12)$$

The optimal cost (obtained by back-substituting the optimal input U^*) results in a quadratic function of the initial condition $x(0)$ as the matrix is just constant:

$$J^*(x(0)) = x(0)^T \left((S^x)^T \bar{Q} S^x - (S^x)^T \bar{Q} S^u \left((S^u)^T \bar{Q} S^u + \bar{R} \right)^{-1} (S^u)^T \bar{Q} S^x \right) x(0) \quad (3.13)$$

If there are state or input constraints, solving this problem by matrix inversion might not yield a feasible input sequence.

Recursive Approach to the Optimal Control Problem

The recursive approach to solve this problem is based on dynamic programming and splitting up the problem into smaller ones. In particular, the problem is solved backwards by minimizing the cost-to-go at some stage j (with $x(j)$ as initial condition):

$$J_j^*(x(j)) := \min_{U_{j \rightarrow N}} x_N^T P x_N + \sum_{i=j}^{N-1} (x_i^T Q x_i + u_i^T R u_i) \quad (3.14)$$

$$s.t. \ x_{i+1} = A x_i + B u_i, \ i = j, \dots, N-1 \quad (3.15)$$

$$x_j = x(j) \quad (3.16)$$

Considering the one step cost-to-go from time $N-1$ and replacing the state x_N by the expression obtained from the dynamics equation $x_N = A x_{N-1} + B u_{N-1}$, results in (using the notation $x_j^T P_j x_j$ to denote the optimal cost-to-go from state j on; in this case only the terminal cost):

$$J_{N-1}^*(x_{N-1}) = \min_{U_{N-1}} \{ x_{N-1}^T (A^T P_N A + Q) x_{N-1} + u_{N-1}^T (B^T P_N B + R) u_{N-1} + 2 x_{N-1}^T A^T P_N B u_{N-1} \} \quad (3.17)$$

Setting the gradient with respect to the input to zero and solving the equation, results in the following optimal input and the cost-to-go (F_i describes the optimal control gain at time i):

$$u_{N-1}^* = -(B^T P_N B + R)^{-1} B^T P_N A x_{N-1} =: F_{N-1} x_{N-1} \quad (3.18)$$

$$J_{N-1}^*(x_{N-1}) = x_{N-1}^T P_{N-1} x_{N-1} \quad (3.19)$$

The cost-to-go from time $N-1$ is based on the matrix P_{N-1} with definition:

$$P_{N-1} = A^T P_N A + Q - A^T P_N B (B^T P_N B + R)^{-1} B^T P_N A \quad (3.20)$$

Going back in time, one would theoretically have to solve a two step problem in the next step, optimizing over u_{N-1} and u_{N-2} . However, as the principle of optimality states that an optimal path consists of optimal subpaths, it can be assumed that the input from $N-1$ to N is already optimal and the cost-to-go and control from $N-1$ can be used as is. Using a similar procedure as before results in the optimal input u_{N-2}^* :

$$u_{N-2}^* = -(B^T P_{N-1} B + R)^{-1} B^T P_{N-1} A x_{N-2} =: F_{N-2} x_{N-2} \quad (3.21)$$

The corresponding optimal cost-to-go from $N-2$ is:

$$J_{N-2}^*(x_{N-2}) = x_{N-2}^T P_{N-2} x_{N-2} \quad (3.22)$$

Where the cost is determined by P_{N-2} :

$$P_{N-2} = A^T P_{N-1} A + Q - A^T P_{N-1} B (B^T P_{N-1} B + R)^{-1} B^T P_{N-1} A \quad (3.23)$$

This recursion can be continued, resulting in the following optimal inputs and corresponding expressions for the optimal cost-to-go from state i :

$$u_i^* = -(B^T P_{i+1} B + R)^{-1} B^T P_{i+1} A x_i =: F_i x_i, \quad i = 0, \dots, N-1 \quad (3.24)$$

$$J_i^*(x_i) = x_i^T P_i x_i \quad (3.25)$$

The matrix P_i is given by the so-called *discrete-time Riccati equation* or the *Riccati difference equation* (RDE) with initialization using the terminal cost $P_N = P$:

$$P_i = A^T P_{i+1} A + Q - A^T P_{i+1} B (B^T P_{i+1} B + R)^{-1} B^T P_{i+1} A \quad (3.26)$$

The N -step cost-to-go from the initial state $x(0)$ results with P_0 :

$$J^*(x(0)) = J_0^*(x(0)) = x(0)^T P_0 x(0) \quad (3.27)$$

Comparison of Batch and Recursive Approaches

There is a fundamental difference between the batch approach to the optimal control problem, where a sequence of inputs $U^*(x(0))$ of *numerical values* only depending on the initial state is returned, and the recursive approach that computes *feedback policies* $u_i^* = F_i x_i$, depending on each x_i . If the model is perfect and the system evolution exactly follows the model, the results are identical for both methods. However, the recursive solution is generally more robust against disturbances and model imperfections as it can use state feedback to still compute an adapted input, even if the prediction deviates from the true state trajectory. It is also computationally more efficient as it only requires to solve small one-step problems at the time, compared to the Batch approach that includes the inversion of some Hessian (that continues to grow for longer time horizons N).

Adding constraints on x_i and/or u_i , as they are present in most system in practice, both models would break down in the form they were presented, but an adaption of the Batch approach can be implemented in an easy way by performing a constrained optimization for the current state. Due to its capability to deal with constraints, the batch approach will be used in the following sections. The problem of disturbance rejection and model errors will be addressed by the receding horizon and the repeated optimization after every step (see next section).

3.2 Receding Horizon

As already illustrated in figure 1.1, the receding horizon strategy allows to introduce some kind of feedback as the previous states measurements are available before each new iteration and influence the optimization problem. This can also be illustrated with the block diagram in figure 3.1 where the optimal input sequence is determined through the feedback gains K (computed offline) multiplied by the initial state of the current iteration $x(k)$ ($U^* = Kx(k)$).

Additionally, it can be shown that the accuracy of the predictions is highly dependent on the horizon length N . Looking further ahead generally results in more accurate predictions for the state trajectory early on. In case of a shorter finite time horizon, the adaption of the predicted trajectory only adjusts to significant changes shortly before they occur. Figure 3.2 illustrates the state trajectory and (past) predictions for $N = 5$ and figure 3.3 for $N = 20$ (current prediction is red, previous predictions are gray, the real trajectory up to this point is black).

Assuming the finite horizon LQR controller with weight R results as $K_{R,N}$, the closed-loop dynamics can be written as $x(k+1) = (A + BK_{R,N})x(k)$. However, even though the controller might lead to a stable closed-loop system for some $N > \bar{N}$, it might happen that there is a range of N beyond \bar{N} where the system becomes unstable again (depending on the weight R).

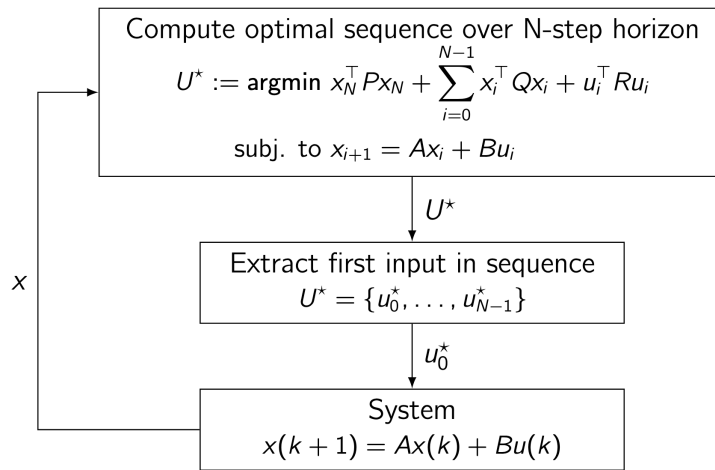


Figure 3.1: Block Diagram illustrating Receding Horizon Control (lecture slides)

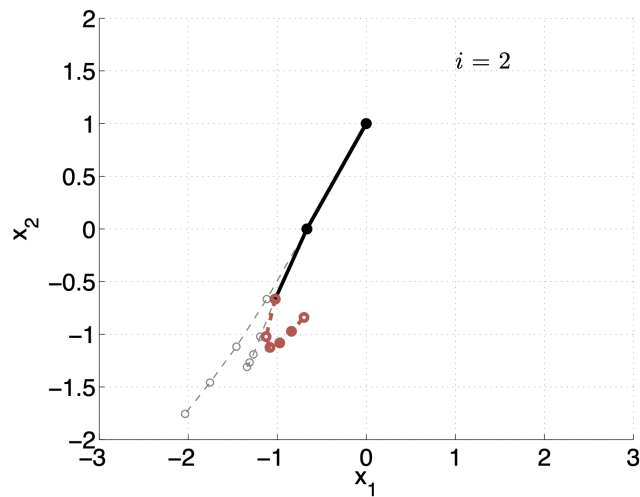


Figure 3.2: Real State Trajectory (black) as well as (past) Predictions (gray and red) for $N = 5$ (lecture slides)

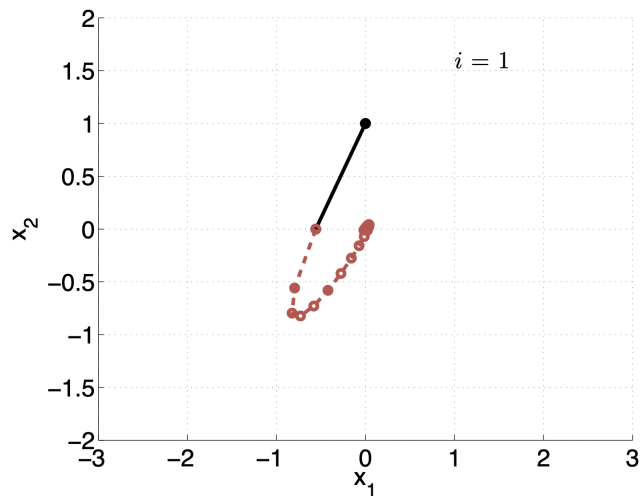


Figure 3.3: Real State Trajectory (black) as well as (past) Predictions (gray and red) for $N = 20$ (lecture slides)

3.3 Infinite Horizon LQR

For an infinite time horizon, the optimal control problem can again be written in form of an optimization problem (note the ∞ subscripts denoting the infinite horizon):

$$J_\infty(x(0)) = \min_{u(\cdot)} \sum_{i=0}^{\infty} (x_i^T Q x_i + u_i^T R u_i) \quad (3.28)$$

$$\text{s.t. } x_{i+1} = A x_i + B u_i, \quad i = 0, 1, 2, \dots, \infty \quad (3.29)$$

$$x_0 = x(0) \quad (3.30)$$

Similar to the dynamic programming approach, this results in the following optimal input:

$$u^*(k) = -(B^T P_\infty B + R)^{-1} B^T P_\infty A x(k) := F_\infty x(k) \quad (3.31)$$

The infinite horizon cost-to-go starting from state $x(k)$ is given by:

$$J_\infty(x(k)) = x(k)^T P_\infty x(k) \quad (3.32)$$

The matrix P_∞ that is required for both the computation of the input signals $u(\cdot)$ and the infinite horizon cost-to-go theoretically can be determined through an infinite recursion of the Riccati difference equation (RDE). While this is not very practical, it turns out that if the solution converges to some P_∞ , this matrix must satisfy the *Algebraic Riccati equation (ARE)*:

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B (B^T P_\infty B + R)^{-1} B^T P_\infty A \quad (3.33)$$

The constant feedback matrix F_∞ is also described as the asymptotic form of the linear quadratic regulator (LQR). The existence of a positive definite solution P_∞ of the ARE can be assumed, if the system (A, B) is stabilizable and $(Q^{1/2}, A)$ is detectable (i.e. if the effect of the state can be seen in the cost).

Stability of Infinite Horizon LQR

Another very significant advantage of infinite horizon LQR is that the close-loop system $x(k+1) = (A + B F_\infty) x(k)$ with optimal input $u(k) = F_\infty x(k)$ is guaranteed to be asymptotically stable under the assumptions that (A, B) is stabilizable and $(Q^{1/2}, A)$ is detectable (this was not given in the finite horizon case). Convergence can be proven using the Lyapunov function given by $J^*(x(k)) = x(k)^T P_\infty x(k)$ (P_∞ solves the algebraic Riccati equation) directly leading to the conclusion for global asymptotic stability:

$$\lim_{k \rightarrow \infty} x(k) = 0 \quad (3.34)$$

The third requirement of Lyapunov functions (steady decrease over time) $J^*(x(k+1)) < J^*(x(k))$ can be shown easily as the the difference is exactly one stage cost. However, infinite horizon solutions are not very practical, wherefore tricks are used to simulate quasi-infinite horizons in model predictive control.

Choice of Terminal Weight P in Finite Horizon Control

The terminal weight P can be chosen arbitrarily in finite horizon control and therefore for example also to match the infinite horizon solution i.e. make it equal to the optimal cost from N to ∞ (fulfilling the algebraic Riccati equation). Another possible choice is to assume no control action at the end of the horizon resulting in $x(k+1) = A x(k)$ for $k = N, \dots$ (only makes sense if the system is asymptotically stable or no p.d. solution exists for P). P can then be determined as the solution to the Lyapunov equation $A^T P A + Q = P$. The last possible choice is to require state and input to be zero after time N ($x_{N+i} = 0, \forall i$) and therefore choose an arbitrary terminal weight P .

4 Introduction to Convex Optimization

4.1 Main Concepts

Many problems in model predictive control include optimization problems in the following form:

$$\min_{x \in \text{dom}(f)} f(x) \quad (4.1)$$

$$\text{s.t. } g_i(x) \leq 0, \quad i = 1, \dots, m \quad (4.2)$$

$$h_i(x) = 0, \quad i = 1, \dots, p \quad (4.3)$$

- The optimization variable might be vector-valued $x = (x_1, \dots, X_n)^T$
- The objective function is defined as $f : \text{dom}(f) \rightarrow \mathbb{R}$
- The domain of the objective function is denoted by $\text{dom}(f) \subseteq \mathbb{R}^n$
- $g_i(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ are optional inequality constraint functions
- $h_i(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ are optional equality constraint functions
- The set of feasible solutions or just the *feasible set* is defined as follows:

$$\mathcal{X} = \{x \in \text{dom}(f) \mid g_i(x) \leq 0, h_k(x) = 0, i = 1, \dots, m, k = 1, \dots, p\} \quad (4.4)$$

The point x (possibly vector-valued in \mathbb{R}^n) can be categorized as follows for the optimization problem in standard form above:

- A point $x \in \text{dom}(f)$ fulfilling all inequality and equality constraints is called *feasible point*
- A point $x \in \text{dom}(f)$ fulfilling all equality constraints and all inequality constraints strictly ($g_i(x) < 0, i = 1, \dots, m$) is called *strictly feasible point*
- The *optimal value* is the lowest possible cost value that can be achieved $p^* = f(x^*) = \min_{x \in \mathcal{X}} f(x)$ (sometimes also denoted by f^* or J^*)
- Any feasible x^* that achieves the optimal value p^* is described as *optimizer* (not necessarily unique). Formally, an optimizer $x^* \in \mathcal{X}$ fulfills $f(x^*) \leq f(x)$ for all feasible $x \in \mathcal{X}$, i.e. is part of the set (ideally unique for control problems):

$$\arg \min_{x \in \mathcal{X}} f(x) = \{x \in \mathcal{X} \mid f(x) = p^*\} \quad (4.5)$$

Active, Inactive and Redundant Constraints

Considering the same problem formulation as before, an inequality constraint $g_i(x) \leq 0$ is said to be *active* if $g_i(\bar{x}) = 0$ holds at \bar{x} and inactive otherwise. Equality constraints $h_i(x) = 0$ are always active for all x in the feasible set. Constraints (equality or inequality) that are *redundant* do not reduce the feasible set and can therefore be removed without changing the solution of the optimization problem. They should in fact always be removed before solving the problem for efficiency reasons.

Optimality

A point $x \in \mathcal{X}$ is *locally optimal* for some $R > 0$ (local minimum) if it fulfills the following inequality:

$$y \in \mathcal{X}, \|y - x\| \leq R \Rightarrow f(y) \geq f(x) \quad (4.6)$$

Similarly, $x \in \mathcal{X}$ is *globally optimal* (global minimum) if it satisfies the following inequality:

$$y \in \mathcal{X} \Rightarrow f(y) \geq f(x) \quad (4.7)$$

The problem is said to be *unbounded below* if $p^* = -\infty$. Also depending on the feasible set \mathcal{X} , the problem is said to be *infeasible* if \mathcal{X} is empty, while it is *unconstrained* if $\mathcal{X} = \mathbb{R}^n$.

Linear Programs and MILPs

Linear programs (linear cost and constraint functions) of the following form are generally easy to solve as the linear optimization over a polytope can be implemented efficiently:

$$\min_x c^T x \quad (4.8)$$

$$\text{s.t. } Gx \leq h \quad (4.9)$$

$$Ax = b \quad (4.10)$$

Mixed integer linear programs are linear programs with additional binary or integer constraints that are part of the problem and can be written in the following form. They are usually hard to solve:

$$\min_x c^T x \quad (4.11)$$

$$\text{s.t. } Gx \leq h \quad (4.12)$$

$$Ax = b \quad (4.13)$$

$$x \in \{0, 1\}^n \text{ or } x \in \mathbb{Z}^n \quad (4.14)$$

4.2 Convex Sets

Convexity helps significantly to solve optimization problems efficiently and reliably as iterative algorithms can be used, resulting in admissible solutions during every iteration. Formally, a set \mathcal{X} is convex, if and only if any pair of points fulfills:

$$\lambda x + (1 - \lambda)y \in \mathcal{X}, \quad \forall \lambda \in [0, 1], \forall x, y \in \mathcal{X} \quad (4.15)$$

Intuitively, this is equivalent to the fact that all lines starting and ending in \mathcal{X} stay within \mathcal{X} (convex sets do not have holes, the boundary does not have any breaks, etc.). Similarly, a convex combination of x_1, \dots, x_k is defined as follows:

$$x = \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_k x_k, \quad \theta_1 + \dots + \theta_k = 1, \quad \theta \geq 0 \quad (4.16)$$

Equality constraints as seen in mathematical optimization problems lead to *hyperplanes* defined by $\{x \in \mathbb{R}^n \mid a^T x = b\}$ for $a \neq 0$, where $a \in \mathbb{R}^n$ is the normal vector. Hyperplanes are both affine and convex as illustrated in figure 4.1 for the case $n = 2$ where a hyperplane corresponds to a simple line.

Halfspaces are all points that lie on one side of a hyperplane (geometric interpretation of inequality constraints) and can therefore be described by $\{x \in \mathbb{R}^n \mid a^T x \leq b\}$. They can either be open (strict inequality) or closed and are always convex (illustrated in figure 4.1 as well).

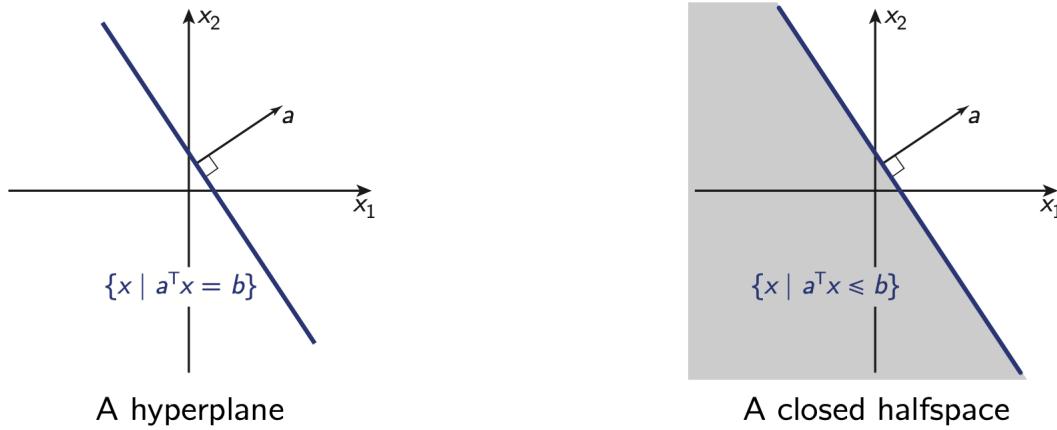


Figure 4.1: Convex Hyperplanes and Halfspaces (lecture slides)

A *polyhedron* is the intersection of a finite number of closed halfspaces, what again results in a convex set as *intersection preserves convexity* (union does not). Such polyhedra might be unbounded in certain directions and can be formally described by (using matrix $A = [a_1, \dots, a_m]^T$ and $b = [b_1, \dots, b_m]^T$):

$$P := \{x \mid a_i^T x \leq b_i, i = 1, \dots, m\} = \{x \mid Ax \leq b\} \quad (4.17)$$

Polytopes are special cases of polyhedra, which are bounded in all directions and commonly occur in MPC problems. They are also always convex.

Another example of convex sets are ellipsoids, which can formally be defined as:

$$\{x \mid (x - x_c)^T A^{-1} (x - x_c) \leq 1\} \quad (4.18)$$

Geometrically x_c then is the center of the ellipsoid and $A > 0$ a positive definite matrix, whose eigenvalues describe the squared length of the semi-axes. The *Euclidean ball* is a special case where both eigenvalues are equal ($A = rI$) and the notation simplifies to $B(x_c, r) = \{x \mid \|x - x_c\|_2 \leq r\}$. Similarly, also other convex norm balls can be defined as $\{x \mid \|x - x_c\|_p \leq r\}$ with a certain ℓ_p norm.

4.3 Convex Functions

The notation of convexity can also be applied to functions where some function is called convex, if the line connecting two points on the graph is always above or on the graph. Formally, a function $f : \text{dom}(f) \rightarrow \mathbb{R}$ is *convex* if and only if its domain is convex and the following inequality holds:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y), \quad \forall \lambda \in [0, 1], \forall x, y \in \text{dom}(f) \quad (4.19)$$

A function is called *strictly convex*, if the inequality for convexity holds strictly. The other way around, a function f is called *concave*, if its domain is convex and the function $-f$ is convex. Convexity can alternatively also be checked through the first- or second-order conditions:

A differentiable function $f : \text{dom}(f) \rightarrow \mathbb{R}$ with convex domain is convex if and only if its first order approximation around any point x is a global underestimator of the function values:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \quad \forall x, y \in \text{dom}(f) \quad (4.20)$$

The gradient $\nabla f(x)$ is generally vector-valued and contains all partial derivatives:

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right]^T \quad (4.21)$$

The second order condition states that a twice differentiable function $f : \text{dom}(f) \rightarrow \mathbb{R}$ with convex domain is convex if and only if its Hessian is positive semi-definite (or strictly convex for a positive definite Hessian):

$$\nabla^2 f(x) \geq 0, \quad \forall x \in \text{dom}(f) \quad (4.22)$$

The Hessian $\nabla^2 f(x)$ is the matrix with the second partial derivatives as entries:

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \quad (4.23)$$

4.4 Level and Sublevel Sets

A *level set* L_α for some function f and value α is the set of all points $x \in \text{dom}(f)$, which have the same value:

$$L_\alpha := \{x \in \text{dom}(f) \mid f(x) = \alpha\} \quad (4.24)$$

In two dimensions, the level sets for a scalar-valued function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ can be drawn as contour lines. Similar to level sets, a *sublevel set* C_α for some function f and value α is defined as all points $x \in \text{dom}(f)$ that have a function value smaller than α :

$$C_\alpha := \{x \in \text{dom}(f) \mid f(x) \leq \alpha\} \quad (4.25)$$

In connection with convex functions it holds that the sublevel sets of a convex function are always convex for all choices of α (the other direction does not hold in general).

4.5 Examples of Convex Functions

The following functions are examples for convex functions (generally on \mathbb{R}):

- Affine functions $ax + b$ for any $a, b \in \mathbb{R}$
- Exponential functions e^{ax} for any $a \in \mathbb{R}$
- Potential functions x^α on the domain $\mathbb{R}_{++} = (0, \infty)$ for $\alpha \geq 1$ and $\alpha \leq 0$
- Vector norms on \mathbb{R}^n : $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ for $p \geq 1$

The following functions are concave:

- Affine functions $ax + b$ for any $a, b \in \mathbb{R}$
- Potential functions x^α on the domain $\mathbb{R}_{++} = (0, \infty)$ for $0 \leq \alpha \leq 1$
- Logarithmic functions $\log(x)$ on $\mathbb{R}_{++} = (0, \infty)$
- Entropy functions $-x \log(x)$ on $\mathbb{R}_{++} = (0, \infty)$

Convexity is preserved under the following operations (among others):

- Non-negative weighted sums of convex functions
- Composition of a convex function with an affine function
- Pointwise maximum and minimum of multiple convex functions
- Partial minimization of a convex function

4.6 Convex Optimization Problems

A convex optimization problem in standard form can be written as below where f, g_1, \dots, g_m are convex functions (element lying in a convex set according to the sublevel set condition), $\text{dom}(f)$ is a convex set and the equality constraint functions $h_i(x) = a_i^T x - b$ are all affine (otherwise convexity cannot be guaranteed):

$$\min_{x \in \text{dom}(f)} f(x) \quad (4.26)$$

$$\text{subj. to } g_i(x) \leq 0, \quad i = 1, \dots, m \quad (4.27)$$

$$a_i^T x = b_i, \quad i = 1, \dots, p \quad (4.28)$$

Usually, the affine constraints are collected in a single matrix, resulting in the form:

$$\min_{x \in \text{dom}(f)} f(x) \quad (4.29)$$

$$\text{subj. to } g_i(x) \leq 0, \quad i = 1, \dots, m \quad (4.30)$$

$$Ax = b, \quad A \in \mathbb{R}^{p \times m} \quad (4.31)$$

A very important property of such convex optimization problems is that the feasible set is always convex as well. Additionally, it can be shown that any locally optimal solution is also globally optimal (otherwise the convexity property would be violated), but the optimizer does not necessarily have to be unique (strong convexity is not guaranteed).

Equivalent Optimization Problems

Two different optimization problems are informally called *equivalent*, if the solution of one problem can be easily converted to the solution of the other one and vice-versa. This is especially important if one of the problems is easier to solve. The first example is the case where equality constraints can be introduced to simplify the following problem:

$$\min_x f(A_0 x + b_0) \quad (4.32)$$

$$\text{subj. to } g_i(A_i x + b_i) \leq 0, \quad i = 1, \dots, m \quad (4.33)$$

This problem can also be formulated as:

$$\min_x f(y_0) \quad (4.34)$$

$$\text{subj. to } g_i(y_i) \leq 0, \quad i = 1, \dots, m \quad (4.35)$$

$$A_i x + b = y_i, \quad i = 0, 1, \dots, m \quad (4.36)$$

Similarly, some problems might be easier to solve using equality constraints instead of inequality constraints, what can be achieved using slack variables s_i , starting with the following system:

$$\min_x f(x) \quad (4.37)$$

$$\text{subj. to } A_i x \leq b_i, \quad i = 1, \dots, m \quad (4.38)$$

With the slack variables s_i this problem can be modified to:

$$\min_{x, s_i} f(x) \quad (4.39)$$

$$\text{subj. to } A_i x + s_i = b_i, \quad i = 1, \dots, m \quad (4.40)$$

$$s_i \geq 0, \quad i = 1, \dots, m \quad (4.41)$$

Linear Programs (LP)

A general linear program can be characterized by the fact that the cost function as well as the constraint functions are affine, resulting in the following form:

$$\min_{x \in \mathbb{R}^n} c^T x \quad (4.42)$$

$$\text{subj. to } Gx \leq h \quad (4.43)$$

$$Ax = b \quad (4.44)$$

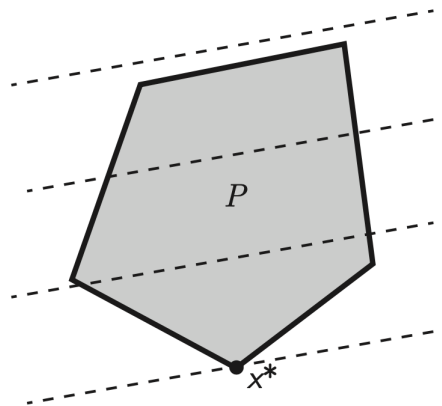


Figure 4.2: Linear optimization on a polytope (lecture slides)

The feasible set P of such a linear program generally is a polyhedron, but often reduces to a polytope (polyhedron that is bounded in all directions), as shown in figure 4.2. The inequalities limit the polyhedron in a certain direction of the n -dimensional space, while equality constraints additionally require all feasible points to lie within a given plane. If the feasible set P is empty, the problem is infeasible.

Given that the feasible set is not empty, the affine cost function defines a hyperplane (perpendicular to the vector c) that is then pushed towards its minimum. The level sets (lines where the cost function assumes a constant value) are marked as dashed lines in the example problem in figure 4.2. If p^* is chosen to denote the optimal value of the cost function and X_{opt} the set of *optimizers* (values for the optimization variable that achieve the optimum cost), the following three cases can be distinguished:

- The solution is bounded ($p^* > -\infty$) and the optimizer is unique (i.e. X_{opt} is a singleton). This is the case depicted in figure 4.2. Such a unique solution of a linear program is always achieved at one of the vertices of the feasible set.
- The solution is bounded, but there are multiple optimizers (might be a finite or infinite set X_{opt}). This happens, if the cost is linearly dependent on some constraint and the cost hyperplane is aligned with one of the polyhedron boundaries (i.e. all of the points on a boundary are optimizers). The problem then also exhibits a very high sensitivity as small changes to the cost function or the constraints will cause the optimizer to jump to one of the vertices (slight variations yield large variations in the solution).
- The solution is unbounded (i.e. $p^* = -\infty$) as the set of feasible solutions (polyhedron) is not bounded in the direction of the minimum of the cost function. No optimizers can be specified in this case.

Quadratic Programs (QP)

Quadratic programs are generally similar to linear programs with the main difference that the cost function is now quadratic:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T H x + q^T x + r \quad (4.45)$$

$$\text{subj. to } Gx \leq h \quad (4.46)$$

$$Ax = b \quad (4.47)$$

The constant r can also be left out to determine the optimizer (not to determine the value of the cost function at the optimizer though) and the cost function is convex, if H is positive definite. In the case of positive semi-definiteness, the cost collapses in a few directions and some state components might not have an effect on the cost, while quadratic programs with concave objectives (negative definite H) are generally hard to solve. If the feasible set P is not empty, a unique solution is guaranteed to exist for positive definite H . The optimizer can then lie strictly inside the feasible polyhedron or on the boundary, depending on the location of the minimum of the cost function (if that lies inside or outside of the polyhedron). In contrast to linear programs, the optimizer may lie anywhere on the boundary and does not have to coincide with a vertex.

4.7 Duality

Lagrange Dual Problem

For the definition of the dual problem, the standard primal optimization problem from the beginning of this chapter is reconsidered with the primal decision variable x , domain $\text{dom}(f)$ and optimal value p^* :

$$\min_{x \in \text{dom}(f)} f(x) \quad (4.48)$$

$$\text{s.t. } g_i(x) \leq 0, \quad i = 1, \dots, m \quad (4.49)$$

$$h_i(x) = 0, \quad i = 1, \dots, p \quad (4.50)$$

The *Lagrangian function* $L : \text{dom}(f) \times \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ of this problem is defined as a weighted sum of the objective and constraint functions:

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{i=1}^p \nu_i h_i(x) \quad (4.51)$$

Here, λ_i are described as the inequality Lagrange multipliers for $g_i(x)$ with $\lambda_i \geq 0$ and ν_i are the equality Lagrange multipliers for $h_i(x) = 0$. Based on this function, the *Lagrange dual function* $d : \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ can be defined (the infimum will usually be simplified to a minimum later on when existence is assumed):

$$d(\lambda, \nu) = \inf_{x \in \text{dom}(f)} L(x, \lambda, \nu) = \inf_{x \in \text{dom}(f)} \left[f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{i=1}^p \nu_i h_i(x) \right] \quad (4.52)$$

Independent from the convexity of the primal problem, the dual function is always concave (can be shown for all pointwise infima of affine functions). Furthermore, it provides a lower bound for the optimal cost p^* :

$$d(\lambda, \nu) \leq f(x^*) + \sum_{i=1}^m \lambda_i g_i(x^*) + \sum_{i=1}^p \nu_i h_i(x^*) \leq f(x^*) = p^* \quad (4.53)$$

The first inequality holds due to $d(\lambda, \nu)$ being the minimum value of the Lagrangian, while the second one can be proven because x^* is feasible and $h_i(x^*) = 0$ as well as $g_i(x^*) \leq 0$ hold. For some problems, this lower bound is very good (as shown in the next sections), for others it might be $-\infty$ and therefore contain no information on the optimal cost. For cases between $d(\lambda, \nu) = p^*$ and $d(\lambda, \nu) = -\infty$, it can be used as a suboptimality bound and is often used as the termination condition for solvers:

$$f(x) - p^* \leq f(x) - d(\lambda, \nu) \leq \epsilon = tol \quad (4.54)$$

This yields the definition of the dual problem (with the domain being defined as all parameter combinations that yield a useful result $\text{dom}(d) = \{\lambda, \nu \mid d(\lambda, \nu) > -\infty\}$):

$$\max_{\lambda, \nu} d(\lambda, \nu) \quad (4.55)$$

$$\text{subj. to } \lambda \geq 0 \quad (4.56)$$

It is convex as $\max_{\lambda, \nu} d(\lambda, \nu) = \min_{\lambda, \nu} -d(\lambda, \nu)$ and the negative of a concave function is convex (even if the primal problem was not convex) and has the optimal value $d^* \leq p^*$. The point (λ, ν) is *dual feasible* if $\lambda \geq 0$ (given as constraint) and $(\lambda, \nu) \in \text{dom}(d) = \{\lambda, \nu \mid d(\lambda, \nu) > -\infty\}$, what can often also be imposed as a constraint. The dual problem of an LP/QP is always an LP/QP.

Weak and Strong Duality

Weak duality requires that $d^* \leq p^*$ holds, what is always fulfilled by the definition of the dual problem. However, in some cases, *strong duality* $d^* = p^*$ holds (usually not for non-convex problems; all LPs fulfill this). It is possible to impose conditions on convex problems to guarantee that strong duality is fulfilled and it might then be much easier to solve the dual problem instead of the primal one to determine the best achievable cost (or vice-versa). For the example of a hard MILP (mixed integer linear program), the dual is an easy standard linear program.

For some standard convex optimization problem of the following form, the *Slater condition* can be used to determine, whether some problem of the following form fulfills strong duality:

$$\min_{x \in \mathbb{R}^n} c^T x \quad (4.57)$$

$$\text{subj. to } g_i(x) \leq 0, \quad i = 1, \dots, m \quad (4.58)$$

$$Ax = b, \quad A \in \mathbb{R}^{p \times n} \quad (4.59)$$

The Slater condition then states that the problem fulfills strong duality ($d^* = p^*$) if there is at least one strictly feasible point:

$$\{x \mid Ax = b, g_i(x) < 0, \forall i \in \{1, \dots, m\}\} \neq \emptyset \quad (4.60)$$

Note that this condition is *sufficient but not necessary* for strong duality. There are also many other constraint qualification conditions that can be used to check for strong duality in convex problems as the one above.

4.8 Optimality / KKT Conditions

Assume that the cost function f and all constraint functions g_i and h_i are differentiable (but not necessarily convex), then the *Karush-Kuhn-Tucker conditions* (KKT conditions) can be formulated:

1. Primal feasibility:

$$g_i(x^*) \leq 0, \quad i = 1, \dots, m \quad (4.61)$$

$$h_i(x^*) = 0, \quad i = 1, \dots, p \quad (4.62)$$

2. Dual feasibility:

$$\lambda^* \geq 0 \quad (4.63)$$

3. Complementary Slackness:

$$\lambda_i^* \cdot g_i(x^*) = 0, \quad i = 1, \dots, m \quad (4.64)$$

4. Stationarity:

$$\nabla_x L(x^*, \lambda^*, \nu^*) = \nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) + \sum_{i=1}^p \nu_i^* \nabla h_i(x^*) = 0 \quad (4.65)$$

As strong duality implies $d^* = d(\lambda^*, \nu^*) = p^* = f(x^*)$, also the following inequality holds, using the fact that d^* is the minimum of the Lagrangian dual function:

$$f(x^*) = d^* = d(\lambda^*, \nu^*) \leq f(x^*) + \sum_{i=1}^m \lambda_i^* g_i(x^*) + \sum_{i=1}^p \nu_i^* h_i(x^*) \leq f(x^*) \quad (4.66)$$

While $h_i(x^*) = 0$ holds by definition also the second expression needs to be zero for the inequalities to hold with equality (what is required as $f(x^*)$ is on both sides of the inequalities). This exactly corresponds to the complementary slackness defined above, meaning that λ_i^* needs to be zero whenever some constraint is satisfied strictly and usually is nonzero if a constraint $g_i(x) = 0$ is active - alternative formulation for the complementary slackness:

$$\begin{cases} \lambda_i^* = 0 & \text{for every } g_i(x^*) < 0 \\ g_i(x^*) = 0 & \text{for every } \lambda_i^* > 0 \end{cases} \quad (4.67)$$

Depending on the type of optimization problem, the KKT conditions pose either necessary and/or sufficient conditions on the optimality of the solutions to primal and dual problems. The duality gap describes the difference between d^* and p^* :

- General optimization problems (*necessary condition*): If x^* and (λ^*, ν^*) are primal and dual optimal solutions with zero duality gap, then x^* and (λ^*, ν^*) satisfy the KKT conditions. This means that if they do not fulfill the KKT conditions that they cannot be primal and dual optimal solutions, but does not state anything on the other direction given that the KKT conditions are fulfilled (not sufficient).
- Convex optimization problems (*sufficient condition*): If x^* and (λ^*, ν^*) satisfy the KKT conditions, then x^* and (λ^*, ν^*) are primal and dual optimal solutions with zero duality gap. Due to sufficiency, the KKT conditions allow to verify that some x^* and (λ^*, ν^*) are optimal solutions, but do not state anything for the other direction if strong duality is not assumed.

- Convex optimization problems with strong duality (Slater condition holds; *necessary and sufficient condition*): If strong duality holds, x^* and (λ^*, ν^*) are primal and dual optimal solutions if and only if they satisfy the KKT conditions. This completes the second case of a convex optimization problem that did not allow to state anything on the optimality of some x^* and (λ^*, ν^*) fulfilling the KKT conditions with arbitrary duality gap.

The fact that the KKT conditions are sufficient for strong duality and optimality comes from the fact that if (x^*, λ^*, ν^*) fulfill the KKT conditions, the following equalities hold (first one due to complementary slackness and the second one uses convexity of the functions and stationarity):

$$p^* = f(x^*) = L(x^*, \lambda^*, \nu^*) \quad (4.68)$$

$$d^* = d(\lambda^*, \nu^*) = L(x^*, \lambda^*, \nu^*) \quad (4.69)$$

4.9 Sensitivity Analysis

Instead of the previously considered general optimization problem, two perturbation parameters u and v are added to arrive at some perturbed optimization problem in order to analyze the sensitivity to small changes:

$$\min_{x \in \text{dom}(f)} f(x) \quad (4.70)$$

$$\text{s.t. } g_i(x) \leq u_i, \quad i = 1, \dots, m \quad (4.71)$$

$$h_i(x) = v_i, \quad i = 1, \dots, p \quad (4.72)$$

The dual problem can be formulated as:

$$\max_{\lambda, \nu} d(\lambda, \nu) - u^T \lambda - v^T \nu \quad (4.73)$$

$$\text{subj. to } \lambda \geq 0 \quad (4.74)$$

As before, x is the primal decision variable, while (λ, ν) are the dual decision variables. The optimal cost value will result as a function of the perturbations $p^*(u, v)$ in this case. Assuming strong duality for the unperturbed problem with the dual optimizer (λ^*, ν^*) , weak duality for the perturbed system can be assumed and results in $(d^*(\lambda^*, \nu^*) = p^*(0, 0))$ holds due to strong duality):

$$p^*(u, v) \geq d^*(\lambda^*, \nu^*) - u^T \lambda^* - v^T \nu^* = p^*(0, 0) - u^T \lambda^* - v^T \nu^* \quad (4.75)$$

Global sensitivity analysis results in the following conclusions:

- For large λ_i^* and $u_i < 0$, $p^*(u, v)$ increases greatly
- For small λ_i^* and $u_i > 0$, $p^*(u, v)$ does not decrease much
- For large ν^* and positive or negative v_i , $p^*(u, v)$ increases greatly
- For small ν^* and positive or negative v_i , $p^*(u, v)$ does not decrease much

This shows that large Lagrange multipliers imply that the corresponding constraint is important and slightly loosening it might yield a significant optimality gain and reduced cost. This is a very helpful information during controller design so that one can take a closer look at the constraint and decide whether or not some looser constraint might also be sufficient.

Locally, assuming that $p^*(u, v)$ is also differentiable at $(0, 0)$, the Lagrange multipliers are exactly equal to the partial derivatives of the cost function at this point and therefore represent the sensitivity of p^* with respect to the corresponding inequality or equality constraint:

$$\lambda_i^* = -\frac{\partial p^*(0, 0)}{\partial u_i}, \quad \nu^* = -\frac{\partial p^*(0, 0)}{\partial v_i} \quad (4.76)$$

5 Constrained Finite Time Optimal Control

The problems that have to be solved in practice are usually constrained infinite time optimal control problems with the following formulation (with the stage cost $I(x, u)$):

$$J_{\infty}^*(x(0)) = \min_{u^{(\cdot)}} \sum_{i=0}^{\infty} I(x_i, u) \quad (5.1)$$

$$\text{subj. to } x_{i+1} = Ax_i + Bu_i, \quad i = 0, \dots, \infty \quad (5.2)$$

$$x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, \infty \quad (5.3)$$

$$x_0 = x(0) \quad (5.4)$$

Optimizing over a infinite-time trajectory leads to a tradeoff between short- and long-term benefits of the actions, while the long-term component is partially lost in the finite time case. However, it is not very practical to solve the infinite-time case. Instead, the problem is modified to a constrained finite time optimal control problem, by adding a terminal cost $I_f(x, u)$ and a terminal constraint $x_f \in \mathcal{X}_f$ to approximate the ‘tail’ of the cost and constraints. These problems can sometimes be solved and are usually written in the following form for a finite time horizon of length N starting at k (receding horizon notation):

$$J_{k \rightarrow k+N|k}^*(x(k)) = \min_{U_{k \rightarrow k+N|k}} I_f(x_{k+N|k}) + \sum_{i=0}^{N-1} I(x_{k+i|k}, u_{k+i|k}) \quad (5.5)$$

$$\text{subj. to } x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \quad i = 0, \dots, N-1 \quad (5.6)$$

$$x_{k+i|k} \in \mathcal{X}, u_{k+i|k} \in \mathcal{U}, \quad i = 0, \dots, N-1 \quad (5.7)$$

$$x_{k+N|k} \in \mathcal{X}_f \quad (5.8)$$

$$x_{k|k} = x(k) \quad (5.9)$$

The optimization is then carried out over the input trajectory $U_{k \rightarrow k+N|k} = \{u_{k|k}, \dots, u_{k+N-1|k}\}$. The main objective is to understand which constrained finite time optimal control problems are feasible and how some of them can be rewritten as linear or quadratic problems.

5.1 Constrained Linear Optimal Control

To simplify notation, the following formulation (that is slightly less general than the previous one) will be used for the constrained finite-time optimal control problem (CFTOC):

$$J^*(x(k)) = \min_U J(x_0, U) \quad (5.10)$$

$$\text{subj. to } x_{i+1} = Ax_i + Bu_i, \quad i = 0, \dots, N-1 \quad (5.11)$$

$$x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, N-1 \quad (5.12)$$

$$x_N \in \mathcal{X}_f \quad (5.13)$$

$$x_0 = x(k) \quad (5.14)$$

If N is the finite time horizon and \mathcal{X} , \mathcal{U} and \mathcal{X}_f are all polyhedral regions, the cost function (decomposed into terminal cost and stage cost) with the definition $J(x_0, U) = I_f(x_N) + \sum_{i=0}^{N-1} I(x_i, u_i)$ can be written as a quadratic cost using the squared Euclidean norm (or using the 1- or ∞ -norms). In the case of the squared Euclidean norm, the cost function is usually formulated using the matrices $P \geq 0, Q \geq 0$ and $R > 0$:

$$I_f(x_N) = x_N^T P x_N, \quad I(x_i, u_i) = x_i^T Q x_i + u_i^T R u_i \quad (5.15)$$

Using a p -norm (where p is just 1 or ∞ for the corresponding norms), the cost function can be written with three full column rank matrices P, Q and R :

$$I_f(x_N) = \|P x_N\|_p, \quad I(x_i, u_i) = \|Q x_i\|_p + \|R u_i\|_p \quad (5.16)$$

The *feasible set* for such finite time optimal control problems is defined as the set of all initial states for which the problem is feasible. Note that this definition deviates from the previous definition of feasible sets (that were all x that fulfill the constraints) for $i = 0, \dots, N - 1$:

$$\mathcal{X}_N = \{x_0 \in \mathbb{R}^n \mid \exists (u_0, \dots, u_{N-1}), x_i \in \mathcal{X}, u_i \in \mathcal{U}, x_N \in \mathcal{X}_f, x_{i+1} = A x_i + B u_i\} \quad (5.17)$$

As the systems considered in this class will usually be time-invariant, the definition of \mathcal{X}_N for times $i = 0, \dots, N - 1$ is sufficient and the time-horizon can be shifted arbitrarily for time-invariant systems.

5.2 Constrained Optimal Control: Quadratic Cost

While time-varying linear control laws $u_i^* = F_i x_i$ were found for the finite time unconstrained control problem and quadratic cost, a time-invariance linear control law (LQR) $u_i^* = F_\infty x_i$ was obtained in the infinite horizon case ($N \rightarrow \infty$). This section and the next one will consider the case of a constrained optimal control problems and their solution. To arrive at the corresponding solution, the constrained linear optimal control problem with finite time horizon and quadratic cost given by formula 5.15 will be reformulated into a quadratic problem:

$$\min_{z \in \mathbb{R}^n} \frac{1}{2} z^T H z + q^T z + r \quad (5.18)$$

$$\text{subj. to } G z \leq h \quad (5.19)$$

$$A z = b \quad (5.20)$$

The construction of such a quadratic program from the constrained finite time optimal control problem with quadratic cost can be done in two ways with or without substitution.

Construction of a QP with substitution

One option to convert the problem into a quadratic program is to use the constraints $x_{i+1} = A x_i + B u_i$ and $x_0 = x(k)$ and rewrite the costs accordingly:

1. Rewriting the cost using the matrices H, F and Y results in the following quadratic form:

$$\begin{aligned} J(x(k), U) &= U^T H U + 2x(k)^T F U + x(k)^T Y x(k) \\ &= [U^T x(k)^T] \begin{bmatrix} H & F^T \\ F & Y \end{bmatrix} [U^T x(k)^T]^T \end{aligned} \quad (5.21)$$

As the cost is always positive, the corresponding matrix $\begin{bmatrix} H & F^T \\ F & Y \end{bmatrix}$ is positive definite.

2. Rewrite the constraints on state and input $\mathcal{X} = \{x \mid A_x x \leq b_x\}$, $\mathcal{U} = \{u \mid A_u x \leq b_u\}$ and $\mathcal{X}_f = \{x \mid A_f x \leq b_f\}$ in a more compact form:

$$GU \leq w + Ex(k) \quad (5.22)$$

$$G = \begin{bmatrix} A_u & 0 & \cdots & 0 \\ 0 & A_u & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_u \\ 0 & 0 & \cdots & 0 \\ A_x B & 0 & \cdots & 0 \\ A_x A B & A_x B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_f A^{N-1} B & A_f A^{N-2} B & \cdots & A_f B \end{bmatrix}, \quad E = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -A_x \\ -A_x A \\ -A_x A^2 \\ \vdots \\ -A_f A^N \end{bmatrix}, \quad b = \begin{bmatrix} b_u \\ b_u \\ \vdots \\ b_u \\ b_x \\ b_x \\ b_x \\ \vdots \\ b_f \end{bmatrix} \quad (5.23)$$

3. Rewrite the constrained optimal control problem as quadratic program:

$$J^*(x(k)) = \min_U [U^T x(k)^T] \begin{bmatrix} H & F^T \\ F & Y \end{bmatrix} [U^T x(k)^T]^T \quad (5.24)$$

subj. to $GU \leq w + Ex(k)$

For any given initial condition, the optimal finite horizon input sequence U^* can then be found by solving the corresponding quadratic problem with one of many available solvers.

Construction of a QP without substitution

While the substitution results in a simple form, it is sometimes less efficient than solving the optimal control problem through some auxiliary quadratic problem that still contains inequality and equality constraints that are listed separately (the state is kept as optimization variable). To state this, the variable $z = [x_1^T \ \dots \ x_N^T \ u_0^T \ \dots \ u_{N-1}^T]^T$ is used to summarize constraints on state and input. The resulting quadratic problem will be of the following form:

$$J^*(x(k)) = \min_z [z^T x(k)^T] \begin{bmatrix} \bar{H} & 0 \\ 0 & Q \end{bmatrix} [z^T x(k)^T]^T \quad (5.25)$$

$$\text{subj. to } G_{in} z \leq W_{in} + E_{in} x(k) \quad (5.26)$$

$$G_{eq} z = E_{eq} x(k) \quad (5.27)$$

The matrices G_{in} , G_{eq} , w_{in} , E_{in} , E_{eq} can be computed from the constraints (using the same notation as before):

$$G_{eq} = \left[\begin{array}{cccc|cccc} I & & & & -B & & & \\ -A & I & & & & -B & & \\ & -A & I & & & & -B & \\ & & & \ddots & & & & \ddots \\ & & & & -A & I & & \\ & & & & & & & -B \end{array} \right], \quad E_{eq} = \begin{bmatrix} A \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (5.28)$$

$$\begin{aligned}
G_{in} &= \left[\begin{array}{cccccc|cccc}
0 & & & & & & 0 & & & & & & \\
A_x & & & & & & 0 & & & & & & \\
& A_x & & & & & & 0 & & & & & \\
& & \ddots & & & & & & \ddots & & & & \\
& & & A_x & & & & & & 0 & & & \\
& & & & A_f & & & & & & 0 & & \\
0 & & & & & A_u & & & & & & & \\
& 0 & & & & & A_u & & & & & & \\
& & & \ddots & & & & \ddots & & & & & \\
& & & & 0 & & & & A_u & & & & \\
& & & & & 0 & & & & A_u & & & \\
& & & & & & & & & & A_u & & \\
& & & & & & & & & & & A_u & \\
& & & & & & & & & & & & A_u
\end{array} \right], \quad w_{in} = \begin{bmatrix} b_x \\ b_x \\ \vdots \\ b_x \\ b_f \\ b_u \\ b_u \\ \vdots \\ b_u \\ b_u \end{bmatrix} \quad (5.29) \\
E_{in} &= [-A_x^T \ 0 \ \cdots \ 0]^T, \quad \bar{H} = \text{diag}(Q, \dots, Q, P, R, \dots, R) \quad (5.30)
\end{aligned}$$

As mentioned, the method without substitution is often preferred in practical applications as it even contains more decision variables ($N(m + n)$ instead of $N \cdot m$ with substitution), but the sparse matrix structure speeds up the computation significantly.

Quadratic Cost State Feedback Solution

The constrained finite time optimal control problem in the form of equation 5.24 can be described as a *multiparametric quadratic program (mp-QP)*, which is a function of $x(k)$ and a common QP for some fixed $x(k)$. The first component of the optimal solution, that will be implemented on the system for receding horizon control, can be written as:

$$u_0^* = \kappa(x(k)), \quad x(k) \in \mathcal{X}_0 \quad (5.31)$$

Where $\kappa : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a continuous and piecewise affine function on polyhedral set of the form $CR^j = \{x \in \mathbb{R}^n | H^j x \leq K^j\} \subset \mathcal{X}_0$:

$$\kappa(x) = F^j x + g^j, \quad x = CR^j, \quad j = 1, \dots, N^r \quad (5.32)$$

The value function $J^*(x(k))$ is then convex and piecewise quadratic on the polyhedra. Plotting this result of piecewise affine control laws $u^*(x)$, it can be observed that larger time horizons N lead to larger feasible sets but higher complexities of the control law itself.

5.3 Constrained Optimal Control: 1- and ∞ -Norm Cost

Considering the constrained finite time optimal control problem with the norm cost function given by formula 5.16, it is possible to convert it to a linear program of the following form:

$$\min_{z \in \mathbb{R}^n} c^T z \quad (5.33)$$

$$\text{subj. to } Gz \leq h \quad (5.34)$$

$$Az = b \quad (5.35)$$

One option to arrive at the desired form with cost functions like $J(x) = \|x\|_\infty$ or $J(x) = \|x\|_1$ is to introduce some auxiliary variable and write the absolute value of the components x_i to be the $\max\{x_i, -x_i\}$ (in order to convert the norm cost notation into one that is compatible with

linear programs). In case of the ℓ_∞ -norm this is possible starting from the following problem statement:

$$\min_{x \in \mathbb{R}^n} \|x\|_\infty \quad (5.36)$$

$$\text{subj. to } Fx \leq g \quad (5.37)$$

Defining $t = \|x\|_\infty = \max\{x_1, \dots, x_n, -x_1, \dots, -x_n\}$ leads to a more extensive notation:

$$\min_{x,t} t \quad (5.38)$$

$$\text{subj. to } x_i \leq t, \quad i = 1, \dots, n \quad (5.39)$$

$$-x_i \leq t, \quad i = 1, \dots, n \quad (5.40)$$

$$Fx \leq g \quad (5.41)$$

Summarizing the n constraint equations in matrix form results in the *epigraph form*:

$$\min_{x,t} t \quad (5.42)$$

$$\text{subj. to } -\mathbf{1}t \leq x \leq \mathbf{1}t \quad (5.43)$$

$$Fx \leq g \quad (5.44)$$

Similarly, the additional variable $t = |x_i| = \max\{x_i, -x_i\}$ can be introduced to simplify the ℓ_1 -norm problem to the following one:

$$\min_{x \in \mathbb{R}^m, t \in \mathbb{R}^m} \mathbf{1}^T t \quad (5.45)$$

$$\text{subj. to } -t \leq x \leq t \quad (5.46)$$

$$Fx \leq g \quad (5.47)$$

While the described procedure is usually more efficient to convert a constrained finite time optimal control problem with 1- or ∞ -norm cost function to an LP, it is in principle also possible using substitution (as shown for the quadratic case before). However, as this approach is much more complicated than the one presented for these norms, it is omitted here and can be found in the lecture slides.

1- or ∞ -Norm State Feedback Solution

Considering the following formulation, which is consistent with the linear program from before, the problem can be described as a *multiparametric linear program (mp-LP)* (similar to mp-QPs before) and as a LP for each fixed $x(k)$:

$$\min_z c^T z \quad (5.48)$$

$$\text{subj. to } \bar{G}z \leq \bar{w} + \bar{S}x(k) \quad (5.49)$$

$$(5.50)$$

As before, the first component of the multiparametric solution trajectory, can be written as a feedback law on the initial state:

$$u_0^* = \kappa(x(0)), \quad \forall x(0) \in \mathcal{X}_0 \quad (5.51)$$

Again, $\kappa : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a continuous and piecewise affine function on some polyhedral sets that are partitions of the feasible polyhedron \mathcal{X}_0 : $CR^j = \{x \in \mathbb{R}^n | H^j x \leq K^j\}$

$$\kappa(x) = F^j x + g^j, \quad x \in CR^j, \quad j = 1, \dots, N^r \quad (5.52)$$

In contrast to quadratic problems, there is a chance that multiple optimizers exist, which implies that there is a piecewise affine control law (only piecewise because some solutions might not fulfill this requirement). At the same time the control law around the origin in the state space diagram (no constraints active) is equal to the LQR solution. The corresponding value function $J^*(x(0))$ for optimal control, turns out to be convex and piecewise linear (due to the linearity of the cost function) on polyhedra.

5.4 Comparison Quadratic and Linear Cost Functions

Assuming the case of n optimization variables, the quadratic cost function (with positive definite coefficients) yields solutions that fulfill one of the following statements:

- unique and in the interior of the feasible set (no active constraints)
- unique and on the boundary of the feasible (at least one active constraints)

For linear cost functions, the solutions are either:

- unbounded
- unique at a vertex of the feasible set (at least n constraints are active)
- a set of multiple optima (at least one active constraint)

5.5 Receding Horizon Control Notation

Considering the discrete time system $x(k+1) = Ax(k) + Bu(k)$ and $y(k) = Cx(k)$ with $x(k) \in \mathcal{X}$ and $u(k) \in \mathcal{U}$ for $k \geq 0$, the constrained finite time optimal control problem for a receding horizon is given by:

$$J_{k \rightarrow k+N|k}^*(x(k)) = \min_{U_{k \rightarrow k+N|k}} I_f(x_{k+N|k}) + \sum_{i=0}^{N-1} I(x_{k+i|k}, u_{k+i|k}) \quad (5.53)$$

$$\text{subj. to } x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \quad i = 0, \dots, N-1 \quad (5.54)$$

$$x_{k+i|k} \in \mathcal{X}, u_{k+i|k} \in \mathcal{U}, \quad i = 0, \dots, N-1 \quad (5.55)$$

$$x_{k+N|k} \in \mathcal{X}_f \quad (5.56)$$

$$x_{k|k} = x(k) \quad (5.57)$$

Accordingly, the problem is then solved to find the input sequence $U_{k \rightarrow k+N|k} = \{u_{k|k}, \dots, u_{k+N-1|k}\}$ applied from time k to $k+N$ (instead of going from 0 to N). In this case it is very important to distinguish between the state of the system $x(k)$ at time k and the prediction of the model state at time $k+1$ made at time k $x_{k+1|k}$. These predictions can be obtained by starting at $x_{k|k} = x(k)$ and applying the system model $x_{k+1|k} = Ax_{k|k} + Bu_{k|k}$. Similarly, $u_{k+i|k}$ is the input at time $k+i$ computed at time k . The first element of the optimal solution trajectory $U_{k \rightarrow k+N|k}^* = \{u_{k|k}^*, \dots, u_{k+N-1|k}^*\}$ will be applied to the system:

$$u(k) = u_{k|k}^*(x(k)) \quad (5.58)$$

According to the receding horizon principle, the problem is then solved again at time $k+1$ using the current true state $x_{k+1|k+1} = x(k+1)$, resulting in the following receding horizon control law and closed-loop system:

$$u_{k|k}^*(x(k)) = \kappa_k(x(k)) \Rightarrow x(k+1) = Ax(k) + B\kappa_k(x(k)), \quad k \geq 0 \quad (5.59)$$

In case of time-invariant systems, the resulting control law and closed-loop system will also be time-invariant, what is denoted by the use of $\kappa(x(k))$ instead of $\kappa_k(x(k))$.

6 Invariance

6.1 Objectives of Constrained Control

For some given closed-loop system of the form $x(k+1) = g(x(k), u(k))$ with $x \in \mathcal{X}$ and $u \in \mathcal{U}$, the control problem is to design a controller $u(k) = \kappa(x(k))$ with the following requirements in order of descending importance:

- State and input satisfy the corresponding constraints for all times k

$$\{x(k)\} \subset \mathcal{X}, \{u(k)\} \subset \mathcal{U} \quad (6.1)$$

- The system is (asymptotically) stable $\lim_{k \rightarrow \infty} x(k) = 0$
- Optimize performance
- Maximize the set of initial conditions for which the previous conditions are fulfilled

LQR controllers lead to optimal results, but they often only work for very restricted ranges for initial state. Nonlinear controllers (as e.g. MPC) however can perform well on a significantly larger regions, while still satisfying all state and input constraints (always assuming that the initial state satisfies the constraint $x \in \mathcal{X}$). To ensure that the constraints are satisfied for all times without having to create periodic trajectories, one would have to look infinitely far into the future. As this is not feasible in practice, it is a lot more suitable to consider the set of possible next states that can be reached using a certain set of inputs from the current step.

6.2 Invariance

Given an autonomous system $x(k+1) = g(x(k))$ (some predefined input might already be included in the general non-linear function $g(\cdot)$), some set \mathcal{O} is said to be *positively invariant* if the following condition holds:

$$x(k) \in \mathcal{O} \Rightarrow x(k+1) \in \mathcal{O}, \forall k \in \{0, 1, \dots\} \quad (6.2)$$

This provides a set of states (potentially initial states), from which all trajectories stay within the constraint sets (can easily be proven applying the invariance property iteratively). The largest such set $\mathcal{O}_\infty \subset \mathcal{X}$ is described as *maximal positively invariant set* with respect to \mathcal{X} and includes all other positively invariant sets.

Pre-Sets

Given the same autonomous system $x(k+1) = g(x(k))$ from before, the *pre-set of some set S* is defined as the set of all states that lie within S at the next timestep:

$$\text{pre}(S) := \{x \mid g(x) \in S\} \quad (6.3)$$

Generally, those pre-sets are hard to compute, but e.g. for linear autonomous systems $x(k+1) = Ax(k)$, the pre-set can be simplified significantly when assuming that the set S itself is given by a polyhedron (the hyperplanes of the polyhedron are modulated by the dynamics matrix A):

$$S = \{x \mid Fx \leq f\} \quad \Rightarrow \quad \text{pre}(S) = \{x \mid FAx \leq f\} \quad (6.4)$$

Computation of Invariant Sets

Even though this might seem clear intuitively, it can be shown that a set \mathcal{O} is positively invariant if and only if it is a subset of its pre-set $\mathcal{O} \subseteq \text{pre}(\mathcal{O})$. This also means that $\text{pre}(\mathcal{O}) \cap \mathcal{O} = \mathcal{O}$, what can again be used as a part in an iterative algorithm to find the maximal positively invariant set \mathcal{O}_∞ for some system function g and state constraint set \mathcal{X} (simply cut off non-invariant parts of the set with each iteration until convergence):

```

 $\Omega_0 \leftarrow \mathcal{X}$ 
loop:
   $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$ 
  if  $\Omega_{i+1} = \Omega_i$  then:
    return  $\mathcal{O}_\infty = \Omega_i$ 
  end if
end loop

```

The sequence $\Omega_{i+1} \subseteq \Omega_i$ starts with the maximum possible set satisfying the constraints and converges to the maximal positively invariant set \mathcal{O}_∞ for the given system. To run this algorithm, one has to be able to represent the set Ω_i (often chosen as a polytope), compute its pre-set and the intersection and perform an equality test on the sets to check when to terminate.

6.3 Control Invariance

A set of states $\mathcal{C} \subseteq \mathcal{X}$ is said to be *control invariant* if there always exists an input sequence (generated by some controller), satisfying the input constraints and leading to a closed-loop trajectory that stays within \mathcal{C} forever:

$$x(k) \in \mathcal{C} \quad \Rightarrow \quad \exists u(k) \in \mathcal{U} \text{ s.t. } g(x(k), u(k)) \in \mathcal{C}, \quad \forall k \in \mathbb{N}^+ \quad (6.5)$$

Similar to before, the *maximal control invariant set* \mathcal{C}_∞ for the closed-loop system $x(k+1) = g(x(k), u(k))$ and constraints $x \in \mathcal{X}$, $u \in \mathcal{U}$ is the largest control invariant set (containing all the others). For all states $x \in \mathcal{C}_\infty$ a control law can be found that results in the input sequence $u(k)$ and the closed-loop satisfying the constraints for all future times (finding this control law is actually the best any controller can do).

Computation of Control Invariant Sets

The notion of pre-sets can easily be extended to the case of a system with input:

$$\text{pre}(S) = \{x \mid \exists u \in \mathcal{U} \text{ s.t. } g(x, u) \in S\} \quad (6.6)$$

As all arguments from common invariant sets still apply, a set \mathcal{C} is a control invariant set if and only if $\mathcal{C} \subseteq \text{pre}(\mathcal{C})$ (using the pre-set definition that was just introduced). This can again be formulated as an iterative algorithm with the important difference that the computation of the pre-set just became significantly harder:

```

 $\Omega_0 \leftarrow \mathcal{X}$ 
loop:
   $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$ 
  if  $\Omega_{i+1} = \Omega_i$  then:
    return  $\mathcal{C}_\infty = \Omega_i$ 
  end if
end loop

```

Considering once more the linear case $x(k+1) = Ax(k) + Bu(k)$ as a simplification with the polyhedral constraint sets $\mathcal{U} = \{u \mid Gu \leq g\}$ and $\mathcal{X} = \{x \mid Fx \leq f\}$, integrating the system dynamics leads to a closed-form expression for the pre-set:

$$\text{pre}(S) = \left\{ x \mid \exists u, \begin{bmatrix} FA & FB \\ 0 & G \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} f \\ g \end{bmatrix} \right\} \quad (6.7)$$

This actually corresponds to a projection operation of the multi-dimensional polyhedron onto the state space, resulting in the set of states for which the combined tuple (state,input) lies within the constraint polyhedron of points lying within \mathcal{U} and \mathcal{X} .

When carrying out the computation (even in the linear case), it is important to consider that through the choice of an arbitrary input (the criterion only states that there exists some input), an entire set of points in a pre-set can be mapped into a single point. This usually results in a significantly larger pre-set that is also harder to compute.

Obtaining a Control Law

Once a control invariant set \mathcal{C} has been computed for the system $x(k+1) = g(x(k), u(k))$, a control law $\kappa(x(k))$ needs to be found so that it can be implemented on the system. As the system has to satisfy the constraints for all times, the control law can be synthesized through an optimization problem (using any function $f(x, u)$ including $f(x, u) = 0$):

$$\kappa(x) = \arg \min \{ f(x, u) \mid g(x, u) \in \mathcal{C} \} \quad (6.8)$$

The resulting control law only guarantees that the input and state constraints will be satisfied for all future times, but does not state anything regarding convergence. Even though this might seem like a very practical approach to obtain control laws satisfying the constraints for all times, they are very often too complex to be computed explicitly.

6.4 Invariant Sets and MPC

Invariant sets are a core component of MPC problems and useful, but at the same time very complex and difficult to compute. The special case of linear systems with polyhedral constraints (as shortly mentioned in the previous section) can lead to the following invariant sets:

- *Polyhedral invariant sets* can represent the maximum invariant set, but can become very complex for many states (especially if somehow round boundaries are part of the set). The resulting MPC problem will turn out to be a quadratic program.
- *Ellipsoidal invariant sets* are smaller than the polyhedral set and can therefore not describe the maximum positively invariant set in general cases. However, they are easy to compute even for large dimensions and defined by a single equation (fixed complexity). The resulting MPC problem will turn out to be a quadratically constrained quadratic program.

6.5 Practical Computation of Invariant Sets

As mentioned, the computation of invariant sets significantly simplifies when polytopes or ellipsoids are used as constraint sets. This section dives a bit deeper into the manipulations that can be used to simplify the computations and make them efficient.

Polytopes

As a reminder, polytopes are general polyhedra that are bounded and can be described by a set of hyperplane constraints of the form $P = \{x \mid \leq Ax \leq b\}$. Another equivalent notation for polytopes can be found using *convex hulls*. For any subset $S \subset \mathbb{R}^d$, the convex hull $\text{co}(S)$ is defined as the intersection of all convex sets containing S , which means that it is the smallest convex set containing S . Formally, a convex hull for a set of vectors $\{v_1, \dots, v_k\} \in \mathbb{R}^d$ can be defined as the following set:

$$\text{co}(\{v_1, \dots, v_k\}) = \left\{ x \mid x = \sum_i \lambda_i v_i, \lambda_i \geq 0, \sum_i \lambda_i = 1, \forall i = 1, \dots, k \right\} \quad (6.9)$$

For every polytope there exists such a finite set of vectors for which $P = \text{co}(v_1, \dots, v_k)$ holds, which results in another completely equivalent description for polytopes besides the one using inequality constraints. For MPC however, the form of inequality constraints is more convenient as it simplifies the intersection and the equality test for the invariant set computation algorithm significantly. The most common polytopic constraints include the following for a linear system $x(k+1) = Ax(k) + Bu(k)$ with $y(k) = Cx(k)$:

$$\begin{aligned} u_{low} \leq u(k) \leq u_{high} \\ y_{low} \leq y(k) \leq y_{high} \end{aligned} \iff \begin{bmatrix} 0 & -I \\ 0 & I \\ -C & 0 \\ C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \leq \begin{bmatrix} -u_{low} \\ u_{high} \\ -y_{low} \\ y_{high} \end{bmatrix} \quad (6.10)$$

Similarly, *rate constraints* (maximum change between two subsequent states) can be expressed in this inequality form:

$$\|x(k) - x(k+1)\|_\infty \leq \alpha \iff \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \end{bmatrix} \leq \mathbf{1}\alpha \quad (6.11)$$

Magnitude constraints on the state can be expressed as follows:

$$\|Cx(k)\|_\infty \leq \alpha \iff \begin{bmatrix} C \\ -C \end{bmatrix} x(k) \leq \mathbf{1}\alpha \quad (6.12)$$

The intersection of two polytopes, that are given in the described inequality form, can be computed very easily by just stacking the corresponding constraints:

$$\begin{aligned} S = \{x \mid Cx \leq c\} \\ T = \{x \mid Dx \leq d\} \end{aligned} \rightarrow S \cap T = \left\{ x \mid \begin{bmatrix} C \\ D \end{bmatrix} x \leq \begin{bmatrix} c \\ d \end{bmatrix} \right\} \quad (6.13)$$

The last step of the invariant set computation requires to check, whether two sets are equal or not. This is obviously equivalent to both sets being subsets of each other. Formally, this subset test requires that for two polytopes $P = \{x \mid Cx \leq c\}$ and $Q = \{x \mid Dx \leq d\}$, $P \subseteq Q$ holds if the following holds for each row D_i or D :

$$P \subseteq \{x \mid D_i x \leq d_i\} \quad (6.14)$$

So that this condition does not have to be checked for every point in P , a *support function* h_P can be defined:

$$h_P(D_i) := \max_x D_i x \quad \text{s.t. } Cx \leq c \quad (6.15)$$

For any given constraint D_i , this support function describes the point that is the maximum in the direction of the hyperplane D_i , while still being part of the polytope P (as shown in figure 6.1). The subset check for each inequality constraint then simplifies to check if the single point obtained from the support function satisfies the constraint D_i , i.e. $h_P(D_i) \leq d_i$. If this condition is fulfilled for all constraints D_i , the set P is a subset of Q .

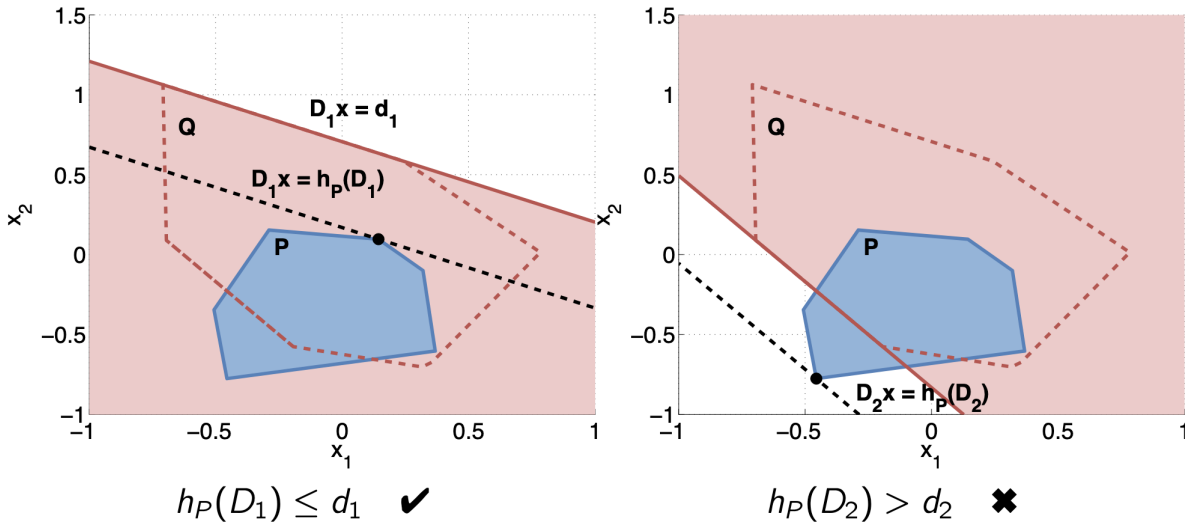


Figure 6.1: Subset test using support functions (lecture slides)

Ellipsoids

An ellipsoid can be easily represented by a symmetric and positive-definite function $P \in \mathbb{R}^{n \times n}$ and some vector $x_c \in \mathbb{R}^n$ as the following set:

$$E = \{x \mid (x - x_c)^T P (x - x_c) \leq 1\} \quad (6.16)$$

Sets of this form are especially useful as containment can be checked using the single inequality above and the complexity is always quadratic in the dimension (not the case for polyhedra). Keeping the quadratic definition in mind, it can be shown that every sublevel set of the following definition for some Lyapunov function $V : \mathbb{R}^n \rightarrow \mathbb{R}$ and system $x(k+1) = g(x(k))$ is invariant (intuitively clear because the system state has to decay towards the origin over time and will not leave the set):

$$Y := \{x \mid V(x) \leq \alpha\} \quad (6.17)$$

When searching for the largest invariant set satisfying the constraints \mathcal{X} , it might be a good approximation to maximize α in order to find the largest ellipsoidal invariant set satisfying the constraints:

$$Y_\alpha := \{x \mid V(x) \leq \alpha\} \subseteq \mathcal{X} \quad (6.18)$$

Considering the linear system $x(k+1) = Ax(k)$ satisfying the condition $A^T P A - P < 0$ for some positive definite P , then $V(x(k)) = x(k)^T P x(k)$ is a Lyapunov function and the largest invariant set Y_α can be found by maximizing the following set (assuming polyhedral constraints):

$$Y_\alpha = \{x \mid x^T P x \leq \alpha\} \subseteq \mathcal{X} = \{x \mid Fx \leq f\} \quad (6.19)$$

The maximization over all possible α satisfying the constraints can be written using the support function (i.e. the points that are the closest to the constraint within the candidate set):

$$\max_{\alpha} \alpha \quad (6.20)$$

$$\text{s.t. } h_{Y_\alpha}(F_i) \leq f_i \quad (6.21)$$

In case of an ellipse, the support function can be written as $h_{Y_\alpha}(F_i) = \max_x F_i x$ subject to the ellipse constraint $x^T P x \leq \alpha$. Transforming the ellipse to a ball, using the substitution $y = P^{1/2} x$, leads to an explicit solution for the support function:

$$h_{Y_\alpha}(F_i) = F_i P^{-1/2} \frac{P^{-1/2} F_i^T}{\|P^{-1/2} F_i^T\|_2} \sqrt{\alpha} = \|P^{-1/2} F_i^T\|_2 \sqrt{\alpha} \quad (6.22)$$

Finding the largest ellipse inside the polytope constraint set has now become a one-dimensional optimization problem:

$$\alpha^* = \max_{\alpha} \alpha, \quad \|P^{-1/2} F_i^T\|^2 \alpha \leq f_i^2, \quad \forall i \in \{1, \dots, n\} \quad (6.23)$$

$$= \min_{i \in \{1, \dots, n\}} \frac{f_i^2}{F_i P^{-1} F_i^T} \quad (6.24)$$

Note that this is often not the maximum invariant set. The result might be improved by choosing different matrices P for the Lyapunov function and therefore rotating the ellipse in the state space. However, even in this case the maximum invariant set might not be such that it can be approximated properly by an ellipse.

7 Feasibility and Stability

As already shown in previous chapters, optimal finite horizon control (and therefore also receding horizon control) might not necessarily lead to stable results of the infinite horizon control problem and true state trajectories might deviate strongly from their predicted ones. However, when accepting some trade-offs, it is possible to meet multiple desirable objectives at once using a predictive control framework. These main objectives are expressed for the following system and control law $u(k) = \kappa(x(k))$ that is to be designed:

$$x(k+1) = g(x(k), u(k)), \quad x, u \in \mathcal{X}, \mathcal{U} \quad (7.1)$$

- Satisfy constraints: $\{x(k)\} \subset \mathcal{X}$ and $\{u(k)\} \subset \mathcal{U}$
- Stable system: $\lim_{k \rightarrow \infty} x(k) = 0$
- Optimize performance
- Maximize the set of initial conditions $\{x(0)\}$ where the constraints are satisfied, the system is stable and performance is optimized.

As also discussed multiple times up to this point, the optimal solution would be to solve the constrained infinite time optimal control problem and choose a stage cost to optimize correspondingly over short- and long-term benefits by influencing the trajectories. However, this problem cannot be solved due to the infinite number of variables that would be required for that. Accordingly, the truncated constrained finite time optimal control problem with time horizon N as stated in the beginning of the previous chapter has to be solved. The information that is lost due to the truncation has to be approximated as well as possible with the terminal cost $I(x_{k+N|k})$, approximating the tail of the cost, and the terminal constraint set \mathcal{X}_f , approximating the tail of the constraints. The following sections will focus on how to choose those constraints to meet the mentioned objectives as well as possible.

7.1 Key Points of MPC

The key points that can be found when comparing the performance of LQR control in the presence of state and input constraints (e.g. saturation) and MPC control are the following:

- Applying LQR control and saturating the controller can lead to instability
- If rate constraints are not considered during MPC (but only saturation and state constraints), the system will be stable but might still not converge
- If all constraints on the actuator and the rate constraint are considered, MPC results in a stable closed-loop system and is efficient with respect to the required control inputs
- Shorter prediction horizon lengths for MPC can lead to a loss of the stability and convergence properties

7.2 Loss of Feasibility and Stability in MPC

For the remainder of this chapter, the following notation of the MPC problem will be used:

$$J^*(x(k)) = \min_U x_N^T P x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \quad (7.2)$$

$$\text{subj. to } x_{i+1} = A x_i + B u_i, \quad i = 0, \dots, N-1 \quad (7.3)$$

$$x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, N-1 \quad (7.4)$$

$$x_N \in \mathcal{X}_f \quad (7.5)$$

$$x_0 = x(k) \quad (7.6)$$

By just applying standard MPC to a problem, there are no guarantees with respect to stability and/or feasibility. This means that the MPC optimization problem might not have a solution satisfying all constraints and even if a solution exists, the trajectory might not converge (due to the finite horizon length). Similarly, depending on the choice of cost function, the finite horizon MPC solution might apply inputs that cause the closed-loop state trajectory to run out of the set of feasible initial states (yielding an infeasible optimization problem). However, it might be possible to tune the horizon length N alongside the input cost R to obtain a trajectory that does not become infeasible or leads to an unstable closed-loop system and still remains tractable.

Concluding, the finite horizon length causes a deviation between the closed-loop trajectory and open-loop predictions, potentially leading to instability and infeasibility. In the ideal case of an infinite horizon MPC solution, the set of feasible open-loop predictions and closed-loop trajectories would actually coincide (or ‘large enough’ horizon lengths). The main problem of MPC is then to design the cost functions and choose a sufficiently large finite horizon length to achieve all mentioned key points.

Comparing infinite and finite horizon solutions to the receding horizon problem lead to the following conclusions:

- For infinite horizons, the open-loop trajectories coincide with the closed-loop trajectories (as for LQR control).
- If the problem is feasible, the infinite horizon solution and the corresponding closed-loop trajectories will always be feasible.
- For infinite-horizon solutions with finite cost, the states and inputs will converge asymptotically to the origin.
- In the finite horizon case, the ‘short-sightedness’ yields a bad approximation for the infinite horizon controller and might lead the solved system to become infeasible after some steps.
- Due to the finite nature of the cost summation, it is possible that the cost is finite, but the states and generated control inputs still do not converge to the origin.

7.3 Feasibility and Stability Guarantees in MPC

Given some non-linear and time-invariant discrete-time system $x(k+1) = g(x(k))$ with an equilibrium point $\bar{x} = g(\bar{x})$, this equilibrium point $x \in \Omega$ is *asymptotically stable* in the positive invariant set $\Omega \subseteq \mathbb{R}^n$ if it is Lyapunov stable and *attractive* (Ω is then described as the region of attraction as all initial conditions from within it yield asymptotic stability):

$$\lim_{k \rightarrow \infty} \|x(k) - \bar{x}\| = 0, \quad \forall x(0) \in \Omega \quad (7.7)$$

If the equilibrium point is now asymptotically stable and the region of attraction is $\Omega = \mathbb{R}^n$, it is described as *globally asymptotically stable*.

The previously introduced concept of *Lyapunov functions* can also be extended to be defined for the equilibrium point $\bar{x} = 0$ (without limit of generality) and the closed and bounded positive invariant set $\Omega \subset \mathbb{R}^n$, by defining the function $V : \mathbb{R}^n \rightarrow \mathbb{R}$ that is continuous as $x = 0$, finite for all $x \in \Omega$ and fulfilling the following criteria:

$$V(0) = 0, \quad V(x) > 0, \quad \forall x \in \Omega \setminus \{0\} \quad (7.8)$$

$$V(g(x)) - V(x) \leq -\alpha(x), \quad \alpha(x) > 0, \quad \forall x \in \Omega \setminus \{0\} \quad (7.9)$$

Recalling the stability properties of Lyapunov functions, it can be shown that if a system admits a Lyapunov function $V(x)$ then $x = 0$ is *asymptotically stable* in Ω . The proof can be split up into a proof for recursive feasibility (the system remains feasible at all times when starting from a feasible initial condition) and stability (the optimal cost function is a Lyapunov function), which can both be found in the lecture slides. Recursive stability can be guaranteed by choosing a suitable terminal constraint set \mathcal{X}_f , while stability can be guaranteed by ensuring that the cost function $J^*(x(k))$ is a Lyapunov function. With this convergence property, it is sufficient for the state trajectory to eventually enter the set \mathcal{X}_f for recursive feasibility, as it might not stay at this first point inside the set necessarily, but it will never leave the set and eventually converge using a corresponding local control law.

This leads to the conclusion that in case of a *linear system with quadratic cost function* (as given in the previous subsections of this chapter), the terminal cost $x_N^T P x_N$ should result in the cost function being a Lyapunov function and the terminal constraint $x_N \in \mathcal{X}_f$ should be an invariant set (i.e. the system stays there). One possible solution is the unconstrained LQR control law:

$$F_\infty = -(B^T P_\infty B + R)^{-1} B^T P_\infty A \quad (7.10)$$

Where the terminal weight P_∞ can be found through the discrete-time algebraic Riccati equation:

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B (B^T P_\infty B + R)^{-1} B^T P_\infty A \quad (7.11)$$

The corresponding terminal set \mathcal{X}_f should be chosen as the maximum invariant set for the closed-loop system (such that all state and input constraints are satisfied):

$$x_{k+1} = Ax_k + BF_\infty x_k \in \mathcal{X}_f, \quad \mathcal{X}_f \subseteq \mathcal{X}, \quad F_\infty x_k \in \mathcal{U}, \quad \forall x_k \in \mathcal{X}_f \quad (7.12)$$

Choosing quadratic matrices for the stage cost, it is a positive definite function, which yields a continuous Lyapunov function in the terminal set \mathcal{X}_f in combination with the terminal cost $x_N^T P_\infty x_N$:

$$x_{k+1}^T P_\infty x_{k+1} - x_k^T P_\infty x_k = \dots = -x_k^T (Q + F_\infty^T R F_\infty) x_k \quad (7.13)$$

Furthermore, the terminal set is invariant under the local control law $\kappa_f(x) = F_\infty x$ by construction. With this, all assumptions for the feasibility and stability theorem are verified and the system is both stable and feasible for all initial conditions. Overall, the following statements on the terminal constraints (terminal set and cost) can be made:

- The terminal constraints provide a sufficient condition for feasibility and stability
- MPC with terminal constraints might lead to a reduction of the region of attraction
- $\mathcal{X}_f = 0$ is the simplest choice of a terminal state constraint, but yields a small region of attraction for small horizon lengths N
- Analytical solutions (ARE and LQR control) are available for linear systems with quadratic cost
- With larger horizon lengths, the region of attraction approaches the maximum control invariant set (done in practice and stability checked by sampling).
- Concepts extend to nonlinear MPC, but computing \mathcal{X}_f and I_f can be very difficult.

8 Practical Issues in MPC

8.1 Common Tasks for MPC

- Beside the classical problem for MPC that is to steer the system to the origin, a common practical task is to *track a given trajectory* with the output values.
- As disturbances might cause the system to include a certain offset, it is important that MPC should also be able to *reject disturbances* to ensure correct convergence.
- As a final limiting factor during MPC controller design, the *feasible set* is important to consider as it represents constraints on the system that have to be fulfilled at all times. It should therefore be as large as possible to leave more room for optimization.

8.2 Reference Tracking

To formulate the reference tracking problem, a linear system with dynamics $x(k+1) = Ax(k) + Bu(k)$, $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ is considered with similar polytopic constraint sets \mathcal{X} and \mathcal{U} as before:

$$\mathcal{X} = \{x \mid G_x x \leq h_x\}, \quad \mathcal{U} = \{u \mid G_u u \leq h_u\} \quad (8.1)$$

The goal is then for a function of the state $z(k) = Hx(k)$, which is not necessarily the output, to track a constant (or at least piece-wise constant) reference $r \in \mathbb{R}^{n_r}$ as time goes to infinity. In order to track a reference with MPC, the target r needs to be transformed to state space and the cost function as well as the terminal set constraint need to be reconsidered and possibly adapted accordingly.

Steady-State Tracking Problem

The target state x_s , that corresponds to the reference r which should be tracked, needs to be a fixed point of the system dynamics $x_s = Ax_s + Bu_s$ (in order for the system not to leave this point anymore once it got there) and needs to fulfill $r = Hx_s$. In matrix form, these conditions read as follows:

$$\begin{bmatrix} I - A & B \\ H & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \quad (8.2)$$

In general, it will be assumed that the tracking problem is feasible and that a steady-state solution (x_s, u_s) exists. If multiple inputs u_s exist for some steady state solution $r = Hx_s$, the one with minimum cost is chosen:

$$\min_{u_s} u_s^T R_s u_s \quad (8.3)$$

$$\text{subj. to } \begin{bmatrix} I - A & B \\ H & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \quad (8.4)$$

$$x_s \in \mathcal{X}, \quad u_s \in \mathcal{U} \quad (8.5)$$

If no solution results, which allows to exactly track the reference, the point inside the feasible set closest to r is chosen:

$$\min_{(x_s, u_s)} (Hx_s - r)^T Q_s (Hx_s - r) \quad (8.6)$$

$$\text{subj. to } x_s = Ax_s + Bu_s \quad (8.7)$$

$$x_s \in \mathcal{X}, \quad u_s \in \mathcal{U} \quad (8.8)$$

Reference Tracking with MPC

In general, the optimization problem for reference tracking can be expressed through the following expression, driving the steady state solution (x_s, u_s) as close as possible to r , subject to the usual constraints:

$$\min_U \|z_N - Hx_s\|_{P_z}^2 + \sum_{i=0}^{N-1} \|z_i - Hx_s\|_{Q_z}^2 + \|u_i - u_s\|_R^2 \quad (8.9)$$

In case of a linear system, the simple coordinate transforms $\Delta x = x - x_s$ and $\Delta u = u - u_s$ can be introduced so that the modified system can just be regulated to the new origin under the following constraints (assuming polytopic state and input constraints as before):

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k \quad (8.10)$$

$$G_x \Delta x \leq h_x - G_x x_s \quad (8.11)$$

$$G_u \Delta u \leq h_u - G_u u_s \quad (8.12)$$

The MPC regulation problem of the transformed problem can then directly be written as:

$$\min_{\Delta U^*} \sum_{i=0}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i + V_f(\Delta x_N) \quad (8.13)$$

$$\text{subj. to } \Delta x_0 = \Delta x(k) = x(k) - x_s \quad (8.14)$$

$$\Delta x_{i+1} = A\Delta x_i + B\Delta u_i \quad (8.15)$$

$$G_x \Delta x_i \leq h_x - G_x x_s \quad (8.16)$$

$$G_u \Delta u_i \leq h_u - G_u u_s \quad (8.17)$$

$$\Delta x_N \in \mathcal{X}_f \quad (8.18)$$

Note that the input that is applied to the system will not be the first component of the resulting ΔU^* , but the back-transformed $u_0^* = \Delta u_0^* + u_s$. Regarding convergence, the steady-state solution for some feasible target $r = Hx_s$ with $(x_s, u_s) \in \mathcal{X} \times \mathcal{U}$ will be reached, if the terminal weight V_f and terminal set constraint \mathcal{X}_f were chosen accordingly. The conditions they need to fulfill include the ones previously stated for regulation and an additional one requiring the steady-state solution of the modified system to satisfy state and input constraints:

$$\mathcal{X}_f \subseteq \mathcal{X}, \quad Kx \in \mathcal{U}, \quad \forall x \in \mathcal{X}_f \quad (8.19)$$

$$V_f(x(k+1)) - V_f(x(k)) \leq -I(x(k), Kx(k)), \quad \forall x \in \mathcal{X}_f \quad (8.20)$$

$$x_s \oplus \mathcal{X}_f \subseteq \mathcal{X}, \quad K\Delta x + u_s \in \mathcal{U}, \quad \forall x \in \mathcal{X}_f \quad (8.21)$$

Formally, the closed-loop system then converges to the target $x(k) \rightarrow x_s$ and $z(k) = Hx(k) \rightarrow r$ for $k \rightarrow \infty$. In other words, if the modified closed-loop system is stable, the set point will be achieved and stability can be guaranteed using the same tools as for regulation before. However, choosing a large terminal set to ensure fast convergence of MPC (still with corresponding

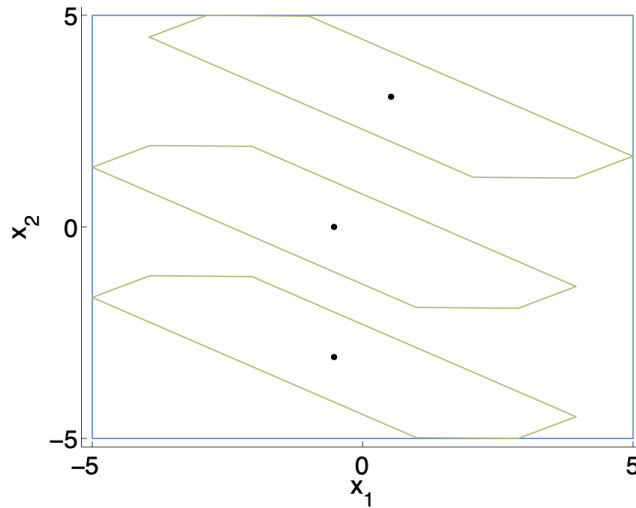


Figure 8.1: Issue with shifted terminal set (lecture slides)

guarantees) for the common problem of regulating a system to its origin, yields problems as the set of feasible targets for which this terminal set is not outside of the state constraints might be very small as shown in figure 8.1.

Using the property that scaled versions of invariant sets $\mathcal{X}_f^{scaled} = \alpha \mathcal{X}_f$ are also invariant set for linear systems, the terminal set can simply be scaled down when the targeted reference moves towards the boundaries of the feasible regions (as shown in figure 8.2) to create a much larger set of feasible targets. Using this trick, all tracking targets within the constraints $(x_s, u_s) \in \mathcal{X} \times \mathcal{U}$ become feasible with respect to those, given that they also fulfill the steady-state condition, which again impose limitations on the points in state space that can be tracked.

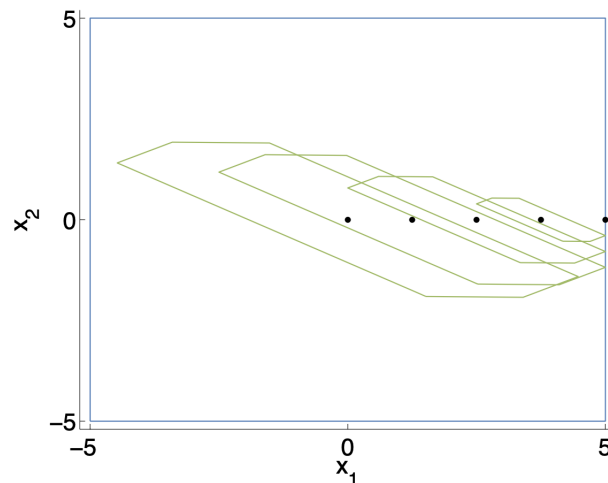


Figure 8.2: Rescaled terminal set to enlarge feasible target set (lecture slides)

Remaining problems for reference tracking at this point include the following:

- The terminal set is a function of the chosen target and shrinks / grows accordingly
- Slightest reference changes can render the problem infeasible
- Model errors or disturbances will cause the solution to have an offset
- State estimators are required to track some reference if the strong assumption of having access to all states does not hold (i.e. if the measurements only cover parts of the states).

Reference Tracking without Offset with MPC

As disturbances would typically lead to offsets in the solution of the MPC problem, they have to be rejected to obtain a proper result when controlling such systems. For this section, the disturbance $d \in \mathbb{R}^{n_d}$ is assumed to be constant so that the augmented model reads as follows:

$$x_{k+1} = Ax_k + Bu_k + B_d d_k \quad (8.22)$$

$$d_{k+1} = d_k \quad (8.23)$$

$$y_k = Cx_k + C_d d_k \quad (8.24)$$

The augmented system is then observable if and only if (A, C) is observable, fulfilling the following condition:

$$\text{rank} \begin{bmatrix} A - I & B_d \\ C & C_d \end{bmatrix} = n_x + n_d \quad (8.25)$$

Accordingly, the maximum dimension of the disturbance is $n_d \leq n_y$ and $n_d = n_y$ will be assumed in the following without limit of generality. This correlation has to hold as d_s has to be uniquely defined by y_s at steady-state. To obtain estimates for state \hat{x} and disturbance \hat{d} , a state observer, based on the measurements $y(k)$, can be used (linear estimator; alternatively also a Kalman estimator could be used):

$$\begin{bmatrix} \hat{x}(k+1) \\ \hat{d}(k+1) \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}(k) \\ \hat{d}(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} L_x \\ L_d \end{bmatrix} (-y(k) + C\hat{x}(k) + C_d\hat{d}(k)) \quad (8.26)$$

The design parameters L_x and L_d have to be chosen such that the following error dynamics are asymptotically stable. This guarantees that the state and disturbance estimates converge to their true values:

$$\begin{bmatrix} x(k+1) - \hat{x}(k+1) \\ d(k+1) - \hat{d}(k+1) \end{bmatrix} = \left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} x(k) - \hat{x}(k) \\ d(k) - \hat{d}(k) \end{bmatrix} \quad (8.27)$$

If the observer is asymptotically stable and $n_d = n_y$ holds, then y_∞ and u_∞ are the measured steady-state outputs and inputs and the steady-state observer output $C\hat{x}_\infty + C_d\hat{d}_\infty$ perfectly tracks y_∞ (without offset):

$$\begin{bmatrix} A - I & B \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_\infty \\ u_\infty \end{bmatrix} = \begin{bmatrix} -B_d\hat{d}_\infty \\ y_\infty - C_d\hat{d}_\infty \end{bmatrix} \quad (8.28)$$

As the measurements $z(k) = Hy(k)$ are a function of the state estimate $y(k)$ and the disturbance has to be included in the new conditions at steady-state, they read as follows:

$$x_s = Ax_s + Bu_s + B_d\hat{d}_\infty \quad (8.29)$$

$$z_s = H(Cx_s + C_d\hat{d}_\infty) = r \quad (8.30)$$

Both the system state and the target at steady state are modified to include the disturbance influences. While the steady-state estimate of the disturbance \hat{d}_∞ (which is equal to the true disturbance assuming constant values) is not available in reality, the best approximation for it is the current estimate \hat{d} . This leads to the following formulation, to which the common regulation techniques can be applied (modify the target to account for the effect of disturbance on the tracked variables):

$$\begin{bmatrix} A - I & B \\ HC & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} -B_d\hat{d} \\ r - HC_d\hat{d} \end{bmatrix} \quad (8.31)$$

Summarizing, the steps that have to be carried out at each sampling instance are the following:

1. Estimate state and disturbance \hat{x} and \hat{d}
2. Obtain the steady-state x_s and u_s from equation 8.31
3. Solve the MPC problem using the disturbance estimate \hat{d} and apply the first component of the resulting input sequence U in the sense of the receding horizon control:

$$\min_U \sum_{i=0}^{N-1} (x_i - x_s)^T Q (x_i - x_s) + (u_i - u_s)^T R (u_i - u_s) + V_f(x_N - x_s) \quad (8.32)$$

$$\text{subj. to } x_0 = \hat{x}(k) \quad (8.33)$$

$$d_0 = \hat{d}(k) \quad (\text{new}) \quad (8.34)$$

$$x_{i+1} = Ax_i + Bu_i + B_d d_i, \quad i = 0, \dots, N \quad (\text{modified}) \quad (8.35)$$

$$d_{i+1} = d_i, \quad i = 0, \dots, N \quad (\text{new}) \quad (8.36)$$

$$x_i \in \mathcal{X}, \quad u_i \in \mathcal{U} \quad (8.37)$$

$$x_N - x_s \in \mathcal{X}_f \quad (8.38)$$

The main result is then that under the assumptions of $n_d = n_y$, the closed-loop system converges and that the target steady-state problem is feasible with no constraints being active at steady-state, the target is reached without offset (i.e. the measurements $z(k) = Hy(k) \rightarrow r$ for $k \rightarrow \infty$) and the estimator converges to the true state and disturbance values $\hat{x}(k) \rightarrow \hat{x}_\infty$ and $\hat{d}(k) \rightarrow \hat{d}_\infty$. Denoting the input applied to the system with $u_0^* =: \kappa(\hat{x}(k), \hat{d}(k), r)$, the true state as well as the estimates for state and disturbances are:

$$x(k+1) = Ax(k) + B\kappa(\hat{x}(k), \hat{d}(k), r) + B_d d \quad (8.39)$$

$$\hat{x}(k+1) = (A + L_x C)\hat{x}(k) + (B_d + L_d C_d)\hat{d}(k) + B\kappa(\hat{x}(k), \hat{d}(k), r) - L_x y(k) \quad (8.40)$$

$$\hat{d}(k+1) = L_d C \hat{x}(k) + (I + L_d C_d)\hat{d}(k) - L_d y(k) \quad (8.41)$$

8.3 Enlarging the Feasible Set

As described in the previous section, the invariant set for the tracking problem can be found by translating and potentially scaling the maximum invariant set towards its tracking target. This yields significantly smaller invariant sets for some references close towards the boundary of the feasible set. While this works in principle, reducing the invariant set to stay within the state constraints is not very attractive for fast convergence, etc. To enlarge this feasible set again, MPC methods without terminal constraints and soft-constrained MPC methods are distinguished.

MPC without Terminal Constraint

In order for the closed-loop system to be stable under the control law u_0^* , the system $x(k+1) = Ax(k) + Bu_0^*(x(k))$ needs to be invariant in the feasible set (i.e. all constraints are satisfied in \mathcal{X}_f and terminal cost is a continuous Lyapunov function in the terminal set \mathcal{X}_f (i.e. the system converges to the origin):

$$I_f(x_{i+1}) - I_f(x_i) \leq -I(x_i, \kappa_f(x_i)), \quad \forall x_i \in \mathcal{X}_f \quad (8.42)$$

Comparing the feasible sets, it is intuitive to see that the feasible set with terminal constraint is usually smaller than the feasible set without such constraints and that the terminal set has

to be chosen to be a subset of the one with terminal constraints. This terminal set potentially adds a large number of extra constraints or even introduces constraints onto the state where there have been none before (e.g. only input constraints). One approach to increase the feasible set, could be to design an MPC optimization problem without terminal constraint, which still guarantees stability. However, note that such a feasible set without the constraints is generally not invariant.

It can in fact be shown that given that the initial state lies in a sufficiently small subset of the feasible set and the horizon length N is sufficiently large, the terminal constraint can be removed without changing the solution (i.e. the terminal constraint is automatically satisfied). In this case, the solution to the finite horizon MPC problem will actually coincide with the infinite horizon solution. Even though this is a possibility to increase the feasible set for the controller (as the terminal set only was a sufficient condition for stability), it comes with the task to choose the correct set of initial states and the corresponding horizon length, what is generally hard. In practice, the initial set is approximated by sampling different values and increasing the horizon length N simultaneously (the region of attraction approaches the control invariant set).

Soft Constrained MPC

Another possibility to increase the set of states for which the MPC problem is feasible (i.e. there exists an input trajectory steering the system to the target), is to introduce *soft constraints*. While constraints that originate from physical limitations or actuator constraints (e.g. saturation) are usually hard, other constraints that were imposed (most state/output constraints e.g. for comfort) might also be slightly violated without significant consequences. As every hard constraint limits the feasible set, even in case of a stable system, state constraints are often softened in practical applications, with the goal still being to minimize the duration and size of the constraint violation by augmenting the cost function accordingly. Note that the duration and the magnitude of the violation might be anti-proportional to each other and that solutions include some trade-off between them.

For these multi-objective problems, pareto-optimal size-duration curves can be drawn for some given system and horizon as shown in figure 8.3. The arrow denotes different initial conditions and the corresponding effect on the curves. The best operating points always lie on these curves as values below them cannot be achieved and above them, improvements in at least one of the dimensions would be possible. However, these solutions are generally difficult to find and need to be chosen depending on the application.

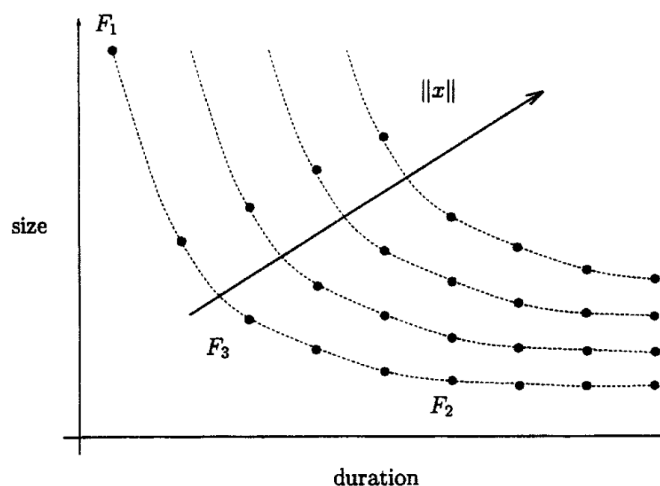


Figure 8.3: Pareto optimal curves for soft constraint violations (lecture slides)

To formulate the MPC problem with soft constraints, so called *slack variables* ϵ_i (vectors for each state $i = 1, \dots, N$) are introduced for each one of the soft constraints that measure the violation of the polytopic constraint and allow to penalize it accordingly in the cost function with $I_\epsilon(\epsilon_i)$. Note that these slack variables are zero, if a constraint is satisfied exactly and that in the following formulation only the state constraints are softened:

$$\min_u \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + I_\epsilon(\epsilon) + x_N^T P x_N + I_\epsilon(\epsilon_N) \quad (8.43)$$

$$\text{s.t. } x_{i+1} = A x_i + B u_i \quad (8.44)$$

$$H_x x_i \leq k_x + \epsilon_i \quad (8.45)$$

$$H_u u_i \leq k_u \quad (8.46)$$

$$\epsilon_i \geq 0 \quad (8.47)$$

In order for the non-modified problem to have a feasible solution z^* without constraint violations, the relaxed problem obviously should have the same solution z^* and additionally ϵ_0 should be as small as possible. The main question that still remains is how the additional cost function term $I_\epsilon(\epsilon_i)$ should be chosen to allow this described connection between the solutions and feasibility. To illustrate the effects of different penalties, the constraint function $g(z) = z - z^* \leq 0$ will be considered with the original cost function $f(z)$ (leading to the minimizer z^* that violates no constraint). Using the following quadratic penalty for constraint violation, yields the result shown in figure 8.4 and a shift of the minimum towards $(z, \epsilon) = (z^* + \epsilon^*, \epsilon^*)$ for the augmented cost function $f(z) + I_\epsilon(\epsilon)$ away from $(z^*, 0)$:

$$I_\epsilon(\epsilon) = \begin{cases} s \cdot \epsilon^2, & \epsilon \geq 0 \\ 0, & \epsilon < 0 \end{cases} \quad (8.48)$$

Intuitively, the reason for this shift is that the penalty function has a gradient of zero at the constraint boundary and marginal violations are therefore penalized very insignificantly and directly compensated by the loss decrease on the original cost function. While the optimum point of the augmented problem goes towards z^* for $s \rightarrow \infty$, this point is never actually reached with final numbers and there will always be a slight violation of the constraint.

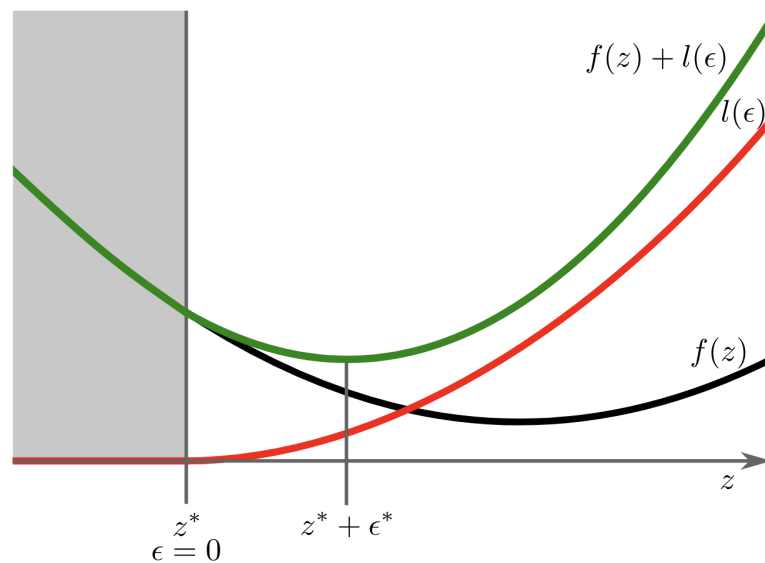


Figure 8.4: Issue with shifted terminal set (lecture slides)

Using the following simple linear penalty function instead (with potential differentiation issues of the augmented cost function), avoids this shift and the augmented cost function achieves its minimum still at $(z^*, 0)$, under the condition that the gradient of the penalty at the boundary is larger than the loss functions gradient in terms of absolute value, i.e. $v + \lim_{z \rightarrow z^*} f'(z) = 0$:

$$I_\epsilon(\epsilon_i) = \begin{cases} v \cdot \epsilon, & \epsilon \geq 0 \\ 0, & \epsilon < 0 \end{cases} \quad (8.49)$$

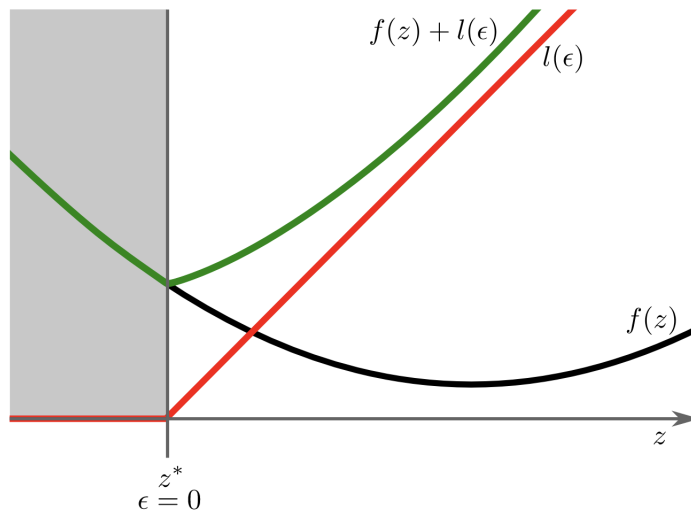


Figure 8.5: Issue with shifted terminal set (lecture slides)

The effect of a linear function with sufficiently large constant factor can be seen in figure 8.5. The value of v for which the linear function fulfills the requirement $v + \lim_{z \rightarrow z^*} f'(z) = 0$, such that the minimum is not shifted, can be found to be the optimal Lagrange multiplier λ^* for the original problem (quantifying the sensitivity with respect to objective changes). As this yields the same result as the problem with hard constraints, it can be seen that penalizing violations in the correct way, the same solution as before without constraint violations will be found, while enlarging the feasible set at the same time. Generally, increasing v results in higher peak violations, but smaller violation durations (more or less the opposite effect of quadratic penalties) and too large penalties might lead to numerical issues. For this reason, linear and quadratic penalties are usually combined for better tuning capabilities in practice:

$$I_\epsilon(\epsilon) = v \cdot \epsilon + s \cdot \epsilon^2 \quad (8.50)$$

Extending the concept to multiple dimensions for r constraints $g_j(z) \leq 0$ results in the equivalent matrix-norm penalty function with $\epsilon = (\epsilon_1, \dots, \epsilon_r)^T$, $S > 0$ and $v > \|\lambda\|_{1/\infty}$:

$$I_\epsilon(\epsilon) = v \cdot \|\epsilon\|_{1/\infty} + \epsilon^T S \epsilon \quad (8.51)$$

The effects on the solution trajectory are very similar to other cost functions. Increasing the cost matrix S for example will lead to a higher quadratic penalty on constraint violations, which will cause them to converge as fast as possible close to the constraint border, but at the same time usually yields longer violation durations.

As shortly mentioned above, it is common that the violation size and duration are coupled in some way and influenced by the choice of the penalty function. In order to decouple these effects, simplify tuning and satisfy the constraints if possible, the objectives can be separated into two

optimization problems. In the first step, the violation over the horizon is minimized:

$$\epsilon^{\min} = \arg \min_{\mathbf{u}, \epsilon} \sum_{i=0}^{N-1} \epsilon_i^T S \epsilon_i + v^T \epsilon_i \quad (8.52)$$

$$\text{s.t. } x_{i+1} = Ax_i + Bu_i \quad (8.53)$$

$$H_x x_i \leq k_x + \epsilon_i \quad (8.54)$$

$$H_u u_i \leq k_u \quad (8.55)$$

$$\epsilon_i \geq 0 \quad (8.56)$$

Subsequently, the controller performance is optimized, assuming the maximum constraint violations to be fixed with ϵ_i^{\min} :

$$\min_{\mathbf{u}} \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T P x_N \quad (8.57)$$

$$\text{s.t. } x_{i+1} = Ax_i + Bu_i \quad (8.58)$$

$$H_x x_i \leq k_x + \epsilon_i^{\min} \quad (8.59)$$

$$H_u u_i \leq k_u \quad (8.60)$$

The only disadvantage of this approach is the computational expensiveness as two optimization problems need to be solved instead of a single one.

Summarizing, soft constraints allow to recover feasibility for the optimization problem, even if the constraints cannot be fully satisfied, but require some tradeoffs between size and duration of the violation. Due to their good performance on practical problems, they are often used in practice. However, note that the soft-constrained MPC problems do not provide any stability guarantees for open-loop unstable systems.

Overview Tracking Problem Steps

1. Estimate the state (e.g. by using a Kalman filter)
2. Design the tracking problem by rewriting the problem in delta-formulation, setting up the steady-state problem and scaling the terminal constraint to guarantee convergence
3. In the presence of disturbances, extend the problem according to the derivations for offset-free tracking. This is implemented by augmenting the model with the disturbance, augmenting the estimator to also take the disturbance into account and adapt the steady-state problem.
4. If required, remove the terminal constraint by choosing a sufficiently long horizon.
5. Introduce soft constraints (especially on the state with slack variables and loss augmentation) to ensure feasibility at all times.

9 Robust MPC

The models that MPC relies on and the noise that might be added to them can lead to inaccuracies that yield states and inputs that violate constraints and result in a suboptimal behavior. Similar to the part on disturbances, persistent noise can as well prevent the system from converging to a single point. Even though some noise models can be integrated into the MPC formulation as already shown, the resulting optimal control problem is usually very hard to solve.

9.1 Uncertainty Models

Up to this point, the system was assumed to evolve in a completely deterministic and predictable fashion, which formally meant that the dynamics are given by:

$$x(k+1) = g(x(k), u(k)) \quad (9.1)$$

In the real world however, there are random noise processes w and model uncertainties (often modeled using some constant but unknown parameter θ that changes the system dynamics:

$$x(k+1) = g(x(k), u(k), w(k); \theta), \quad (x, u) \in \mathcal{X} \times \mathcal{U}, \quad w \in \mathcal{W}, \quad \theta \in \Theta \quad (9.2)$$

The objectives for the control law $u(k) = \kappa(x(k))$ are then very similar to before, but taking into account the two new and unknown quantities:

1. The constraints $\{x(k)\} \subset \mathcal{X}$ and $\{u(k)\} \subset \mathcal{U}$ are satisfied for all possible disturbance realizations
2. The resulting system is stable and converges to a neighborhood of the origin (exactly reaching the origin is not possible due to the random noise that cannot be predicted).
3. Optimize the worst-case performance
4. Maximize the set of feasible initial states $x(0)$ for which the conditions above are fulfilled.

In order to solve these problems, some assumptions on the parameter θ and the noise $w(k)$ have to be made. Example uncertainty models are the following ones:

1. *Measurement or input bias* assuming an unknown but constant parameter θ similar to the disturbance in the previous chapter:

$$g(x(k), u(k), w(k); \theta) = \tilde{g}(x(k), u(k)) + \theta \quad (9.3)$$

2. *Linear parameter varying systems* where the time varying parameters $\theta(k)$ include both the time-varying noise and the constant parameter, using a convex combination of the matrices A_j and B_j with $\mathbf{1}^T \theta(k) = 1$, $\theta(k) \geq 0$:

$$g(x(k), u(k), w(k); \theta) = \left(\sum_{j=1}^{n_\theta} \theta_j(k) A_j \right) x(k) + \left(\sum_{j=1}^{n_\theta} \theta_j(k) B_j \right) u(k) \quad (9.4)$$

3. *Additive stochastic noise* models assume an unknown and time-varying additive noise component $w(k)$ and linear system dynamics:

$$g(x(k), u(k), w(k); \theta) = Ax(k) + Bu(k) + w(k), \quad w \in \mathcal{W} \quad (9.5)$$

This allows to model many non-linearities and even is conservative in many cases. Note that the noise is persistent and does not converge to zero for $k \rightarrow \infty$. The following sections will focus on uncertainty models of this form.

9.2 Impact of Bounded Additive Noise

Considering the last type of the introduced noise models with additive stochastic noise and the design objective for the optimal control law, it is obvious that the main challenge will be to compute the set of trajectories $\phi_i(x_0, U, W)$ the system might follow, as exact predictions are not possible. The solution approach is to design a control law that will satisfy all constraints and result in a stable system for all possible disturbances (i.e. for all trajectories that could occur). Graphically, this is shown in figure 9.1, where the bold curve denotes the nominal trajectory with zero noise and the dashed ones correspond to different noise values $w \in \mathcal{W}$, describing the ‘tube’ of possible trajectories around the nominal one.

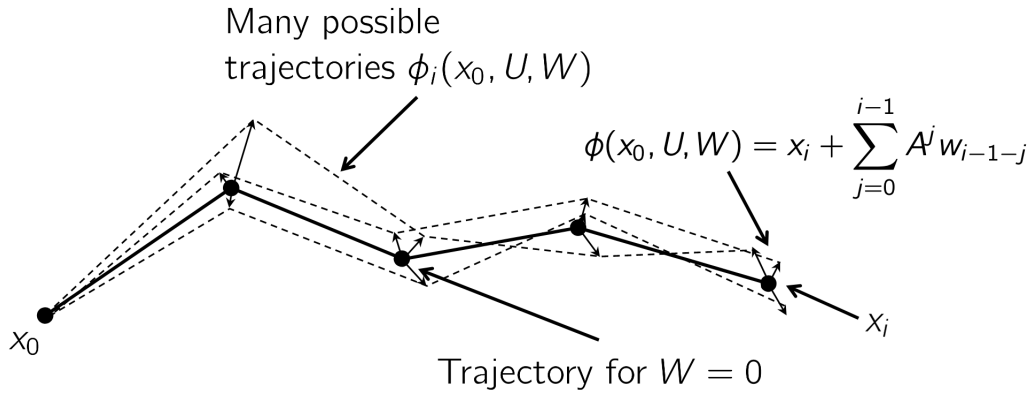


Figure 9.1: Trajectories for different disturbance realizations (lecture slides)

$\phi_i(x_0, U, W)$ is defined as the state where the system would be starting from x_0 and applying $U = \{u_0, \dots, u_{N-1}\}$ with the disturbance realizations $W = \{w_0, \dots, w_{N-1}\}$. To find an expression for this state, the system dynamics equation can be applied multiple times, which results in the following expression for the state at time i , depending on the nominal state x_i that was derived earlier:

$$\phi_i = A^i x_0 + \sum_{j=0}^{i-1} A^j B u_{i-1-j} + \sum_{j=0}^{i-1} A^j w_{i-1-j} = x_i + \sum_{j=0}^{i-1} A^j w_{i-1-j}, \quad x_i = A^i x_0 + \sum_{j=0}^{i-1} A^j B u_{i-1-j} \quad (9.6)$$

It can be seen that in case of additive noise $w \in \mathcal{W}$, the system state at time i will correspond to the nominal system state x_i at this time and an additional offset that is caused by the disturbance (only holds for linear systems of the presented form for $g(x(k), u(k), w(k); \theta)$). For the third objective of the optimization problem (i.e. optimizing the worst-case performance) for robust constrained control, the cost function has to be augmented with a dependency on the disturbance realization, as the different resulting state trajectories will result in different cost values:

$$J(x_0, U, W) = \sum_{i=0}^{N-1} I(\phi_i(x_0, U, W), u_i) + I_f(\phi_N(x_0, U, W)) \quad (9.7)$$

However, as it is intractable to compute the costs for all possible disturbance realizations, a certain optimization objective has to be determined. One option would be to optimize the expected value (given that the noise fulfills some assumptions):

$$J_N(x_0, U) = \mathbb{E}[J(x_0, U, W)] \quad (9.8)$$

Another option would be to optimize the cost for the disturbance with the worst-case cost outcome. This is often done, but computationally relatively expensive:

$$J_N(x_0, U) = \max_{W \in \mathcal{W}^{N-1}} J(x_0, U, W) \quad (9.9)$$

Finally, the disturbance could also just be ignored and the nominal cost is used, which can also be interpreted as kind of a mean (given that the noise does not have a significant bias) and is also often done in practice:

$$J_N(x_0, U) = J(x_0, U, 0) \quad (9.10)$$

The last option to simply optimize the cost for the nominal case is chosen for the scope of this lecture.

Continuing with the constraints on state and input (first objective of the optimization problem), the constraint equations need to be adapted to take into account the disturbance terms. As before, the constraint satisfaction can be split into two parts.

In a first step, the input and state constraints $\phi_i \in \mathcal{X}$ and $u_i \in \mathcal{U}$ are explicitly enforced for all $W \in \mathcal{W}^N$ over the optimization horizon $i = 0, \dots, N - 1$ assuming the dynamics $\phi_{i+1} = A\phi_i + Bu_i + w_i$. For all steps from time $i = N$ forward (i.e. $i = N, \dots$), a linear control law $u_i = K\phi_i$ is used and the constraints are implicitly enforced by requiring the terminal set $\mathcal{X}_f \subseteq \mathcal{X}$ is *robust invariant* (invariant independent of the disturbance realization) and the input constraints are always satisfied $K\mathcal{X}_f \subseteq \mathcal{U}$ under the closed-loop state dynamics $\phi_{i+1} = (A + BK)\phi_i + w_i$. As the invariant set will not be left, the constraints will be satisfied for all future times.

Robustly Enforcing Constraints of a Linear System

Starting with the second part, the terminal set \mathcal{X}_f has to be modified in order to be invariant independent of the noise disturbance acting on the system. Extending the definition of positive invariance to autonomous systems with noise (dynamics $x(k+1) = g(x(k), w(k))$) or closed-loop systems with some given controller κ and noise (dynamics $x(k+1) = g(x(k), \kappa(x(k)), w(k))$), a set $\mathcal{O}^{\mathcal{W}}$ is said to be *robust positive invariant* for the autonomous system, if every state in this set is again mapped to the set (independent of the disturbance):

$$x \in \mathcal{O}^{\mathcal{W}} \Rightarrow g(x, w) \in \mathcal{O}^{\mathcal{W}}, \quad \forall w \in \mathcal{W} \quad (9.11)$$

This is equivalent to the condition that the nominal system dynamics always need to move the state at least as far towards the center of the set so that it still is part of the set for the worst-case disturbance. Similarly, the pre-set of some Ω for the autonomous system is the set of all points that evolve into Ω within the next timestep for all values $w \in \mathcal{W}$ (not only for a single w):

$$\text{pre}^{\mathcal{W}}(\Omega) := \{x \mid g(x, w) \in \Omega, \quad \forall w \in \mathcal{W}\} \quad (9.12)$$

Given a linear autonomous system with additive noise $g(x(k), w(k)) = Ax(k) + w(k)$, the robust pre-set of some polytopic set $\Omega = \{x \mid Fx \leq f\}$ can be formulated as the following polytopic set:

$$\text{pre}^{\mathcal{W}}(\Omega) = \{x \mid Ax + w \in \Omega, \quad \forall w \in \mathcal{W}\} = \{x \mid FAx + Fw \leq f, \quad \forall w \in \mathcal{W}\} \quad (9.13)$$

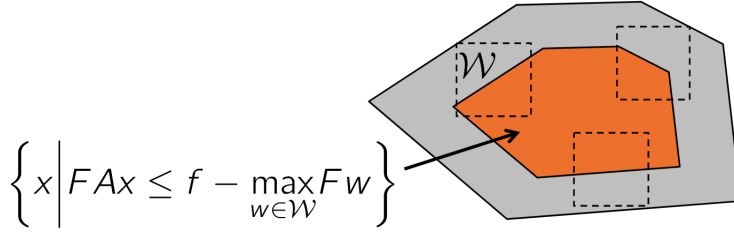


Figure 9.2: Robust pre-set computation (lecture slides)

The polytopic pre-set for the nominal system can be easily denoted through its constraints $F_j Ax \leq f_j$. Assuming some non-zero noise w now, the constraints change to $F_j Ax \leq f_j - F_j w$ and assuming the worst case noise (i.e. the one restricting the pre-set the most) yields the constraint inequalities:

$$F_j Ax \leq f_j - \max_{w \in \mathcal{W}} F_j w \quad (9.14)$$

Using the matrix notation, the robust pre-set can be denoted by tightening all constraints by the maximum disturbance (or the support function $h_{\mathcal{W}}(F) = \max_{w \in \mathcal{W}} Fw$):

$$\text{pre}^{\mathcal{W}}(\Omega) = \{x \mid FAx \leq f - \max_{w \in \mathcal{W}} Fw\} = \{x \mid FAx \leq f - h_{\mathcal{W}}(F)\} \quad (9.15)$$

As before, $\mathcal{O}^{\mathcal{W}}$ is a robust positive invariant set if and only if it is a subset of its pre-set:

$$\mathcal{O}^{\mathcal{W}} \subseteq \text{pre}^{\mathcal{W}}(\mathcal{O}^{\mathcal{W}}) \quad (9.16)$$

Even the algorithm for the computation of the positively invariant set $\mathcal{O}_{\infty}^{\mathcal{W}}$ with the inputs g , \mathcal{X} and \mathcal{W} remains largely unchanged except from the modification of the pre-set computation:

```

 $\Omega_0 \leftarrow \mathcal{X}$ 
loop:
   $\Omega_{i+1} \leftarrow \text{pre}^{\mathcal{W}}(\Omega_i) \cap \Omega_i$ 
  if  $\Omega_{i+1} = \Omega_i$  then:
    return  $\mathcal{O}_{\infty}^{\mathcal{W}} = \Omega_i$ 
  end if
end loop

```

Naturally, the resulting maximum robust invariant set is smaller or equal the maximum nominal invariant set, as the constraints are tightened by the worst-case noise. All the different trajectories that have to be considered under the linear control law $u_i = Kx_i$ for $i = N, \dots$ that start in this set then also stay in this set, independent of the disturbance realization.

Robustly Ensuring Constraints of the State Sequence

After considering the method how the invariant set notation can be extended to the presence of noise (ensuring that the state and input constraints are not violated for $i = N, \dots$ under some constant control law $u = Kx$), the constraints have to be explicitly satisfied for $\phi_1, \dots, \phi_{N-1}$. This is required to ensure that all possible uncertain states $\phi_i(x_i, U, W)$ lie within the constraint set \mathcal{X} . The main idea to implement this is to tighten the constraints such that the nominal system fulfilling the new constraints is a sufficient condition for the uncertain system to fulfill them too. MPC can then be applied directly to the nominal system again. Formally, this condition can be written as follows (where x_i is the nominal state and ϕ_i the uncertain one):

$$\phi_i(x_0, U, W) = \left\{ x_i + \sum_{j=0}^{i-1} A^j w_{i-1-j} \mid W \in \mathcal{W}^i \right\} \subseteq \mathcal{X} \quad (9.17)$$

Assuming a polytopic constraint set $\mathcal{X} = \{x \mid Fx \leq f\}$, the condition on the uncertain states to all lie within the constraint set can be reformulated:

$$Fx_i + F \sum_{j=0}^{i-1} A^j w_{i-1-j} \leq f \quad \forall W \in \mathcal{W}^i \quad (9.18)$$

Rearranging the terms and assuming the worst case disturbance at every time i , the state constraint can be written as follows (where $h_{\mathcal{W}^i}$ again is the support function):

$$Fx_i \leq f - \max_{W \in \mathcal{W}^i} F \sum_{j=0}^{i-1} A^j w_{i-1-j} = f - h_{\mathcal{W}^i} \left(F \sum_{j=0}^{i-1} A^j \right) \quad (9.19)$$

The support function that is required to compute these tightened constraints for the nominal system can be computed offline, assuming that an error bound is known ahead of runtime. Beyond the requirement that all uncertain states need to satisfy the constraints, the final state $\phi_N(x_0, U, W)$ additionally has to lie within the robust invariant terminal set \mathcal{X}_f in order for the system to stay there, resulting in a tightened terminal state (implemented in the same way as before):

$$\phi_N(x_0, U, W) \subseteq \mathcal{X}_f \quad (9.20)$$

While the initial state x_0 can lie anywhere in the set \mathcal{X} , the first nominal state x_1 is already restricted to lie in a subset of the state (as shown in figure 9.3, tightened by the maximum amount of disturbance that is expected. This ensures that any uncertain state $\phi_1(x_0, U, W)$ lies within the constraint set \mathcal{X} , independent of the disturbance realization.

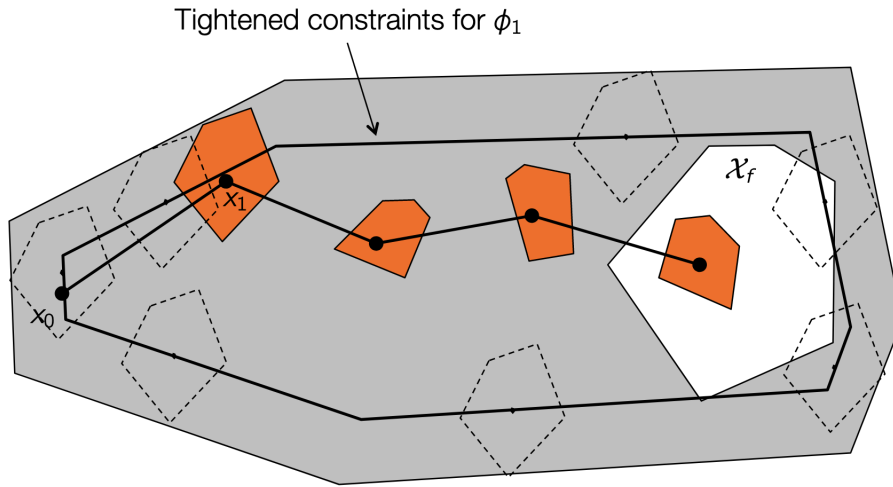


Figure 9.3: Trajectories for different disturbance realizations (lecture slides)

In order to describe this tightening of the constraint set formally, additional notation is required. A common measure to formulate the sum of two sets is the *Minkowski sum* with the following definition:

$$A \oplus B := \{x + y \mid x \in A \subseteq \mathbb{R}^n, y \in B \subseteq \mathbb{R}^n\} \quad (9.21)$$

Inversely, shrinking a set, as it is required for tightening a set, is provided by the *Pontryagin difference*:

$$A \ominus B = \{x \mid x + \epsilon \in A, \epsilon \in B\} \quad (9.22)$$

Even though these measures are similar, the difference often cuts off regions of the set that just cannot be followed by the uncertainty polytope, wherefore adding it again often only results in a subset of the original one:

$$(A \ominus B) \oplus B \subseteq A \quad (9.23)$$

The condition of the uncertain state $\phi_i(x_0, U, W)$ to lie inside the constraint set tightened by the worst possible noise realization, can also be denoted as follows:

$$\phi_i(x_0, U, W) \in x_i \oplus (\mathcal{W} \oplus A\mathcal{W} \oplus \dots \oplus A^{i-1}\mathcal{W}) \subseteq \mathcal{X} \quad (9.24)$$

Tightening the constraints and expressing it as a condition for the nominal state x_i results in:

$$x_i \in \mathcal{X} \ominus (\mathcal{W} \oplus A\mathcal{W} \oplus \dots \oplus A^{i-1}\mathcal{W}) = \mathcal{X} \ominus \left(\bigoplus_{j=0}^{i-1} A^j \mathcal{W} \right) \quad (9.25)$$

9.3 Robust Open-Loop MPC

Putting everything together, the robust open-loop MPC problem can be formulated as follows with some robust positive invariant set $\mathcal{X}_f \subseteq \mathcal{X}$:

$$\min_U \sum_{i=0}^{N-1} I(x_i, u_i) + I_f(x_N) \quad (9.26)$$

$$\text{s.t. } x_{i+1} = Ax_i + Bu_i \quad (9.27)$$

$$x_0 = x(k) \quad (9.28)$$

$$x_i \in \mathcal{X} \ominus (\mathcal{W} \oplus A\mathcal{W} \oplus \dots \oplus A^{i-1}\mathcal{W}) \quad (9.29)$$

$$u_i \in \mathcal{U} \quad (9.30)$$

$$x_N \in \mathcal{X}_f \ominus (\mathcal{W} \oplus A\mathcal{W} \oplus \dots \oplus A^{N-1}\mathcal{W}) \quad (9.31)$$

It is assumed that the system is either stable or has been pre-stabilized by choosing a suitable control law $Kx \in \mathcal{U}$ such that the closed-loop system $x(k+1) = (A+BK)x(k) + w(k)$ is stable for all disturbances $w(k) \in \mathcal{W}$. This control law is also assumed to exist for all $x \in \mathcal{X}_f$ such that it can be applied for all times $k = N, \dots$ that are beyond the current optimization horizon (positive invariant terminal set \mathcal{X}_f while the input constraints are satisfied). Note that the problem then corresponds to the nominal MPC problem with tightened constraints on the state. If the nominal system satisfies the tightened constraints, one can be sure that the uncertain system will satisfy the original state constraints \mathcal{X} as well. However, this condition is only necessary but not sufficient, as the potential violation of constraints strongly depends on the disturbance realizations.

An important property of the optimizer $U^*(x(k))$ of the open-loop MPC problem is that for any $x(k) \in \mathcal{X}_N$, the next possible uncertain state $Ax(k) + Bu_0^*(x(k)) + w(k) \in \mathcal{X}_N$ lies within the same set for all possible disturbances $w \in \mathcal{W}$ (robust invariance). This follows from the fact that the trajectory that was computed is assuming the worst-case disturbance realization, wherefore the properties will also hold for the actual observation with some arbitrary disturbance realization $w \in \mathcal{W}$. This robust invariance is the key property of robust MPC and a sketch of the proof (for stable systems) can be found in the lecture notes. In a next step, the convergence requirements and optimization to get a maximum set of feasible initial states $x(0)$ will be tackled.

9.4 Closed-Loop Predictions

The MPC problem in the presence of disturbances can also be modeled as a game with two players where one of them is the controller and the other one is nature, potentially reacting with the worst possible disturbance to the input u chosen by the controller before (or equivalently inputs and disturbances over the entire optimization horizon N might be chosen).

Up to this point, it was assumed that the controller just continues to apply its input that it has computed at time k until $k + N$ without taking previous disturbance realizations into account (worst-case assumption). However, by applying online control methods, the controller also has the possibility to choose its input at the second timestep based on the real state $x(1) = Ax_0 + Bu_0 + w(0)$, which already includes the first disturbance realization (and so on for the following stages). Normally, this allows to obtain a better controller performance than just always assuming the worst case and the tightening the constraints by the maximum amount.

Instead, the formal approach for closed-loop predictions would then be to optimize over a sequence of functions $\{u_0, \mu_1(\cdot), \dots, \mu_{N-1}(\cdot)\}$ that contains the control policies $\mu_i(x_i) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ mapping the real states to inputs during execution (online algorithm; except from the first control input u_0 that can be computed as it is a function of the known initial state). The main problem of this approach is that optimization over arbitrary functions it not feasible, wherefore one of the following structures is usually assumed on μ_i :

- *Pre-stabilization:* $\mu_i(x_i) = Kx_i + v_i$ where K is fixed such that the closed loop system $A + BK$ is stable. This approach is simple and often conservative (better performance can be achieved).
- *Linear feedback:* $\mu_i(x_i) = K_i x_i + v_i$ where the optimization is carried out over K_i as well as v_i . This formulation results in a non-convex problem, which is hard to solve.
- *Disturbance feedback:* $\mu_i(x_i) = \sum_{j=0}^{i-1} M_{ij} w_j + v_i$ where the optimization is carried out over M_{ij} and v_i . While the problem formulation is equivalent to linear feedback, the problem is now convex and can therefore be solved (even with potentially high computational cost). Besides this, the approach is usually less conservative than the worst-case assumption from pre-stabilization.
- *Tube-MPC:* $\mu_i(x_i) = v_i + K(x - \bar{x}_i)$ where K is fixed such that the closed-loop system $A + BK$ is stable and the optimization can be carried out over \bar{x}_i and v_i . This approach is effective and can be reduced to the nominal MPC problem formulation, what will be shown in the next section.

9.5 Tube-MPC

Tube-MPC again considers a system with linear dynamics and additive disturbance $w \in \mathcal{W}$, while the states and inputs have to satisfy the constraints $x \in \mathcal{X}$ and $u \in \mathcal{U}$. The main idea is to split the available control authority into two parts, one part to steer the nominal system to the origin and thus lead to stability and another part to compensate for the deviations from the nominal system. To avoid confusions, the nominal state and input will be denoted with $z(k)$ and $v(k)$ respectively:

$$z(k+1) = Az(k) + Bv(k) \quad (9.32)$$

The part to compensate for deviations from the nominal system can be achieved by applying a tracking controller that keeps the disturbed trajectory as close as possible to the nominal system (using some linear controller gain K to multiply the difference between nominal and uncertain state):

$$u_i = K(x_i - z_i) + v_i \quad (9.33)$$

The linear feedback controller K can then be fixed offline and let the MPC problem optimize over the nominal input trajectory $\{v_0, \dots, v_{N-1}\}$, resulting in a convex problem. Defining the error as the distance between nominal and disturbed state, yields the following error dynamics:

$$e_{i+1} = x_{i+1} - z_{i+1} = Ax_i + BK(x_i - z_i) + Bv_i + w_i - Az_i - Bv_i \quad (9.34)$$

$$= (A + BK)(x_i - z_i) + w_i = (A + BK)e_i + w_i, \quad w_i \in \mathcal{W} \quad (9.35)$$

In order to bound the maximum error, the dynamics $A + BK$ should be stable, which is fulfilled by the condition on the choice of K , and the noise $w_i \in \mathcal{W}$ should be bounded. In this case there will be some set \mathcal{E} , which the error will never leave. In contrast to previous problems, the objective is now to find the minimal robust invariant set (i.e. the smallest set in which the error will stay for all times). The computation of this will be considered later on.

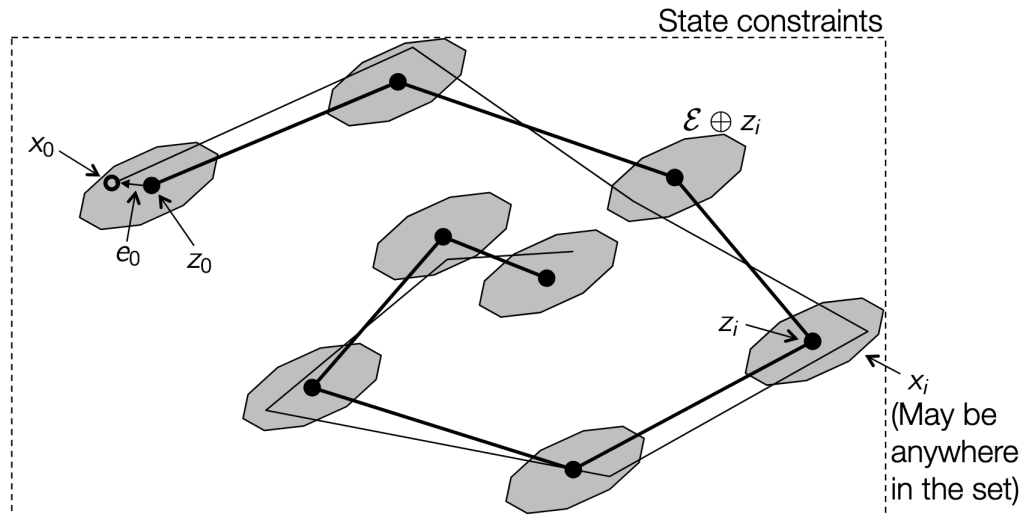


Figure 9.4: Illustration of Tube-MPC (lecture slides)

The idea of tube-MPC is now to ignore the noise and only optimize for the nominal trajectory with some changed constraints. Using the fact that there exists a robust invariant set \mathcal{E} around each nominal state z_i in which the uncertain system state has to lie (assuming bounded disturbances $w_i \in \mathcal{W}$), it is sufficient for the nominal state to be sufficiently far away from the boundary so that all points $x_i \in z_i \oplus \mathcal{E} \subseteq \mathcal{X}$ lie inside the state constraints as illustrated in figure 9.4. This assumption that the real state trajectory will stay close to the nominal one holds because of the plan to apply the valid but suboptimal controller $u_i = K(x_i - z_i) + v_i$ in the future (even if this control law is changed afterwards the plan allows to assume something else than the worst-case disturbance). As the initial condition is assumed to be known, choosing it allows to position it optimally for the MPC cost. To achieve these objectives, the minimum robust invariant set \mathcal{E} of the error needs to be computed, the state and input constraints need to be modified to be applied to the nominal trajectory $\{z_i\}$ and the problem needs to be formulated as a convex optimization problem. Finally, it needs to be shown that the constraints are robustly satisfied and that the closed-loop system is robustly stable.

Computing the Set \mathcal{E}

For the formulation of the minimum robust invariant set for the state to remain inside despite disturbances, the corresponding definition for a general autonomous system $g(x(k), w(k))$ (or also some closed loop system with known controller) has to fulfill the following property for some set $\mathcal{O}^{\mathcal{W}}$ to be robust positive invariant:

$$x \in \mathcal{O}^{\mathcal{W}} \Rightarrow g(x, w) \in \mathcal{O}^{\mathcal{W}}, \quad \forall w \in \mathcal{W} \quad (9.36)$$

Reconsidering the error dynamics $e_{i+1} = A_k e_i + w_i$ and assuming $e_0 = 0$ (A_k can either just be the system matrix A for open-loop systems or the pre-stabilized system $A_k = A + BK$), results in the following expression for the error evolution:

$$e_i = \sum_{j=0}^{i-1} A_k^j w_{i-1-k} \quad (9.37)$$

The set F_i containing all possible states e_i can be defined using the Minkowski sum:

$$F_i = \mathcal{W} \oplus A_k \mathcal{W} \oplus \dots \oplus A_k^{i-1} \mathcal{W} = \bigoplus_{j=0}^{i-1} A_k^j \mathcal{W}, \quad F_0 := \{0\} \quad (9.38)$$

As the horizon goes to infinity, the minimum robust invariant set results as:

$$F_\infty = \bigoplus_{j=0}^{\infty} A_k^j \mathcal{W}, \quad F(0) := \{0\} \quad (9.39)$$

Optimally, this point can be found through the fix point condition $F_n = F_{n+1} = F_\infty$. However, this is not always possible and large values for n are required for good approximations. The following algorithm allows to find a minimal robust invariant set (in contrast to the algorithms finding maximal robust invariant sets before), requiring A as an input and returning F_∞ :

```

 $\Omega_0 \leftarrow \{0\}$ 
loop:
   $\Omega_{i+1} \leftarrow \Omega_i \oplus A^i \mathcal{W}$ 
  if  $\Omega_{i+1} = \Omega_i$  then:
    return  $F_\infty = \Omega_i$ 
  end if
end loop

```

Constraint Modification on Nominal Trajectory

Given the nominal trajectory values z_i , the real system is given by $x_i = w_i + e_i$ with an unknown $e_i \in \mathcal{E}$, leading to the upper bound:

$$x_i \in z_i \oplus \mathcal{E} = \{z_i + e \mid e \in \mathcal{E}\} \quad (9.40)$$

In order to achieve $x_i, u_i \in \mathcal{X} \times \mathcal{U}$ (i.e. the state and input lie within the respective constraints at all times) for all disturbance realizations $w_0, \dots, w_{i-1} \in \mathcal{W}$, the constraints on the nominal system $z(k+1) = Az(k) + Bv(k)$ need to be tighter than \mathcal{X} itself in the case of a non-zero disturbance. Otherwise the disturbed system $x(k+1) = Ax(k) + Bu(k) + w(k)$ might violate some constraints. The necessary and sufficient condition for the nominal states is the following:

$$z_i \oplus \mathcal{E} \subseteq \mathcal{X} \iff z_i \in \mathcal{X} \ominus \mathcal{E} \quad (9.41)$$

The condition on the right directly implies that $x_i \in z_i \oplus \mathcal{E} \subseteq \mathcal{X} \ominus \mathcal{E} \oplus \mathcal{E} \subseteq \mathcal{X}$ and therefore constraint satisfaction for the disturbed state trajectory. Note that the tightened constraint set $\mathcal{X} \ominus \mathcal{E}$ can be computed offline. A similar condition can be formulated for the input:

$$u_i \in K\mathcal{E} \oplus v_i \subset \mathcal{U} \iff v_i \in \mathcal{U} \ominus K\mathcal{E} \quad (9.42)$$

While v_i obviously is the control authority needed for the nominal system, $K\mathcal{E}$ is the part that is required to compensate for the encountered disturbances. The real input for the disturbed system will then be computed using the expression with feedback gain K , combining nominal and uncertain state with the nominal input:

$$u_i = K(x_i - z_i) + v_i = Ke_i + v_i \quad (9.43)$$

Formulate Convex Optimization Problem

The Tube-MPC problem can be written in the following form where only the constraints on the nominal states and inputs have been tightened and the initial condition modified compared to the nominal MPC problem:

$$\min_{Z,V} \sum_{i=0}^{N-1} I(z_i, v_i) + I_f(z_N) \quad (9.44)$$

$$\text{subj. to } z_{i+1} = Az_i + Bv_i \quad (9.45)$$

$$z_i \in \mathcal{X} \ominus \mathcal{E} \quad (9.46)$$

$$v_i \in \mathcal{U} \ominus K\mathcal{E} \quad (9.47)$$

$$z_N \in \mathcal{X}_f \subseteq \mathcal{X} \ominus \mathcal{E} \quad (9.48)$$

$$x(k) \in z_0 \oplus \mathcal{E} \quad (9.49)$$

The control law additionally not only contains the first element of the optimal nominal input sequence, but also the required feedback to keep the error in the invariant set \mathcal{E} :

$$\mu_{\text{tube}}(x) := K(x - z_0^*(x)) + v_0^*(x) \quad (9.50)$$

Note that this formulation allows to use the first tube center z_0 also as an optimization variable with the constraint that it has to be within \mathcal{E} of $x_0 = z_0$. The cost function $J(Z, V)$ is formulated with respect to the tube centers, i.e. the nominal system. The feasible set can now be written as the set of nominal states and inputs for some given initial condition x_0 that fulfill all given constraints:

$$\mathcal{Z}(x_0) := \left\{ Z, V \left| \begin{array}{l} z_{i+1} = Az_i + Bv_i, \quad i = 0, \dots, N-1 \\ z_i \in \mathcal{X} \ominus \mathcal{E}, \quad i = 0, \dots, N-1 \\ v_i \in \mathcal{U} \ominus K\mathcal{E}, \quad i = 0, \dots, N-1 \\ z_N \in \mathcal{X}_f \\ x(k) \in z_0 \oplus \mathcal{E} \end{array} \right. \right\} \quad (9.51)$$

Correspondingly, the optimization will return the following optimal nominal state and input sequences, of which the first nominal state and input are used in the control law given by equation 9.50:

$$(V^*(x_0), Z^*(x_0)) = \arg \min_{V,Z} \{J(Z, V) \mid (Z, V) \in \mathcal{Z}(x_0)\} \quad (9.52)$$

The following assumption, which are very similar to the ones for nominal MPC, are still required for tube-MPC to achieve the expected performance:

- The stage cost is positive definite and only zero at the origin
- The terminal set is invariant for the nominal system under the local control law $\kappa_f(z)$ (no robust invariance is required as this is achieved through the constraint tightening):

$$Az + B\kappa_f(z) \in \mathcal{X}_f, \quad \forall z \in \mathcal{X}_f \quad (9.53)$$

Meanwhile, all state and input constraints need to be satisfied in \mathcal{X}_f :

$$\mathcal{X}_f \subseteq \mathcal{X} \ominus \mathcal{E}, \quad \kappa_f(z) \in \mathcal{U} \ominus K\mathcal{E}, \quad \forall z \in \mathcal{X}_f \quad (9.54)$$

- Finally, the terminal cost is a continuous Lyapunov function in the terminal set \mathcal{X}_f , ensuring that the nominal system converges to the origin, while the disturbed system stays in the error bounds specified by \mathcal{E} :

$$I_f(Az + B\kappa_f(z)) - I_f(z) \leq -I(z, \kappa_f(z)), \quad z \in \mathcal{X}_f \quad (9.55)$$

Using these assumptions, it can be shown that the set $\mathcal{Z} := \{x \mid \mathcal{Z}(x) \neq \emptyset\}$ is in fact a robust invariant set of the system $x(k+1) = Ax(k) + B\mu_{\text{tube}}(x(k)) + w(k)$ subject to the constraints $(x, u) \in \mathcal{X} \times \mathcal{U}$. The proof just uses that the disturbed state $x(k+1)$ after the application of the tube-MPC control law has to lie within $z_1^* \oplus \mathcal{E}$ by construction, independent from the disturbance realization $\bar{w} \in \mathcal{W}$. Therefore, using the same trick as in standard MPC and just applying the optimal state and input sequence from the previous state $x(k)$ given by $(\{v_0^*, \dots, v_{N-1}^*\}, \{z_0^*, \dots, z_N^*\})$ that can be shifted by one and appended with the local control law in the terminal set, results in:

$$(\{v_1^*, \dots, v_{N-1}^*, \kappa_f(z_N^*)\}, \{z_1^*, \dots, z_N^*, Az_N^* + B\kappa_f(z_N^*)\}) \quad (9.56)$$

By construction of the problem, all states in the previous optimal sequence are optimal and $Az_N^* + B\kappa_f(z_N^*) \in \mathcal{X}_f$ by the definition of the feasible set \mathcal{Z} . Accordingly, this solution is feasible for all possible $x(k+1)$ (i.e. independent from the noise realization) and the set \mathcal{Z} is robust invariant.

Furthermore, it can be shown that the state $x(k)$ of the system $x(k+1) = Ax(k) + B\mu_{\text{tube}}(x(k)) + w(k)$ converges in the limit to the set \mathcal{E} , corresponding to robust stability of tube-MPC. Note that $\lim_{k \rightarrow \infty} J(z_0^*(x(k))) = 0$, which is a Lyapunov function for the nominal system, implies that the nominal state converges to zero $\lim_{k \rightarrow \infty} z_0^*(x(k)) = 0$. Also note that the disturbed state $x(k)$ does not converge to zero, but can only stay within the robust invariant set $z^*(x) \oplus \mathcal{E}$ as disturbances cannot be anticipated.

9.6 Implementation of Tube-MPC

When implementing Tube-MPC, some quantities can already be computed offline with the corresponding knowledge of model and disturbances:

- Choose a stabilizing controller K to stabilize the error dynamics with $\|A + BK\| < 1$
- Compute the minimal robust invariant set $\mathcal{E} = F_\infty$ for the disturbed, but stabilized system: $x(k+1) = (A + BK)x(k) + w(k)$, $w \in \mathcal{W}$
- Compute the tightened constraints $\tilde{\mathcal{X}} := \mathcal{X} \ominus \mathcal{E}$, $\tilde{\mathcal{U}} = \mathcal{U} \ominus K\mathcal{E}$
- Choose the terminal weight function I_f and the constraint set \mathcal{X}_f satisfying the corresponding assumptions formulated earlier. This terminal constraint set as well as the control law $\kappa_f(x) = Kx$ are often chosen as the LQR control law (and \mathcal{X}_f then needs to be invariant under this controller).

The online part of tube-MPC then consists of the following points:

1. Measure or estimate the state x
2. Solve the optimization problem $(Z^*(x), V^*(x)) = \arg \min_{V, Z} \{J(Z, V) \mid (Z, V) \in \mathcal{Z}(x)\}$
3. Set the input according to the tube-MPC control law $u = K(x - z_0^*(x)) + v_0^*(x)$

The main benefits of tube-MPC are that it is less conservative than open-loop robust MPC, works for general unstable systems and can be implemented by solving relatively easy optimization problems. However, note that the solution is still sub-optimal (due to the worst-case noise assumption for robustness; optimality is very difficult), that the feasible set is reduced compared to nominal MPC and that \mathcal{W} needs to be known (what is usually problematic in practice). More generally, standard robust open-loop MPC for uncertain systems usually becomes relatively complex, the largest noise in \mathcal{W} needs to be known, it is conservative and the resulting feasible set might be very small. The robust MPC problem however directly results in an invariant feasible set and the trade-off between robustness and performance is very easily implementable.

9.7 Robust Constraint Tightening MPC

The main idea between robust constraint tightening MPC is to combine the two concepts of robust open-loop MPC, where the optimization was carried out over the disturbed states and inputs with a robust invariant set as terminal set constraint, and robust tube-based MPC where the optimization is carried out over the nominal system, but with constraints tightened by the robust invariant set \mathcal{E} , taking into account the disturbances until $k \rightarrow \infty$. In reality however, this uncertainty bound with the Minkowsky sum of \mathcal{E} might be far too high (especially for states close to the initial condition) as the system needs time to deviate from the nominal trajectory using disturbances $w \in \mathcal{W}$. The error at state i will always be within the following sum (derived earlier):

$$e_i = \sum_{j=0}^{i-1} A^j w_{i-1-j} \in (\mathcal{W} \oplus A_k \mathcal{W} \oplus \dots \oplus A_k^{i-1} \mathcal{W}) \quad (9.57)$$

Robust positively invariant (RPI) tubes:

Disturbance reachable sets (DRS):

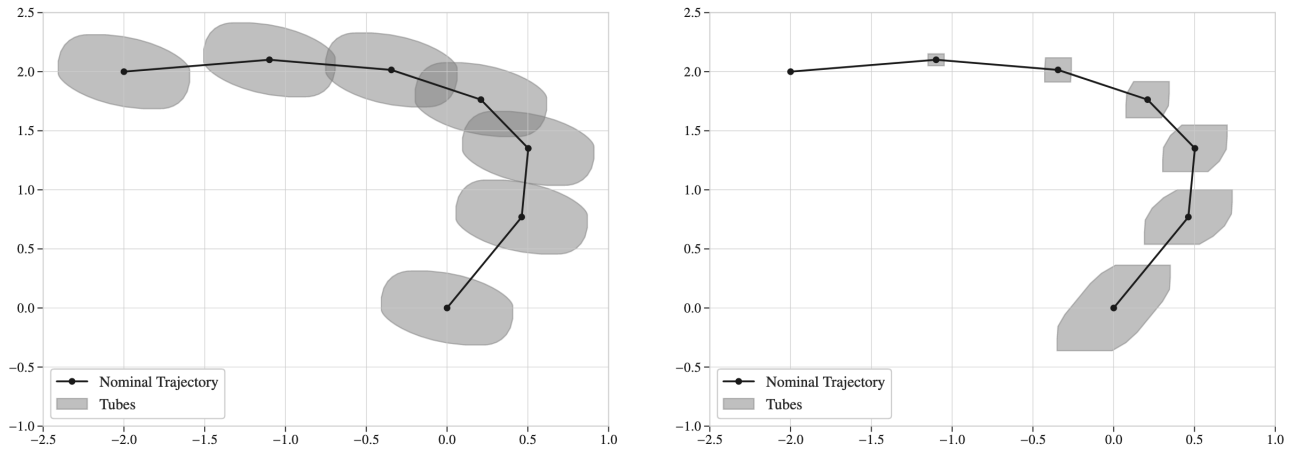


Figure 9.5: Difference between invariant tube sets and actually reachable sets (lecture slides)

The different estimated disturbance deviations can be seen in figure 9.5 for robust tube-MPC and the case which disturbed values can actually be reached at this stage, using maximum disturbances. Fusing those concepts, the optimization problem can still be written in terms of the nominal state and input, but with the modification that the constraint tightening now also depends on actually reachable states:

$$\min_{Z, V} \sum_{i=0}^{N-1} I(z_i, v_i) + I_f(z_N) \quad (9.58)$$

$$\text{subj. to } z_{i+1} = Az_i + Bv_i \quad (9.59)$$

$$z_i \in \mathcal{X} \ominus (\mathcal{W} \oplus A_k \mathcal{W} \oplus \dots \oplus A_k^{i-1} \mathcal{W}) \quad (9.60)$$

$$u_i \in \mathcal{U} \ominus K(\mathcal{W} \oplus A_k \mathcal{W} \oplus \dots \oplus A_k^{i-1} \mathcal{W}) \quad (9.61)$$

$$z_N \in \mathcal{X}_f \ominus (\mathcal{W} \oplus A_k \mathcal{W} \oplus \dots \oplus A_k^{N-1} \mathcal{W}) \quad (9.62)$$

$$z_0 = x(k) \quad (9.63)$$

The control law that will be applied after optimizing over the nominal inputs and states is given as follows:

$$u(k) = v_0^* + K(x(k) - z_0) = v_0^* \quad (9.64)$$

In this way the constraint satisfaction at each timestep can be ensured, with the additional requirement that the terminal set \mathcal{X}_f now again needs to be robust invariant under the tube controller K (instead of just being invariant for tube-MPC).

9.8 Robustness of Nominal MPC

Another option is to just ignore the presence of noise and/or uncertainty and control the noisy system with the solution found through the nominal MPC problem that was given by:

$$J^*(x_0) = \min_U \sum_{i=0}^{N-1} I(x_i, u_i) + I_f(x_N) \quad (9.65)$$

$$\text{subj. to } x_{i+1} = Ax_i + Bu_i \quad (9.66)$$

$$(x_i, u_i) \in \mathcal{X} \times \mathcal{U} \quad (9.67)$$

$$x_N \in \mathcal{X}_f \quad (9.68)$$

After solving the optimization problem, the closed-loop system results as:

$$x(k+1) = Ax(k) + Bu_0^*(x(k)) + w(k) \quad (9.69)$$

Observing simulation results, it can be found that the controller still works sometimes. Formally, assuming that the cost function is Lipschitz continuous, it can be derived that the Lyapunov decrease still holds in general, but is slowed down and limited by the maximum disturbance $\{\|w\| \mid w \in \mathcal{W}\}$ (larger decrease for larger state values $\|x\|$):

$$J^*(Ax + Bu^*(x) + w) - J^*(x) \leq I(x, u^*(x)) + \gamma\|w\| \quad (9.70)$$

In other words, the system will move towards the origin until there is a balance between the values of the state and the disturbance. Simply ignoring the noise and applying nominal MPC has the following important benefits:

- Simple implementation
- No knowledge of the noise set \mathcal{W} is required
- Even though the noise is ignored, the controller is usually still very effective in practice
- The feasible set is large, but the solution that was found might not work
- The region of attraction might be larger for nominal MPC in this case than when using alternative methods.

The main disadvantages of just blindly applying nominal MPC in the presence of noise include:

- Description of the region of attraction (i.e. the set of states for which the controller works) is very difficult
- Tuning is difficult as there is no obvious way how to choose the trade-off between robustness and performance
- The controller only works for linear systems and for nonlinear systems where the continuity assumption holds

10 Implementation & Remarks

In order to solve the basic MPC problem as it was already presented in the first chapter with figure 3.1, the optimal input sequence $U_k^* = \{u_k^*, u_{k+1}^*, \dots, u_{k+N-1}^*\}$ has to be computed at each sample time. In order to obtain an efficient MPC implementation, both iterative optimization methods and an explicit solution exist for the following standard formulation:

$$J^*(x(k)) = x_N^T P x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \quad (10.1)$$

$$\text{subj. to } x_{i+1} = A x_i + B u_i, \quad i = 0, \dots, N-1 \quad (10.2)$$

$$x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, N-1 \quad (10.3)$$

$$x_N \in \mathcal{X}_f \quad (10.4)$$

$$x_0 = x(k) \quad (10.5)$$

10.1 Explicit Solution

Considering the mathematical problem formulation, it is easy to observe that the only thing that changes during the different receding horizon steps is the initial state $x_0 = x(k)$. It should therefore be possible to formulate and pre-compute the ideal control sequence for the next N steps as a function of the initial state x_0 . This control law will turn out to be piecewise affine for linear systems and constraints and can just be evaluated at the corresponding initial condition x_0 at each timestep. The resulting online computation is dramatically reduced and can be carried out in real-time.

Reconsidering the case of a continuous finite-time optimal control problem, the resulting control law $\kappa(x(k)) = u_0^*$ turned out to be piecewise affine $\kappa(x) = F^j x + g^j$ on a number of polyhedra CR^j that span the entire feasible state space \mathcal{X}_0 with $j = 1, \dots, N^r$ (solution of the multi-parametric quadratic program). The corresponding value function $J^*(x(k))$ is convex and piecewise quadratic on these polyhedra. These polyhedra are so-called *critical regions*, which contain all feasible states for which the same constraints are active. Formally, first a so-called *active set* $A(x)$ has to be computed that is the set of all constraint indices I that are active at this given state x (and its complement $AN(x)$; G_j and w_j denote the j th row of the matrices or vectors respectively):

$$A(x) := \{j \in I : G_j z^*(x) - S_j x = w_j\} \quad (10.6)$$

$$NA(x) := \{j \in I : G_j z^*(x) - S_j x < w_j\} \quad (10.7)$$

The so-called *critical region* CR_A is then defined as the set of parameters x for which the same set of constraints $A \subseteq I$ is active at the optimum:

$$CR_A := \{x \in \mathcal{K}^* \mid A(x) = A\} \quad (10.8)$$

The control law that is affine on these critical regions can be found through the KKT conditions. A corresponding example can be found in the lecture slides. Similarly, it has been shown that also the 1- or ∞ -norm continuous finite-time optimal control problem resulted in a continuous and affine control law on polyhedral critical regions $CR^j = \{x \in \mathbb{R}^n \mid H^j x \leq K^j\}$ (solution of the multi-parametric linear program).

Once the affine control law for each of the polygons spanning up the feasible set of states \mathcal{X} has been defined, any state just needs to be assigned to the correct critical region in order to compute the correct control input from the control law. The simplest way for a small number of critical regions is a simple *sequential search* where the algorithm iterates over all critical regions, until the queried state x belongs to one. To achieve logarithmic costs, the regions can be sorted into a binary search tree that allows for more efficient querying. This type of online evaluation is very fast, but only feasible for small systems with approximately 3-6 states.

10.2 Iterative Optimization Methods

To simplify the notation in this section, only linear and quadratic programs in their standard form with equality and inequality constraints will be considered. The linear program reads then:

$$\min c^T x \quad (10.9)$$

$$Ax = b \quad (10.10)$$

$$Gx \leq f \quad (10.11)$$

The quadratic program can be written as follows (cost function is convex for $H \geq 0$):

$$\min \frac{1}{2}x^T Hx + c^T x \quad (10.12)$$

$$Ax = B \quad (10.13)$$

$$Gx \leq f \quad (10.14)$$

As analytical solutions often do not exist or are difficult to be found, iterative numerical optimization methods are chosen that produce a sequence of iterates starting at some initial guess $x^{(0)}$ until $x^{(m)}$ and yield a minimum decrease during each step while satisfying $x \in \mathbb{Q}$ (m is variable to meet this condition; δ and ϵ are user defined tolerances):

$$x^{(i+1)} = \Psi(x^{(i)}, f, \mathbb{Q}), \quad i = 0, 1, \dots, m-1 \quad (10.15)$$

$$\text{s.t. } |f(x^{(m)}) - f(x^*)| \leq \epsilon, \quad \text{dist}(x^{(m)}, \mathbb{Q}) \leq \delta \quad (10.16)$$

Unconstrained Minimization Principle

Assuming that the objective function $f(x)$ is twice differentiable and convex and that the optimal value $p^* = \min_x f(x)$ is finite, the global minimizer x^* has to satisfy $\nabla f(x^*) = 0$ (necessary and sufficient condition). Unconstrained optimization methods try to iteratively find the point where this condition is fulfilled. *Descent methods* then choose a descent direction Δx and stepsize h such that the value of the objective function decreases steadily towards the minimum:

$$x^{(i+1)} = x^{(i)} + h^{(i)} \Delta x^{(i)}, \quad \text{with } f(x^{(i+1)}) < f(x^{(i)}) \quad (10.17)$$

The corresponding algorithm starting at $x^{(0)} \in \text{dom}(f)$ can be written as follows:

1. Compute a descent direction Δx
 2. Line search: Choose a step size $h^{(i)} > 0$ such that $f(x^{(i)} + h^{(i)} \Delta x^{(i)}) < f(x^{(i)})$
 3. Update $x^{(i+1)} := x^{(i)} + h \Delta x^{(i)}$
- Stop iteration once $f(x^{(m)}) - f(x^*) \leq \epsilon_1$ or $\|x^{(m)} - x^{(m-1)}\| \leq \epsilon_2$

In order to speed up convergence, choose an initial condition $x^{(0)}$ of the iterative algorithm that is already close to the optimal solution (*warm start*). In MPC the warm start initial condition is usually chosen as the estimated state solution from the previous timestep.

Gradient Methods

Gradient descent methods choose the negative gradient direction as descent direction to go directly towards the global minimum (for convex objective functions):

$$x^{(i+1)} = x^{(i)} - h^{(i)} \nabla f(x^{(i)}) \quad (10.18)$$

Assuming that the gradient of the objective function is Lipschitz continuous with constant L , i.e. $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, the stepsize can be chosen to be constant with:

$$h^{(i)} = \frac{1}{L} \quad (10.19)$$

If the constant L is not known exactly, this method might lead to slow convergence or oscillations.

Newton's Method

Instead of minimizing the objective function $f(x)$ itself, Newton's method uses the second-order approximation $\tilde{f}(x, x^{(i)})$ around the current iterate $x^{(i)}$:

$$x^{(i+1)} = \min_x f(x^{(i)}) + \nabla f(x^{(i)})^T (x - x^{(i)}) + \frac{1}{2} (x - x^{(i)})^T \nabla^2 f(x^{(i)}) (x - x^{(i)}) \quad (10.20)$$

Note that this approximation $\tilde{f}(x, x^{(i)})$ is not necessarily either an upper bound or a lower bound of the original objective $f(x)$. Computing the derivative of the second-order approximation yields the descent direction for Newton's method, which is the product of the negative gradient direction and the inverse of the Hessian:

$$\Delta x_{nt} = - \left(\nabla^2 f(x^{(i)}) \right)^{-1} \cdot \nabla f(x^{(i)}) \quad (10.21)$$

As the approximation is not necessarily an upper bound of the function, the stepsize $h^{(i)}$ has to be chosen carefully in order to satisfy the descent condition. The Newton step can be written as follows:

$$x^{(i+1)} = x^{(i)} - h^{(i)} \left(\nabla^2 f(x^{(i)}) \right)^{-1} \cdot \nabla f(x^{(i)}) \quad (10.22)$$

One possible *line search* method to find the best stepsize $h^{(i)}$ is to compute it as the solution of the following one-dimensional optimization problem:

$$h^{(i)*} = \arg \min_{h>0} f(x^{(i)} + h^{(i)} \Delta x_{nt}) \quad (10.23)$$

As this approach is computationally expensive, the stepsize is often chosen using the following iterative (but inexact) backtracking approach with $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$:

```
Initialize  $h^{(i)} = 1$   
While  $f(x^{(i)} + h^{(i)} \Delta x_{nt}) > f(x^{(i)}) + \alpha h^{(i)} \nabla f(x^{(i)})^T \Delta x_{nt}$  do  $h^{(i)} \leftarrow \beta h^{(i)}$ 
```

10.3 Constrained Minimization

Projected Gradient Descent

The projected gradient descent method is based on the same assumptions as the discussed gradient method with the additional capability to take into account a convex constraint set \mathbb{Q} . Instead of just using the result of the descent iteration, it is projected back into the constraint set:

$$x^{(i+1)} = \pi_{\mathbb{Q}}(x^{(i)} - h^{(i)} \nabla f(x^{(i)})) \quad (10.24)$$

The projection operator $\pi_{\mathbb{Q}}$ moves a point y to the point $x \in \mathbb{Q}$ as follows:

$$\pi_{\mathbb{Q}}(y) = \arg \min_x \frac{1}{2} \|x - y\|_2^2 \quad (10.25)$$

$$\text{s.t. } x \in \mathbb{Q} \quad (10.26)$$

If there are both input and state constraints present in the system, it might make sense to dualize the problem as individual projections are easy, but intersections can become expensive.

Interior-Point Methods

Interior-point methods generally need significantly less iterations than projected gradient descent, all of which are more expensive with respect to computational time. At the same time, the number of iterations stays roughly constant throughout the MCP problem. The algorithm is defined for optimization problems of the following form where f and g_i are convex and twice differentiable, $f(x^*)$ is attained and finite, there exist points $\tilde{x} \in \text{dom}(f)$ that are strictly feasible and the feasible set is compact:

$$\min f(x) \quad (10.27)$$

$$\text{s.t. } g_i(x) \leq 0, \quad i = 0, 1, \dots, m \quad (10.28)$$

$$Cx = d \quad (10.29)$$

The solution approach is to solve the constrained problem as an unconstrained problem with equality constraints and slack variable $s \in \mathbb{R}^m$ (relaxed KKT system):

$$Cx^* = d \quad (10.30)$$

$$g_i(x^*) + s_i^* = 0, \quad i = 0, 1, \dots, m \quad (10.31)$$

$$\nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) + C^T \nu^* = 0 \quad (10.32)$$

$$\lambda_i^* g_i(x^*) = -\kappa, \quad i = 0, 1, \dots, m \quad (10.33)$$

$$\lambda_i^*, s_i^* \geq 0, \quad i = 0, 1, \dots, m \quad (10.34)$$

By solving the primal and dual problems at the same time through the system above, a primal-dual central path $\{(x, \nu, \lambda, s)\}$, possibly depending on the variable κ , results. The optimal solution of the problem can then be found by following the path towards $\kappa = 0$.