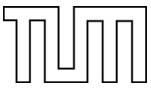


Technische Universität München

Department of Mathematics

Master's Thesis

Sequential Convex Programming in Nonlinear Model Predictive Control

Philipp Franz-Xaver Fröhlich

Supervisor: Prof. Dr. Michael Ulbrich

Advisor: Dr. Sebastian Albrecht

Submission Date: April 15, 2016

I hereby declare that this thesis is my own work and that no other sources have been used exept those clearly indicated and referenced.

Munich, April 15, 2016

Zusammenfassung

Zahlreiche technische Systeme, wie zum Beispiel Wasserkraftanlagen oder Autos, können mit modellprädiktiver Regelung autonom gesteuert werden. Für Echtzeitanwendungen ist eine schnelle und effiziente Lösung der einzelnen Regelungsprobleme erforderlich. Oft ist dabei die Auswertung der Nebenbedingung und ihrer Ableitungen sehr zeitintensiv. Die Lösung eines geeigneten linearisierten Subproblems kann diese Auswertungen erheblich verkürzen, hat jedoch zur Folge, dass die optimale Lösung nur noch annähernd gefunden wird. Diese Arbeit basiert auf der Veröffentlichung [19] in mit dem *adjungierte Prädiktor Korrektor SCP Verfahren* ein solches Verfahren vorgestellt wird. Darüber hinaus kann gezeigt werden, dass unter bestimmten Voraussetzungen der Approximationsfehler für alle Zeitpunkte von einer Konstante beschränkt wird.
In dieser Arbeit werden zunächst die grundlegenden Konzepte der modellprädiktiven Regelung erklärt und dann das Verfahren ausführlich besprochen. Die Modellierung eines Pumpspeicherkraftwerks und Automobils wird diskutiert, erweitert und in verschiedenen Szenarien angewendet, um die numerische Eigenschaften des Verfahrens zu testen. Darüber hinaus wird auch der Einfluss von parallelisierter Programmierung, Zeitdiskretisierung und externen Fehlern untersucht.

Acknowledgements

Eventhough this thesis has only a single author, this work would not have been possible without the help of many people. First and foremost, I would like to thank Prof. Dr. M. Ulbrich for this challenging and fascinating research topic.

I would like to express my thanks and recognition to my advisor Dr. S. Albrecht for his useful guidance. His ability to provide answers to questions the author is hardly able to formulate is astonishing.

Further, a special thanks goes to my friends, especially A. Brandl and Dr. T. Satzger, for their support and feedback.

Finally, I would like o thank my beloved parents and grandparents. Their mental and financial support gave me the opportunities for the entire study.

Contents

1. Theory

1.1. Introduction

1.1.1. Motivation

Technical systems, such as hydro power plants with multiple turbines and pumps, have a large number of adjusting screws. Tweaking only one screw might influence the state of various components, which makes the task of finding the best adjustment very difficult for humans. If the system is monitored with sensors or has some inputs with varying quantities (e.g. a river providing an changing flow of water into the system) the system has to adapt to the changed situation or measurement errors. If it is not possible to turn the system off, a nonstop adjustment of these control screws is required.
Automated control of such technical systems has numerous benefits, as it helps to lower manpower costs and the risks of fatal human faults. Furthermore, it allows running the system with maximal efficiency. One of the most prominent automated control approaches in this task is model predictive control (MPC) also known as receding horizon control. In this setting the system is modeled and based on that model, the control with the best predicted result is executed. Compared to other approaches like proportional–integral–derivative (PID) controllers, the design of a MPC controller is more time consuming and its application requires more computational effort. But it allows the control of more complicated systems and obtains better results. The origins of MPC lay in the control of oil refineries and chemical plants in the 1980s as the involved processes were slow enough to solve the underlying optimization problems.
From a mathematical point of view, this approach leads to a series of parametric optimization problems, where the parameter represents the change in the environment. As a consequence of the MPC approach, the solution must be found in a limited time frame. Instead of solving every problem separately, an online algorithm solves the optimization problem with Gauss-Newton, one step SQP [14] or interior points method [15, 43] [6, pages 245-267]. A relatively new iterative programming idea is sequential convex programming, which came up in the early 2000s [21, 22]. Instead of solving a sequence of quadratic approximations of the original program as in the SQP method, a series of convex approximations is solved. In many applications, the calculation of the exact optimal control is too time consuming to establish a real-time control. In this cases, calculating a fast approximation of the optimal control is often sufficient [14].

Outline

This thesis is organized in the subsequent way. In section 1.1 the concept of *real-time receding horizon control* is motivated and the real-time iteration (Algorithm 2) is introduced.

In every iteration, a parametric optimization problem must be solved. The remainder of chapter 1 discusses an approximative solver for theses problems, which is known as *adjoint-based predictor-corrector sequential convex programming* (APCSCP) [19]. The three main ideas of this approach are individually explained in section 1.2. In section 1.3 these pieces are put together for the formulation of Algorithm 4 and a contraction result is shown and discussed in detail. In chapter 2 an extended model of a hydro power plant with pump storage facilities from [19, 29] is explained and tested in different settings to evaluate the performance of the introduced solver. In chapter 3 a simple car model is discussed and applied as a traffic jam assistant. The effects of different time discretizations and parallelization on the performance of APCSCP are evaluated. In appendix A notation and elementary concepts are introduced.

1.1.2. Real-time Optimal Control

In this chapter the general optimization problem P (1.1) is defined, to discuss the basic terminology of control theory. A more comprehensive introduction can be found in [9]. From this not necessarily solvable problem the idea of *receding horizon control* and *real-time iteration* is deduced, which is able to compensate occuring disturbances and errors. In order to implement and solve the obtained problems on computers, the problem is discretized and this leads to problem $P_T(\xi)$ (1.9). The time efficient solution of this problem is then the main subject in section 1.3.

Global problem

The optimal control problem P has the form:

$$\begin{aligned} \min_{x} \quad & f(x), \\ \text{s.t.} \quad & \frac{\partial}{\partial t} w = d(x), \\ & w(0) = w_0, \\ & x(t) \in \Omega \ \forall t > 0. \end{aligned} \tag{1.1}$$

The optimization function $x = (w, u), x \in \mathscr{C}\left[\mathbb{R}, \mathbb{R}^{\widetilde{n}_{opt}}\right], \widetilde{n}_{opt} \in \mathbb{N}$ describes the whole system at time point $t \in \mathbb{R}, t \geq 0$. The *control* $u \in \mathscr{C}\left[\mathbb{R}, \mathbb{R}^{n_{control}}\right], n_{control} \in \mathbb{N}$ can be chosen freely within the bounds given in the constraints and does not depend on other components of the system. The *state* $w \in \mathscr{C}[\mathbb{R}, \mathbb{R}^{n_{state}}], n_{state} \in \mathbb{N}$ describes the state of all components, which are not directly controllable. The *dynamics* $d(x) \in \mathscr{C}\left[\mathbb{R}^{\widetilde{n}_{opt}}, \mathbb{R}^{n_{dyn}}\right]$ of the system describes the behaviour. The initial setting of the system is described in $w_0 \in \mathbb{R}^{n_{state}}$. The feasible set Ω is closed and convex. The goal function $f : \mathscr{C}\left[\mathbb{R}, \mathbb{R}^{\widetilde{n}_{opt}}\right] \mapsto \mathbb{R}$ maps the optimization function x to $\mathbb{R}$ and has to be continuous, convex, and differentiable. An exemplary goal function f might be

$$f(x) = \int_0^\infty \left\| w(\tau) - w^{des}(\tau) \right\|_{\mathfrak{W}}^2 + \left\| u(\tau) - u^{des} \right\|_{\mathfrak{U}}^2 \mathrm{d}\tau. \tag{1.2}$$

The desired state and control are $w^{des}(t) \in \mathscr{C}$ and $u^{des} \in \mathbb{R}^{n_{control}}$. The operators $\mathfrak{W}, \mathfrak{U} \in \mathcal{S}_{++}$ are symmetric positive definite weighting operators.

In this general formulation problem P does not have to be solvable. There is in fact, a great number of examples for unsolvable instances of P. This thesis always assumes, that the considered optimization problem has a solution $\bar{x}$. Note that the desired state and control do in general not define a feasible solution of the problem. The optimal solution $\bar{x}$ is closest to desired state and control.

Remark 1.1. The exemplary goal function f is typical for optimal control applications. The systems state must be kept at a desired value with a minimal usage of the controls. The desired control u^{des} can be seen in many applications as the control with the least cost.

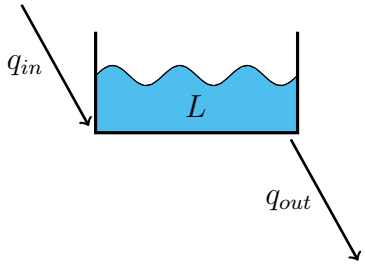


Figure 1.1.: Visualization of the model described in the Example (section 1.1.2). There is river providing a constant flow q_{in} of water into the lake L and a controllable flow q_{out} flowing out of the lake.

Example: Lake height control Assume that there is a lake L given with water flowing into $(q_{in}(t))$ and out of $(q_{out}(t))$ it, see figure 1.1. $q_{in}(t)$ is constant. It is possible to control the amount of water flowing out of it, so this is the *control*. The goal is to keep the lake height h_L as close as possible to its initial value $h_L(0)$. This system is fully described by $x(t) = (q_{in}, h_L, q_{out})(t)$, $\mathbb{R} \mapsto \mathbb{R}^3$. In this setting the *state* is (q_{in}, h_L), where q_{in} is not and h_L is just indirectly controllable. The *dynamics* describe the system's behaviour (e.g. the water level rises if in total water flows into the lake). A more detailed description of the dynamics is given in section 2.1.1. The goal function measures the distance between $h_L(t)$ and the desired level. But for this simple example, the intuitive understanding might be sufficient. If $q_{out}(t)$ is chosen constant as q_{in}, the total inflow of the lake is 0 at every time point. When the amount of water in the lake stays constant, the lake height stays constantly at $h_L(0)$ and an optimal control is found.

Remark 1.2. If the feasible set has the form $\Omega(t) = \Omega_w(t) \times u_{sim}(t)$, the global optimal control problem turns into a simulation of the system when the control u_{sim} is applied. This means that finding the global solution of P is equivalent to simulating the system with all possible controls and selecting the one which leads to the smallest goal function value $f(x)$.

Real-time iteration

Unpredictable external forces or errors in the execution of the optimal control $\bar{u}$ influence the system. In the following it is assumed that the resulting error is continuous. In this setting the precomputed controls $\bar{u}$ are not able to compensate this error and over time the disturbed state trajectory may diverge from the optimal trajectory $\bar{w}$. For the computation

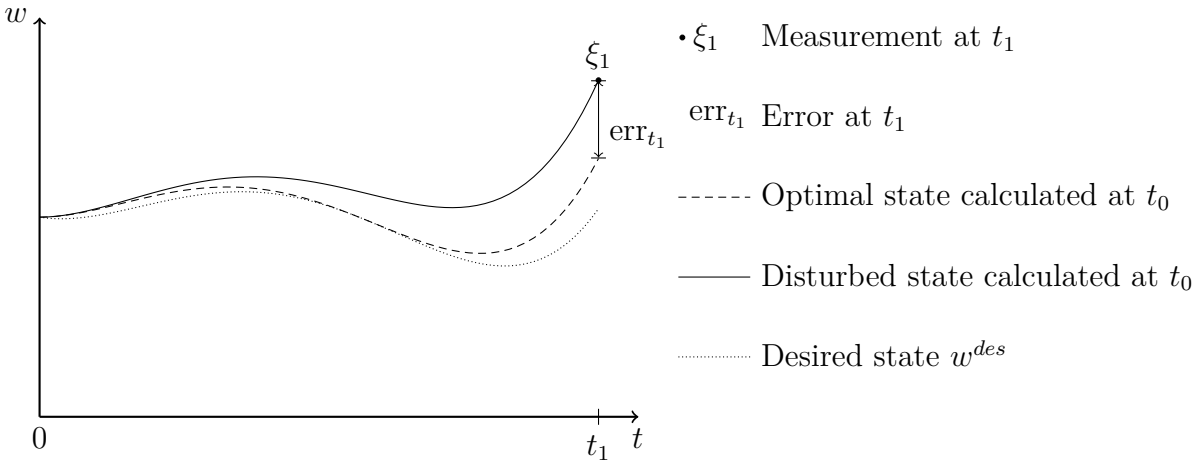


Figure 1.2.: Visualization of the global optimal control. If there are no external errors, the system will follow the calculated optimal state trajectory when the optimal control is applied. External errors occur in a majority of real world applications (e.g. the optimal control is not executed exactly). This leads to a disturbance of the optimal state trajectory and the actual trajectory may diverge over time from the optimal trajectory.

of error compensating controls, the system must be monitored. For this purpose, sensors are installed measuring the actual state of the system. The *measurement* at time point t_k is called $\widehat{\xi}_k$. If the measurement is correct the error at t_k is $\text{err}_{t_k} = \|w_k - \widehat{\xi}_k\|$, see figure 1.2. Note that in general sensors are faulty as well. But the distance between measurement and the actual state is bound by the sensor's accuracy. The measurement error will be part of the overall error in the next iteration step. The measured actual state $\widehat{\xi}_k$ is then used as an initial value for a new global optimal control problem P. The problem returns a new optimal value $\bar{x}^k = (\bar{w_k}, \bar{u_{k+1}})$ compensating err_{t_k}. At t_{k+1} the sensors measure $\widehat{\xi}_{k+1}$ and a new iteration starts, see Algorithm 1 and a visualization in figure 1.3. This method is able to compensate errors. The worst case computational costs multiply by the number of simulated time steps.

Algorithm 1 Real time iteration

Require: Set the initial state w_0 as $\widehat{\xi}_0$

1: **loop** $k \in \{0, 1, \dots\}$

2: Get new measurement $\widehat{\xi}_k$ at t_k.

3: Solve the optimal control problem $P_T(\xi)$ with $w_0 = \widehat{\xi}_k$ at t_k to get a new optimal control $u_k(t)$.

4: Send u_k to control unit.

5: **end loop**

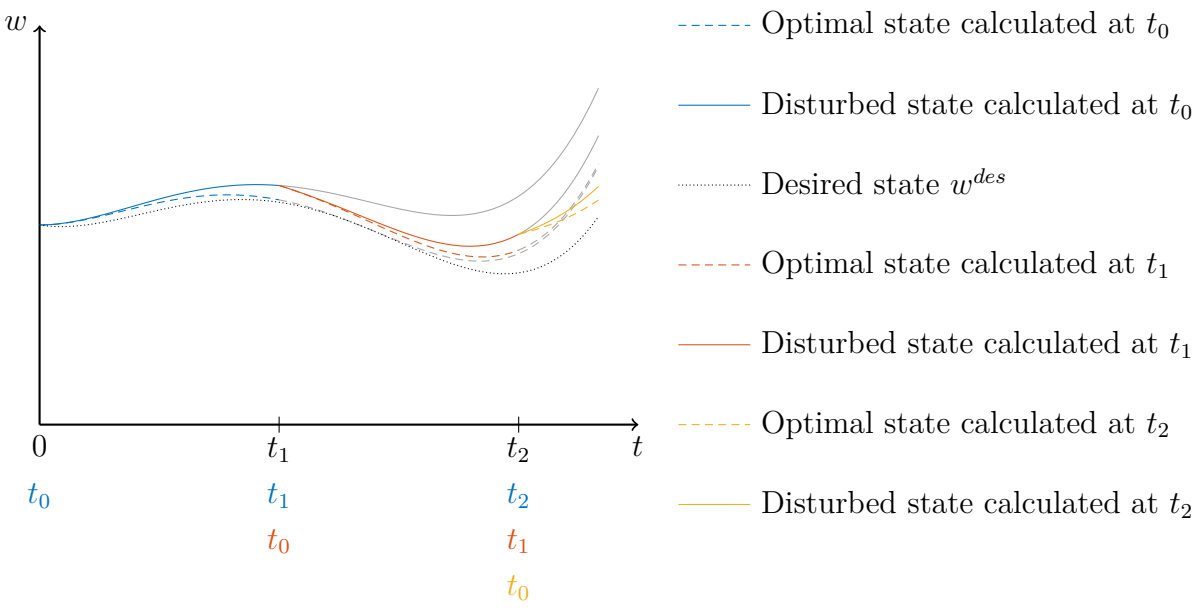


Figure 1.3.: Visualization of the real-time iteration idea.

Example: Lake height control If q_{in} is faulty and provides more water than specified, the control obtained above inevitably leads to an increasing water level. At some point, the maximum lake height is exceeded and the vicinity is flooded. This can be prevented by an additional sensor providing *measurements* $\{\widehat{\xi}\}_{k\geq 0}$ of the lake height. In every iteration $\widehat{\xi}$ is the initial value for a new optimal control problem. If the water level is measured higher as $h_L(0)$, the outflow q_{out} will be larger than q_{in}. Moreover, if q_{out} was set too high in the previous iteration steps and the lake height is too small, q_{out} will be smaller than q_{in}. This approach may prevent the obtained state trajectory from diverging.

Global and local time frame As visualized in figure 1.3, there are two different notations for time points. In the *global* time (black label), $t_0 = 0$ is the beginning of the first optimal control problem. In global time, the time point t_i denotes the ith measurement $\widehat{\xi}_i$. The ith *local* time (colored labels) $t_0 = 0$ corresponds to t_i in global time. This is the beginning of the ith optimal control problem. In the following, global time is used when the focus is on the real-time iteration. Local time is used when the scope is on the solution of a single optimal control problem. This thesis maintains that the global time is the 0th local time. Every time point t_i where a new measurement $\widehat{\xi}_i$ is obtained, is the beginning of a new optimal control problem. In the ith local frame $\widehat{\xi}_i$ is denoted as $\widehat{\xi}$, as this is the only relevant measurement for this subproblem. With this distinction multiindices can be avoided. The measurement time steps will be equidistant with a step size of Δt.

Receding horizon control

In time step t_k of the real-time iteration Algorithm 1 a solution $\bar{x}^k$ for the interval $[t_k, \infty]$ is calculated. For the real-time iteration it would be sufficient to know the solution $\bar{x}$ only for the actual interval $[t_k, t_{k+1}]$, because in time point t_{k+1} a new control $\widehat{u}_{k+1}$ will be executed. Therefore, the global optimal control problem P (1.1) can be simplified to the

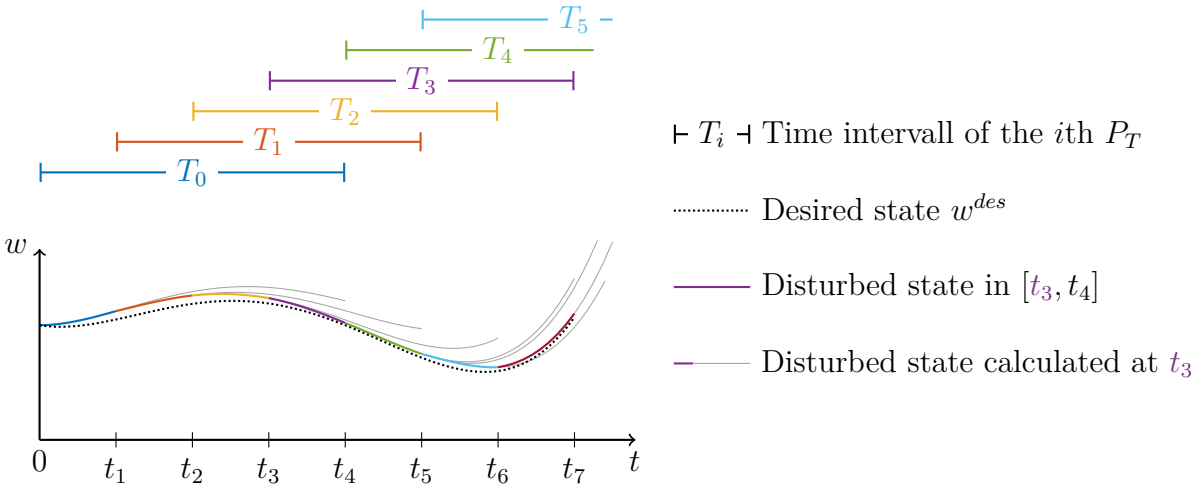


Figure 1.4.: Visualization of the receding horizon control idea.

continuous receding horizon problem $\widetilde{P}_T$

$$\begin{aligned} \min_{x} \quad & f(x(\cdot)) \\ \text{s.t.} \quad & \frac{\partial}{\partial t} w = d(x), \\ & w(0) = w_0, \\ & x(t) \in \Omega \ \forall t \in [0, T], \\ & w(T) \in \mathcal{R}_T. \end{aligned} \tag{1.3}$$

f maps $\mathscr{C}(\mathbb{R}, \mathbb{R}^{\widetilde{n}_{opt}})$ to $\mathbb{R}$ and has to be convex and differentiable. The exemplary f equation (1.2) is then:

$$f(x(\cdot)) = \int_0^T \left\| w(\tau) - w^{des} \right\|_{\mathfrak{W}}^2 + \left\| u(\tau) - u^{des}(\tau) \right\|_{\mathfrak{U}}^2 \mathrm{d}\tau + \left\| w(T) - w^{des} \right\|_{\mathfrak{S}}^2 . \tag{1.4}$$

$\mathfrak{S} \in \mathcal{S}_{++}$ is a symmetric positive definite weight operator and the terminal region is $\mathcal{R}_T = \left\{ w \in \mathbb{R}^{n_{state}} \middle| \left\| w - w^{des}(T) \right\|_{\mathfrak{S}}^2 \leq \mathfrak{r} \right\}$. This additional term ensures, that the system at time point T is close enough to the desired state. T is called *horizon*, as this variable defines how far the control looks into the future. The real-time iteration with $\widetilde{P}_T$ as subproblems is visualized in figure 1.4.

Remark 1.3. As $\widetilde{P}_T$ (1.3) only considers the system behaviour in the interval $[0, T]$, the obtained solution $\bar{x}$ is just valid in this interval and the behaviour was optimized on this interval. In general, the solution $\bar{\widetilde{x}}$ of P and $\bar{x}$ differ on $[0, T]$.

1.1.3. Discretization

Problem $\widetilde{P}_T$ (1.3) is formulated with continuous functions and ordinary differential equations. Although optimization over function spaces is a wide and interesting field, which

is, under additional assumptions, applicable to $\widetilde{P}_T$ (1.3), a different approach is chosen here. The majority of computer controlled actuators requires a discrete input signal and sensors deliver a discrete measurement. Therefore, the problem is discretized first to meet the requirements of the system and to reduce the computational effort. This means that instead of applying a discretized optimal continuous function, an optimal discretized function is applied. In the following, the discretization of the receding horizon control problem $\widetilde{P}_T$ (1.3) is discussed in local time frame. At first the multiple shooting approach is applied to the dynamics constraint. The discretization of the feasible set constraint and the goal function follows the definitions in the multiple shooting approach.

Multiple Shooting

The discretization of the constraint $\frac{\partial}{\partial t}w = d(x)$ is done with the *multiple shooting* approach. The origins of this method where developed in the early 1980s [5] and were adapted to nonlinear control problems in the early 2000s [14]. In this approach $[0, T]$ is partitioned in smaller intervals. This allows parallelization[1] and transforms the differential equation into a nonlinear equality constraint.

Time The controlling computer observes the system in a discrete way, as every sensor signal has to be converted in a digital signal. The interval $[0, T]$ is partitioned in $n_{ms} - 1$ intervals where $n_{ms} \in \mathbb{N}$ is the number of time points and

$$\begin{aligned} [t_{k-1}, t_k], k \in \{1, \dots, n_{ms} - 1\}, \\ t_0 = 0,\ t_{n_{ms}} = T, \\ t_k - t_{k-1} = \widehat{\Delta t}\ \forall k \in \{1, \dots, n_{ms}\}. \end{aligned} \tag{1.5}$$

The constant $\widehat{\Delta t}$ is called the shooting step size, in the following we assume $\widehat{\Delta t} = \Delta t$. This choice leads to the fact that t_1 of the previous control problem is t_0, see section 1.1.2. In general Δt should be an integer multiple of $\widehat{\Delta t}$. The next real-time iteration starts at a time point with a multiple shooting node. A shifted solution of the actual solution can be used as an initial value for the next optimization problem.

Control In the following, we assume that the control u is piece-wise constant in every shooting interval $[t_{k-1}, t_k)$. The discretized control is therefore a multi-dimensional step function, which is in general not continuous. If the control actuators' reaction time is notably smaller than Δt this simplification leads to errors. However, the receding horizon control is designed to compensate errors and therefore the resulting error will be compensated in the next iteration.

Shooting Nodes For every time point t_k a corresponding *shooting node* $s_k \in \mathbb{R}^{n_{state}}$ exists. When $w(s_k, u_k)$ is the integration of the differential equation $\frac{\partial}{\partial t}w = d(x)$ with initial value (s_k, u_k) over the interval $[t_k, t_{k+1}]$, the dynamics constraint can be transformed

[1] see section 3.2.2

into

$$\begin{pmatrix} w_0 - \xi \\ w(s_0, u_0) - s_1 \\ \vdots \\ w(s_{n_{ms}-2}, u_{n_{ms}-2}) - s_{n_{ms}-1} \end{pmatrix} = g(x) + M\xi = 0. \tag{1.6}$$

$M \in \mathbb{R}^{n_{eqC} \times p}$ embeds the measurement ξ linearly. If it is possible to measure all states, M contains the negative identity matrix in the first n_{state} rows. But this approach is also valid, when the sensors are not able to measure all components, or when the result of a sensor is a linear combination of multiple states. This leads to a nonlinear and differentiable equality condition, visualized in figure 1.5.

State The state function w is represented by evaluations at all shooting points $t_0, \ldots, t_{n_{ms}}$. As mentioned in Remark 1.2, the state at other time points can be computed afterwards with the dynamics of the system and known controls.

Optimization value In the continuous setting, the optimization value x is defined as (w, u). Therefore, its discretization follows from state and control. The discretized vector x has the form

$$x = \left(s_0^T, u_0^T, s_1^T, u_1^T, \ldots s_{n_{ms}-1}^T, u_{n_{ms}-1}^T, s_{n_{ms}}^T\right)^T \in \mathbb{R}^{n_{opt}} \tag{1.7}$$

with $n_{opt} = n_{ms}(n_{state} + n_{control}) + n_{state}$. Note that shooting nodes and control can be arbitrarily permuted as long as it is done consistently.

In the implementation, a order by the system's components is has the advantage that is convenient to implement the integration vector-wise. The order shown above has the advantage that the conversion between consecutive local time frames is just a shift by $n_{state} + n_{control}$ entries. The choice of Δt influences n_{ms} and therefore the problem size, see section 3.2.3. In the discrete setting, time points t_k are often identified with their index k. The notations $x(k)$ and x^k for $x(t_k)$ are also common. This holds for controls u and states w as well.

Computational consequences With this approach it is possible to run the integrations of the shooting interval in parallel. Many CPUs are equipped with 4 or 8 processors and so the computation time for this task on these CPUs is reduced by a factor up to $\frac{1}{4}$ resp. $\frac{1}{8}$. As the multi-threaded programming requires more memory and these are only theoretical factors, see section 3.2.2.

Feasible Set

The system has to be feasible at every time point $t_0, \ldots, t_{n_{ms}}$. Due to the fact that the control u is constant in every interval $[t_k, t_{k+1})$ and Ω convex, there is no weakening in the control subspace. This does not hold for the state. By reason of continuity of w and a small step size Δt, the violation can be bounded. The discretization must preserve closedness and convexity.

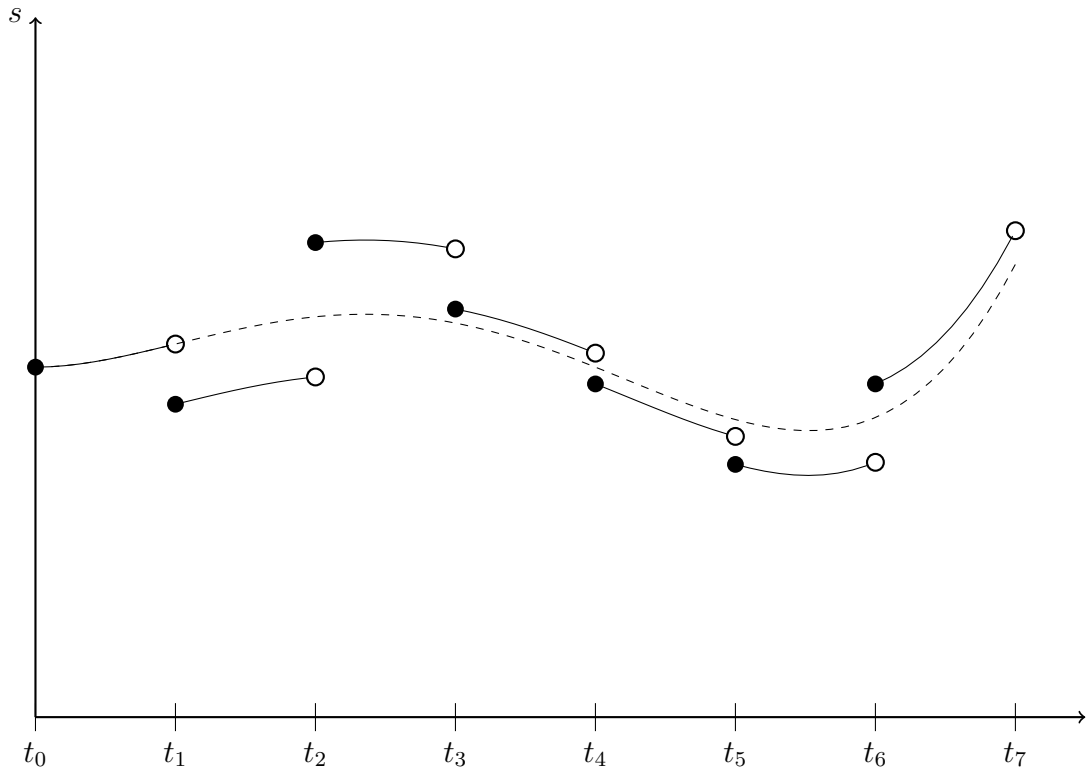


Figure 1.5.: Visualization of the multiple shooting idea. The distance between s_{k+1} and $w(s_k, u_k)$ must be zero. Otherwise, the dynamics constraint does not hold for the whole interval $[0, T]$.

Goal function

The exemplary goal function f (1.4) is discretized to

$$f(x) = \sum_{k=0}^{n_{ms}-2} \left[\|(s_k - {w^{des}}_k)\|_{\mathcal{W}}^2 + \|u_k - u^{des}\|_{\mathcal{U}}^2 \right] + \|s_{n_{ms}-1} - {w^{des}}_{n_{ms}-1}\|_{\mathcal{S}}^2 \tag{1.8}$$

with $\mathcal{W}, \mathcal{U}$ and $\mathcal{S} \in \mathcal{S}_{++}$ symmetric positive definite matrices approximating the operators $\mathfrak{W}, \mathfrak{U},$ and $\mathfrak{S}$. In general it is required that $f \in \mathscr{C}\left[\mathbb{R}^{n_{opt}}, \mathbb{R}\right]$ is convex and differentiable.

Summary

The discretization of the receding horizon control problem $\widetilde{P}_T$ (1.3) is denoted as $P_T(\xi)$ and has the following form:

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & f(x) \\ \text{s.t.} \quad & g(x) + M\xi = 0, \\ & x \in \Omega. \end{aligned} \tag{1.9}$$

With $f : \mathbb{R}^{n_{opt}} \mapsto \mathbb{R}$ convex and differentiable, $g : \mathbb{R}^{n_{opt}} \mapsto \mathbb{R}^{n_{eqC}}$ nonlinear and differentiable, $\emptyset \neq \Omega \subseteq \mathbb{R}^{n_{opt}}$ a closed convex set and the measurement $\xi \in \mathcal{P} \subseteq \mathbb{R}^p$ is a parameter of the multiple shooting condition. The state of the system and the measurements change slowly in between two time steps, i.e. $\{\xi_k\}_{k\geq 0}$ with $\|\xi_{k+1} - \xi_k\|$. The matrix $M \in \mathbb{R}^{n_{eqC} \times p}$ embeds the measurement ξ linearly in the equality constraint. It is possible to solve this problem with standard solvers like `IPOpt`, see section 2.2.1. Another algorithm solving this problem faster is explained in detail in section 1.3 and is the main topic of the thesis.

1.1.4. Real-time Algorithm

Algorithm 1 has a major weakness. The time point when u_k is applied in the formulation of the subproblem is t_k (in global frame) and this is also the time point when the solution of this subproblem begins. The solution if this subproblem takes time $t_{calc,k}$ and during this time the system changes. This means that the computed control is not optimal when it is applied. This is avoided by simulating the system from t_k to t_{k+1} and calculating the control for interval t_{k+1}, t_{k+2} with subproblem k, see Remark 1.2. The computational costs of Algorithm 2 increase as the horizon must be increased by Δt to compensate the simulation step. If it is possible to guarantee, that $t_{calc,k} < \Delta t, \forall k \geq 0$ this algorithm runs successfully.

Algorithm 2 Real time iteration with precomputed controls

Require: Set the initial state w_0 as ξ_0, compute optimal control u_0 for interval $[0, t_k)$

1: **loop** $k \in \{0, 1, \dots\}$

2: Get new measurement ξ_k at t_k.

3: Send u_k to control unit.

4: Solve the optimal control problem $P_T(\xi)$ with $w_0 = \xi_k, u_0 = u_k$ to get a new optimal control u_{k+1}.

5: **end loop**

1.2. Main Principles of the APCSCP Algorithm

In this chapter, the three main ideas of the APCSCP approach, namely *sequential convex programming*, *predictor-corrector methods*, and *adjoint based methods* are explained separately. This serves as a preparation of section 1.3 where these ideas are put together during the discussion of the APCSCP method.

1.2.1. Sequential Convex Programming

As the *sequential convex programming* approach (SCP) can deal with semi-definite problems an other name is sequential semi-definite programming (SSDP). First introduced in [21] with affine linear semi-definiteness constraints and a proof of local superlinear and quadratic convergence, the approach was extended in [10] where a global convergence result is shown. In the following, we stick to the work of [23], which includes semi-definiteness, inequality and equality constraints. This approach is related to the SQP method and has shown good results in practice as well [12, 19]. [38] extends this approach to solve matrix-valued optimization problems.
In the beginning, the problem and the necessary assumptions are introduced. The semi-definitness constraints do not occur as this constraint type is not needed in the rest of the thesis. Then the linearization of the first order optimality condition leads to the SCP subproblem and the formulation of the SCP algorithm. At the close of this subsection, a contraction result is given.

Problem Formulation and Assumptions

In this section the following problem P_{gen} is discussed:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{s.t.} \quad & h(x) \leq 0, \\ & \widehat{g}(x) = 0. \end{aligned} \tag{1.10}$$

Where $c \in \mathbb{R}^n$ is a vector and $h \in \mathscr{C}^3\left[\mathbb{R}^n, \mathbb{R}^{m_1}\right], \widehat{g} \in \mathscr{C}\left[\mathbb{R}^n, \mathbb{R}^{m_2}\right]$ are vector valued functions. Furthermore, we assume that the problem has a locally unique and strictly complementary solution $\bar{x}$ and it satisfies the Slater's condition; see (A.2.1). At $\bar{x}$ the

Robinson constraint qualification (CQ) (A.2.2) and the second-order sufficient condition hold. For a profound discussion of the second-order sufficient condition can be found in [23, page 594] [34, page 202 ff.] [41, page 102 ff.].

Remark 1.4. The assumption that Slater's condition and the second-order sufficient condition holds at the locally unique and strictly complementary solution $\bar{x}$ is sufficient to ensure, that problem P_{gen} (1.10) has a unique solution for small perturbation of the data $(c, , h, \widehat{g})$ and this solution depends smoothly on the perturbation, see [23, page 601]. $\mathscr{C}^3$ differentiability and the Robinson CQ are needed to proof Theorem 1.1.

Remark 1.5. In contrast to $P_T(\xi)$ (1.9), the goal function is linear. In section 1.3.2 is shown how problems with convex goal functions can be transformed into problems with linear goal functions. The subproblems in Algorithm 2 can be solved with SCP if it is possible to represent the feasible set Ω with a finite set of $\mathscr{C}^3$ equalities and inequalities and the postulated assumptions hold. For the numerical examples in this work this is possible, as the feasible set is a bounding box which only represents the lower and upper bounds of some components. Note that the assumption of $\mathscr{C}^3$ differentiability of the functions h and $\widehat{g}$ in (1.10) can be weakened to $\mathscr{C}^2$ differentiability and a Lipschitz condition for the Hessian H of the Lagrangian at point $\bar{x}$.

Lagrange Function & Optimality Constraints

The Lagrangian is defined as:

$$\mathcal{L}(x, \widetilde{y}, y) = c^T x + \widetilde{y}^T h(x) + y^T \widehat{g}(x). \tag{1.11}$$

Its first derivative is:

$$\nabla_x \mathcal{L}(x, \widetilde{y}, y) = c + \nabla_x h(x)\widetilde{y} + \nabla_x \widehat{g}(x) y. \tag{1.12}$$

With $z = (x, \widetilde{y}, y)$ we can also write $\nabla_x \mathcal{L}(x, \widetilde{y}, y) = \nabla_x \mathcal{L}(z)$. The Hessian of the Lagrangian is:

$$\nabla_x^2 \mathcal{L}(z) = H(z) = \nabla_x^2 h(x)\widetilde{y} + \nabla_x \widehat{g}(x) y. \tag{1.13}$$

It is assumed that Robinson CQ holds at $\bar{x}$. Then, if $\bar{x}$ is a local minimizer of P_{gen} (1.10) the first order optimality conditions must hold. This means, that there exist vectors $\widetilde{y} \in \mathbb{R}^{m_1}, y \in \mathbb{R}^{m_2}$ with:

$$\begin{aligned} h(\bar{x}) &\leq 0, \\ \widehat{g}(\bar{x}) &= 0, \\ \nabla_x \mathcal{L} &= 0, \\ \widetilde{y} &\geq 0. \end{aligned} \tag{1.14}$$

For a non-optimal Karush-Kuhn-Tucker (KKT) tuple $z^k = (x^k, \widetilde{y}^k, y^k)$ the first order optimality condition does not hold in general, as this point is not necessarily an optimal point with an CQ. In order to find a new iterate z^{k+1} with $x^{k+1} = x^k + \Delta x$, which is closer to $\bar{x}$, the following linearization is used to calculate corrections $\Delta z = (\Delta x, \Delta\widetilde{y}, \Delta y)$:

$$\begin{aligned} h(x^k) + \nabla_x h(x^k)\Delta x &\leq 0, \\ \widehat{g}(x^k) + \nabla_x \widehat{g}(x^k)\Delta x &= 0, \\ c + H_k \Delta x + \nabla_x h(x^k)(\widetilde{y}^k + \Delta\widetilde{y}) + \nabla_x \widehat{g}(x^k)(y^k + \Delta y) &= 0, \\ \widetilde{y} + \Delta\widetilde{y} &\geq 0. \end{aligned} \tag{1.15}$$

The matrix H_k is the evaluation of the Hessian of the Lagrangian at z^k. The fourth equation in 1.15 is the following approximation:

$$\begin{aligned}\nabla_x \mathcal{L}(z^k + \Delta z) &= c + \nabla_x h(x^k + \Delta x)(\widetilde{y}^k + \Delta\widetilde{y}) + \nabla_x \widehat{g}(x^k + \Delta x)(y^k + \Delta y) \\ &\widetilde{=} c + H_k \Delta x + \nabla_x h(x^k)(\widetilde{y}^k + \Delta\widetilde{y}) + \nabla_x \widehat{g}(x^k)(y^k + \Delta y).\end{aligned} \tag{1.16}$$

The equations in 1.15 are linearized first order optimality conditions. But can also be interpreted as the first order optimality conditions of an other problem $P_{\text{sub}}(\Delta x; x^k)$.

Subproblem and Algorithm

The subproblem $P_{\text{sub}}(\Delta x; x^k)$ is defined as follows:

$$\begin{aligned}\min_{\Delta x \in \mathbb{R}^n} \quad & c^T \Delta x + \frac{1}{2} \Delta x^T H_k \Delta x \\ \text{s.t.} \quad & h(x^k) + \nabla_x h(x^k) \Delta x \leq 0, \\ & \widehat{g}(x^k) + \nabla_x \widehat{g}(x^k) \Delta x = 0.\end{aligned} \tag{1.17}$$

The matrices H_k can be approximated by positive semi-definite matrices $\widetilde{H}_k$ to solve the subproblem more efficiently. Possible ways to generate the approximations are BFGS updates or projection of the Hessian of the Lagrangian, see [23]. Now it is possible to formulate the SCP algorithm, see Algorithm 3. The algorithm terminates if $\Delta z = 0$.

Algorithm 3 SCP algorithm

Require: Postulated assumptions hold, see section 1.2.1.

1: **loop**
2: Solve the subproblem $P_{\text{sub}}(\Delta x; x^k)$ to get Δz.
3: Set $z^{k+1} = z^k + \Delta z$.
4: **end loop**

Remark 1.6. In [23] it is shown, that this algorithm is locally quadratic convergent. In [10] global convergence is shown with a modified version of the SCP algorithm which uses the Armijo line search strategy and the non-differentiable Han merit function.

Contraction

Theorem 1.1. *Assume that the functions h and $\widehat{g}$ in* (1.10) *are $\mathscr{C}^3$ differentiable, and that problem* (1.10) *has a locally unique and strictly complementary solution $\bar{z} = (\bar{x}, \bar{\widetilde{y}}, \bar{y})$ that satisfies the Robinson CQ* (A.2.2) *and the second-order sufficient condition. Let z^k be some given iterate, and let the next iterate*

$$z^{k+1} = z^k + \Delta z \tag{1.18}$$

be defined as the local solution of (1.17) *that is closest to z^k. Then there exists $\epsilon > 0$ and $\gamma < \frac{1}{\epsilon}$ such that*

$$\left\| z^{k+1} - \bar{z} \right\| \leq \gamma \left\| z^k - \bar{z} \right\|^2 \tag{1.19}$$

whenever $\left\| z^k - \bar{z} \right\| < \epsilon$.

Proof. The main idea is to reduce the problem to a quadratic semi-definite program with a linear semi-definitness constraint. These constraints require the semi-definitness of a given matrix valued function. This concept generalizes equality and inequality constraints. With the help of diagonal matrices, where the eigenvalues are given by the diagonal elements, equality and inequality constraints can be converted into semi-definitness constraints. In this thesis, the linearization of equality and inequality constraints are shown explicitly, as the Algorithm 4 is formulated for these constraint types only. Note that in the SCP literature formulation with a semi-definite constraint is more common.
In the second step of the proof, it is shown that x^k is the optimal solution of a subproblem close to $P_{\text{sub}}(\Delta x; x^k)$ (1.17). With perturbation analysis and the sensitivity result mentioned in Remark 1.4, the proposition follows. As this proof is not crucial for the discussion of APCSCP, it is not given here. For the complete proof with all details, see [23, pages 601 ff.]. □

Remark 1.7. As $\gamma < \frac{1}{\epsilon} < \frac{1}{\|z^k - \bar{z}\|}$ holds, we can follow:

$$\|z^{k+1} - \bar{z}\| \leq \gamma \|z^k - \bar{z}\|^2 < \|z^k - \bar{z}\| \tag{1.20}$$

This shows that $z^k \xrightarrow{k\to\infty} \bar{z}$ and that $\|z^{k+1} - \bar{z}\| = \mathcal{O}(\|z^k - \bar{z}\|^2)$ for $k \to \infty$ and therefore the local quadratic convergence. This is literally a contraction theorem, as it is possible to prove an estimate of the form $\|z^{k+1} - \bar{z}\| < \widehat{\gamma}\|z^k - \bar{z}\|$ with $\widehat{\gamma}$ less than 1. In section 1.3 two other contraction theorems are given, but in there holds that the contraction factor $\widehat{\gamma}$ equals 1. This means, that with $k \to \infty$ the distance to $\bar{z}$ is bounded by a constant, but does not necessarily converge.

1.2.2. Predictor-Corrector Method

The basic idea of a *predictor-corrector* approach as shown in [13, pages 238 ff.] is the following: In the *prediction* step is, based on the old data $(z(\xi_k), \xi_k)$, predicted how the problem will change with the new parameter ξ_{k+1}. As in general z^k is only an approximation to the exact solution $\bar{z}^k$, Newton's method is used in the *correction* step to reduce the approximation error. These two steps combined lead to z^{k+1}, an approximation of the new optimal point $\bar{z}^{k+1}$. Note that this method can be applied to solve the subproblems $P_T(\xi_k)$ (1.9) of real-time Algorithm 2. In contrast to [19], correction and prediction are not done at the exact solution $\bar{z}^k$ but at the approximate z^k, as the correction step for the exact solution $\bar{z}^k$ will always be 0.
The necessary first order optimality condition says, that in an exact local solution $\bar{z}(\xi_k) = \bar{z}^k$ with constraint qualification the KKT conditions hold [41]. This can be written in the form:

$$\mathscr{F}(\bar{z}(\xi_k), \xi_k) = 0, \tag{1.21}$$

where z is the primal dual variable for problem $P_T(\xi)$. Note that this is possible for problem $P_T(\xi)$ (1.9) with $\Omega = \mathbb{R}^n$. The solution satisfying the KKT condition $\bar{z}(\xi_k)$ is a continuously differentiable map. Therefore the implicit function theorem is applicable and if z^k is close enough to $\bar{z}^k$, the following equation follows:

$$\frac{\partial}{\partial \xi}\mathscr{F}(z(\xi_k); \xi_k) + \frac{\partial}{\partial z}\mathscr{F}(z(\xi_k); \xi_k)\frac{\partial}{\partial \xi} z(\xi_k) = 0. \tag{1.22}$$

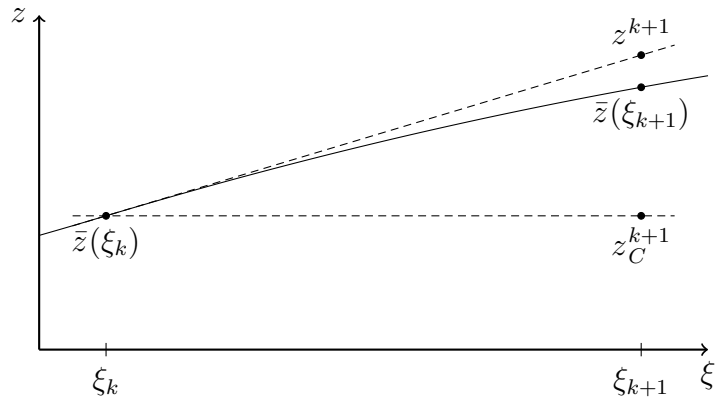


Figure 1.6.: Visualization of classic and tangential continuation for $z = \bar{z}$. The classic prediction is denoted as z_C^{k+1} and the tangential as z^{k+1}.

This can be reformulated to:

$$\frac{\partial}{\partial \xi} z(\xi_k) = -\left[\frac{\partial}{\partial z}\mathscr{F}(z(\xi_k);\xi_k)\right]^{-1} \frac{\partial}{\partial \xi}\mathscr{F}(z(\xi_k);\xi_k). \tag{1.23}$$

Predictor

The map $\bar{z}(\xi)$ is continuously differentiable and ξ_k is close to ξ_{k+1}. Therefore, the old data z^k can be used to predict the result of $\bar{z}(\xi_{k+1})$. See [13, page 238].

Classic Continuation The easiest approach is the *classic continuation*:

$$z_C^{k+1} = \bar{z}(\xi_k). \tag{1.24}$$

The approximation z^k is used as a guess for the new one. See figure 1.6.

Tangential Continuation A more sophisticated way to predict the outcome for ξ_{k+1} is the *tangential continuation*. See [13, page 239]:

$$z^{k+1} = \bar{z}(\xi_k) + \frac{\partial}{\partial \xi} z(\xi_k)(\xi_{k+1} - \xi_k). \tag{1.25}$$

Here the derivative of z is used to improve the estimate. The product in the second summand is meant element-wise. This is visualized in figure 1.6. Inserting equation (1.23) leads to:

$$z^{k+1} = \bar{z}(\xi_k) \underbrace{-\left[\frac{\partial}{\partial z}\mathscr{F}(z^k;\xi_k)\right]^{-1} \frac{\partial}{\partial \xi}\mathscr{F}(z^k;\xi_k)(\xi_{k+1} - \xi_k)}_{\Delta z_{\text{predictor}}}. \tag{1.26}$$

Remark 1.8. Note that these continuation formulas correspond to the Taylor approximation of order 0 (constant) and 1 (tangential), when $z = \bar{z}$. Otherwise the approximation error in $z(\xi)$ is already corrected. Continuations based on higher order Taylor approximations are possible, but then also the corresponding derivatives of the unknown map $z(\xi)$ are necessary.

Corrector

In general, the exact solution $\bar{z}^k$ is found by correcting the prediction error with Newton's method [13] with initial value z^k. Due to the limited time in the real-time iteration at every time step k only one Newton step is performed[2] and the exact solution $\bar{z}^k$ is approximated with $\tilde{z}^k$:

$$\begin{aligned} \tilde{z}^k &= z^k + \Delta z_{\text{corrector}}, \\ \Delta z_{\text{corrector}} &= -\left[\frac{\partial}{\partial z}\mathscr{F}(z^k;\xi_k)\right]^{-1} \mathscr{F}(z^k;\xi_k). \end{aligned} \tag{1.27}$$

The combination of Predictor (1.26) and Corrector (1.27) leads to the following:

$$z^{k+1} = z^k + \Delta z_{\text{corrector}} + \Delta z_{\text{predictor}} = \tilde{z}^k + \Delta z_{\text{predictor}} \tilde{=} \bar{z}^k + \Delta z_{\text{predictor}}. \tag{1.28}$$

this can also be written as:

$$\mathscr{F}(z^k;\xi_k) + \frac{\partial}{\partial \xi}\mathscr{F}(z^k;\xi_k)(\xi_{k+1} - \xi_k) + \frac{\partial}{\partial z}\mathscr{F}(z^k;\xi_k)(z^{k+1} - z^k) = 0. \tag{1.29}$$

Linearity If $\mathscr{F}(z;\xi)$ is linear in ξ, it holds:

$$\frac{\partial}{\partial \xi}\mathscr{F}(z^k;\xi)(\xi_k - \xi_{k+1}) = \mathscr{F}(z^k;\xi_k) - \mathscr{F}(z^k;\xi_{k+1}). \tag{1.30}$$

Then equation (1.29) can be simplified to:

$$F(z^k;\xi_{k+1}) + \frac{\partial}{\partial z}F(z^k;\xi_k)(z^{k+1} - z^k) = 0. \tag{1.31}$$

Note that the KKT condition of problem $P_T(\xi)$ (1.9) is linear in ξ, because ξ is linear in the equality constraint, see section 1.1.3.

Remark 1.9. In the classic predictor-corrector method, the predictor step is applied to find a good initial value for the Newton method in the correction step. If the Newton method converges, the result in the kth step is the exact solution $\bar{z}(\xi_k)$. The initial value z^k is a temporary value within the PC method.
In the real-time setting, only the first step of the Newton iteration is performed. Moreover, the iteration is shifted by half an iteration cycle. Firstly, a Newton step correction is done to obtain a better approximation $\tilde{z}^k$. The result of the correction step is now just a temporary value. Secondly, the prediction is calculated at the approximate solution z and added to the temporary value obtained by the correction step. This is visualized in figures 1.7 and 1.8. This approach has the advantage that the calculation is faster and deterministic, as the corrector always performs only one Newton step. It has the disadvantages that the exact solution is not necessarily found and it must be ensured that z^k is sufficiently close to $\bar{z}^k$. These issues will also be subject of the proofs in section 1.3.

Remark 1.10. The inverse matrices in equations (1.26) and (1.27) are identical. This allows the simplification to equation (1.31) and fits to the generalized equation in equation (1.45). The author's efforts to derive a new algorithm by performing the correction step at z^{k+1} were not successful, as this leads to two different inverses and it was not possible to identify the obtained predictor-corrector formula with KKT conditions of a new subproblem.

[2] A similar approach can be seen in [14].

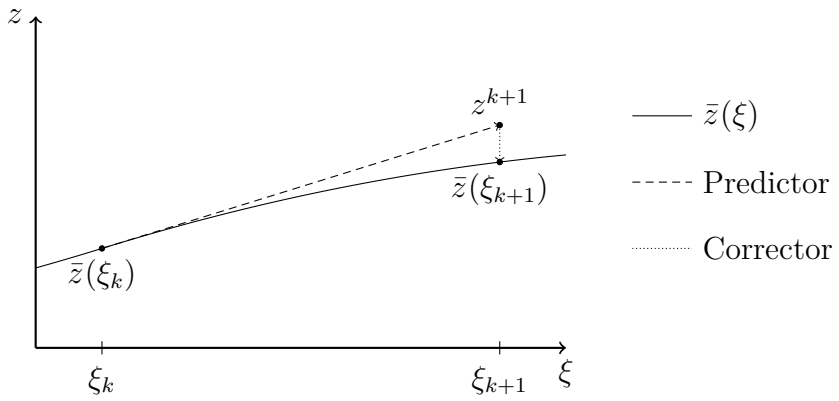


Figure 1.7.: Visualization of the classic predictor corrector approach with tangential continuation and a successfully converging Newton's method as corrector.

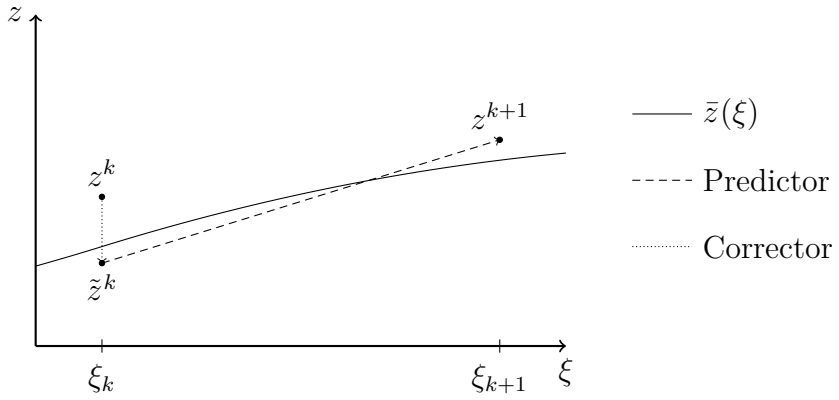


Figure 1.8.: Visualization of the adapted predictor corrector approach with tangential continuation and an approximative corrector. Note that the prediction is calculated at z^k.

1.2.3. Adjoint-based Optimization

In receding horizon control problems, a large computational effort is the evaluation of goal and constraint functions and their derivatives using an iterative nonlinear problem solver like `IPOpt`. A significant amount of time can be saved if a method can deal with approximate but fast calculable derivative information . The adjoint based methods in [36] and [24] require an exact Lagrange gradient, but cope with an approximate Jacobian. These methods still converge to the exact solution but in less time. One of the main reasons for the better performance of Algorithm 4 compared to `IPOpt` is the faster function evaluation time.

1.3. The Adjoint-based Predictor Corrector SCP Algorithm

In this section, the *Adjoint-based Predictor-Corrector SCP* algorithm (APCSCP) [19] is introduced and discussed. This algorithm extends the work in [18] and combines the ideas shown in section 1.2 in order to approximately solve the discretized receding horizon control problem $P_T(\xi)$ (1.9) faster.
In this section, we go a similar way as in section 1.2.1 but in the opposite direction. In the beginning, the subproblem, the algorithm, and necessary assumptions are formulated and then the KKT conditions of the original problem and the subproblem are discussed. The concept of *strong regularity* is introduced, as it is needed to show the upcoming contraction theorems. The proof is a corrected and more detailed version of the proof given in [19]. The consequences of the contraction theorems are discussed and visualized. In section 1.3.6, an example is given and the obtained algorithms are compared to the SQP method.

1.3.1. Algorithm

Instead of solving the original problem $P_T(\xi)$ (1.9), a convex subproblem $P(z^k, A_k, \widetilde{H}_k; \xi)$ of the following form is solved to approximate $\bar{z}^{k+1}(\xi_{k+1})$:

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & f(x) + (m^k)^T(x - x^k) + \frac{1}{2}(x - x^k)^T \widetilde{H}_k (x - x^k) \\ \text{s.t.} \quad & A_k(x - x^k) + g(x^k) + M\xi = 0, \\ & x \in \Omega, \end{aligned} \tag{1.32}$$

with $A_k \in \mathbb{R}^{n_{eqC} \times n_{opt}}$ approximating $g'(x^k)$, $\widetilde{H}_k \in \mathcal{S}_+^{n_{opt}}$ approximating or regularizing the second derivative of the Lagrangian and $m^k = m(z^k, A) = (g'(x^k) - A_k)^T y^k$ correcting the inconsistency between $g'(x^k)$ and the approximation A_k. It is assumed that Assumptions 1 to 3 hold and that the original problem has a KKT point $\bar{z}^k$ and is strongly regular (Definition 1) in this point. Moreover, the subproblem $P(z^k, A_k, \widetilde{H}_k; \xi)$ is solvable and Slater's condition (see appendix A.2.1) holds.

The APCSCP real-time algorithm is given in Algorithm 4. The result is visualized in figure 1.9. Note that the structure of the subproblem has a similar structure to the

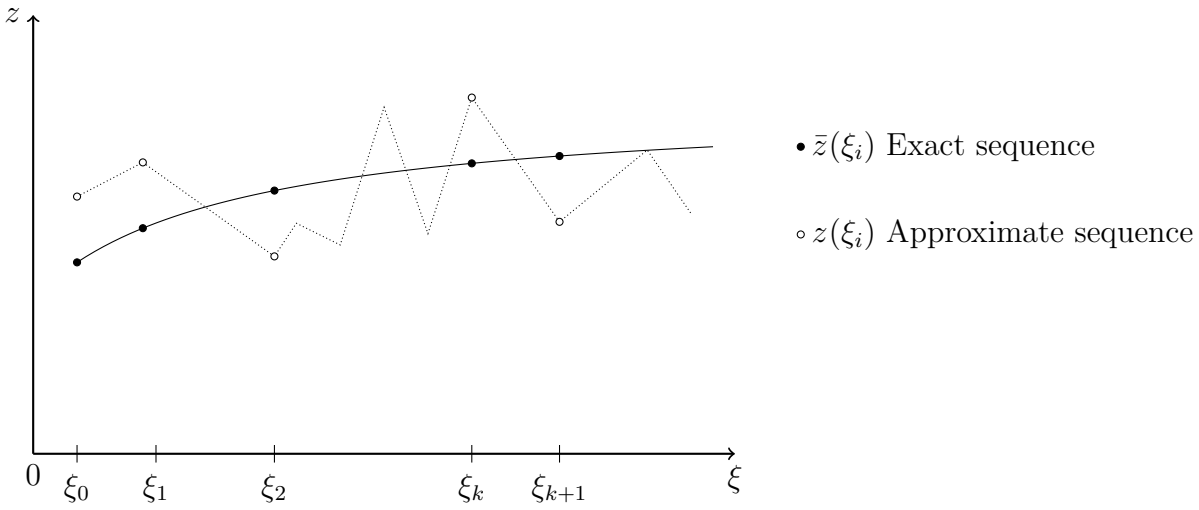


Figure 1.9.: Visualization of the obtained approximated KKT sequence $\{z(\xi_k)\}_{k\geq 0}$ obtained by Algorithm 4. The horizontal axis might induce, that the parameters ξ are ordered. This simplifies the visualization, but does not hold in general.

subproblem obtained in the classic SCP approach, see $P_{\text{sub}}(\Delta x; x^k)$ (1.17). The connection to the predictor-corrector approach and to adjoint methods is not obvious. In the following subsection, a closer look at the KKT conditions of the subproblem will show these connections.

Remark 1.11. The calculation of m^k does not require the calculation of the whole Jacobian $g'(x^k)$. Only the product $g'(x^k)^T y^k$ is of interest. Most automated differentiation libraries (e.g. `CasADi`, see section 2.2.1), can calculate the result of this product in the same cost order as the evaluation of g. This is also the reason why Algorithm 4 is faster than the later presented PCSCP algorithm, where $m = 0$ at all times, but $g'(x^k)$ must be fully calculated to update $A_k = g'(x^k)$.

Algorithm 4 Adjoint-based predictor-corrector SCP (APCSCP) Algorithm [19]

Require: For a given parameter $\xi_0 \in \mathcal{P}$ solve offline $P_T(\xi)$ (1.9) to get an approximate KKT point $z^0 = (x^0, y^0)$. Compute $g(x^0)$, find a matrix A_0 approximating $g'(x)$ and $\widetilde{H}_0 \in \mathcal{S}^n_+$

1: **loop** $k \in \{0, 1, \dots\}$
2: Get a new parameter value $\xi_{k+1} \in \mathcal{P}$.
3: Solve $P(z^k, A_k, \widetilde{H}_k; \xi_{k+1})$ to obtain a solution for $z^{k+1} = (x^{k+1}, y^{k+1})$.
4: Evaluate $g(x^{k+1})$, update $A_{k+1}, \widetilde{H}_{k+1}, m^{k+1} = g'(x^{k+1})^T y^{k+1} - A_{k+1}^T y^{k+1}$.
5: **end loop**

Assumptions

To show the contraction estimates, the following assumptions are required.

Assumption 1. *The function g is twice differentiable on its domain.*

If this assumption does not hold, the Hessian of the Lagrangian $\mathcal{L}$ does not exist.

Assumption 2. *For a given $\xi \in \mathcal{P}$, the problem $P_T(\xi)$ has at least one KKT-point $\bar{z}$, i.e. the set of all KKT-points $Z^*(\xi)$ is not empty.*

This assumption is necessary as in general we can not assume the existence of a KKT point and therefore definitions in section 1.3.2 are not well-defined without this assumption. The approximated Jacobian $\widetilde{F}'_k$ (1.44) has the following properties:

Assumption 3. *For a given $\bar{z}^k \in Z^*(\xi_k),\ k \leq 0$ holds:*

a) There exists a constant $0 \leq \kappa < \frac{1}{2\gamma}$ such that:

$$\|F'(\bar{z}^k) - \widetilde{F}'_k\| \leq \kappa. \tag{1.33}$$

b) The Jacobian mapping $F'(\cdot)$ is Lipschitz continuous with Lipschitz constant $0 \leq \omega < \infty$ on $\mathcal{B}\left(\bar{z}^k, r_z\right)$ with $r_z > 0$.

Remark 1.12. Note that, Assumption 3.a can be written as:

$$\left\|F'(\bar{z}^k) - \widetilde{F}'_k\right\| = \left\| \begin{pmatrix} \nabla^2_x \mathcal{L}(\bar{z}^k) - \widetilde{H}_k & g'(\bar{x}^k)^T - A_k^T \\ g'(\bar{x}^k) - A_k & 0 \end{pmatrix} \right\|. \tag{1.34}$$

So this assumption ensures that the approximation is sufficiently good. An important point is that the matrix $\nabla^2_x \mathcal{L}(\bar{z}^k)$ does not have to be positive semi-definite, but its approximation $\widetilde{H}_k$ has to. This kind of assumption appears also in Newton-type methods theory to ensure that the residual is sufficiently small in a neighborhood of the exact solution, see [13, 15].

1.3.2. KKT Conditions

As f is assumed to be convex, $P_T(\xi)$ (1.9) can be written as:

$$\begin{aligned} &\min_{s \in \mathbb{R}} s \\ &\text{s.t.} g(x) + M\xi = 0, \\ &\quad x \in \Omega, \\ &\quad f(x) \leq s \end{aligned} \tag{1.35}$$

Therefore the goal function f is in the following without loss of generality linear and has the form $f(x) = c^T x$.

Original Problem

The Lagrangian $\mathcal{L}$ of problem $P_T(\xi)$ is

$$\mathcal{L}(z; \xi) = c^T x + (g(x) + M\xi)^T y. \tag{1.36}$$

The second constraint ($x \in \Omega$) is implicitly given and not represented in $\mathcal{L}$. But it occurs in the Karush-Kuhn-Tucker (KKT) conditions for problem $P_T(\xi)$:

$$\begin{aligned} 0 &\in c + g'(x)^T y + \mathcal{N}_\Omega(x), \\ 0 &= g(x) + M\xi, \end{aligned} \tag{1.37}$$

where $\mathcal{N}_\Omega(x)$ (A.4) is the normal cone of the feasible set at point x. The first condition of the equation above ensures the stationarity and the feasibility of $x \in \Omega$, see Remark A.1. The second condition ensures the feasibility of the equality condition $g(x) + M\xi = 0$. A point z where (1.37) holds is called *KKT-point* of $P_T(\xi)$. x is called *stationary point* of problem $P_T(\xi)$ and y is the corresponding Lagrange multiplier. $Z^*(\xi)$ is defined as the set of all KKT points of problem $P_T(\xi)$ and $X^*(\xi)$ as the set of all stationary points of problem $P_T(\xi)$.

For F defined as

$$F(z) = \begin{pmatrix} c + g'(x)^T y \\ g(x) \end{pmatrix}, \tag{1.38}$$

the KKT conditions (1.37) are a parametric generalized equation of the following form:

$$0 \in F(z) + C\xi + \mathcal{N}_K(z) \tag{1.39}$$

for $C = \begin{pmatrix} 0 \\ M \end{pmatrix}$ and $K = \Omega \times \mathbb{R}^{n_{eqC}}$. This holds as $\mathcal{N}_{\mathbb{R}^{n_{eqC}}} = \{0\}$, see appendix A.1.1. For a KKT point $\bar{z} \in Z^*(\xi)$ it is possible to define:

$$L(z; \bar{z}, \xi) = F(\bar{z}) + F'(\bar{z})(z - \bar{z}) + C\xi + \mathcal{N}_K(z). \tag{1.40}$$

For $\bar{z} \in Z^*(\xi)$ the KKT conditions hold by definition and therefore $0 \in L(\bar{z}; \bar{z}, \xi)$. L can be interpreted as the linearization of equation (1.39) in the KKT point $\bar{z}$. The corresponding inverse map is:

$$L^{-1}(\mathfrak{z}; \bar{z}, \xi) = \{z \in \mathbb{R}^{n_z} : \mathfrak{z} \in L(z; \bar{z}, \xi)\}. \tag{1.41}$$

These mappings will be used in section 1.3.3, note that Assumption 2 ensures that $Z^*(\xi)$ is not empty and therefore, that L and L^{-1} are defined.

Subproblem

The KKT conditions of $P(z^k, A_k, \widetilde{H}_k; \xi)$ for given matrices $A_k, \widetilde{H}_k \in \mathcal{S}^n_+$, $z^k \in \mathcal{B}\left(\bar{z}^k, r_z\right), \xi \in \mathcal{B}\left(\xi_k, r_\xi\right)$, and given radii $r_z, r_\xi > 0$ have the form:

$$\begin{aligned} 0 &\in c + m(z^k, A_k) + \widetilde{H}_k(x - x^k) + A_k^T y + \mathcal{N}_\Omega(x), \\ 0 &= g(x^k) + A_k(x - x^k) + M\xi. \end{aligned} \tag{1.42}$$

Furthermore, Slater's condition (A.2.1) holds for every subproblem $P(z^k, A_k, \widetilde{H}_k; \xi)$. As Ω is assumed to be convex and a CQ holds, $\bar{z}$ is a KKT point of $P(z^k, A_k, \widetilde{H}_k; \xi)$ iff $\bar{x}$ solves $P(z^k, A_k, \widetilde{H}_k; \xi)$ with the Lagrange multiplier $\bar{y}$.

As $m(z^k, A_k)$ was defined as $(g'(x^k) - A_k)^T y^k$, the KKT condition (1.42) can be reformulated to:

$$0 \in \begin{pmatrix} c + g'(x^k)^T y^k \\ g(x^k) \end{pmatrix} + \begin{pmatrix} \widetilde{H}_k(x - x^k) + A_k(y - y^k) \\ A_k(x - x^k) \end{pmatrix} + \begin{pmatrix} 0 \\ M\xi \end{pmatrix} + \begin{pmatrix} \mathcal{N}_\Omega \\ 0 \end{pmatrix}. \tag{1.43}$$

Note that this equation corresponds to (1.39), which describes the KKT conditions of the original problem. With

$$\widetilde{F}'_k = \begin{pmatrix} \widetilde{H}_k & A_k^T \\ A_k & 0 \end{pmatrix}, \tag{1.44}$$

$\widetilde{H} \in \mathcal{S}_+^{n_{opt}}$, and the function F (1.38), equation (1.43) can be written as a parametric linear generalized equation:

$$0 \in F(z^k) + \widetilde{F}'_k(z - z^k) + C\xi + \mathcal{N}_K(z). \tag{1.45}$$

Remark 1.13. As w.l.o.g. $f(x) = c^T x$ is linear and Assumption 1 holds, the Hessian of the Lagrangian $\mathcal{L}$ (1.36) of the original problem $P_T(\xi_k)$ exists and has the form

$$\nabla_x^2 \mathcal{L}(z) = \sum_{i=1}^{n_{eqC}} y_i \nabla_x^2 g_i(x). \tag{1.46}$$

The Jacobian of $F(z^k)$ is:

$$F'(z^k) = \begin{pmatrix} \nabla_x^2 \mathcal{L}(z^k) & g'(x^k)^T \\ g'(x^k) & 0 \end{pmatrix}. \tag{1.47}$$

For $A_k = g'(x^k)$ and $\widetilde{H}_k = \nabla_x^2 \mathcal{L}(z^k)$ holds $\widetilde{F}'_k = F'(z^k)$. Equation (1.45) is the linearization of the KKT conditions of the original problem $P_T(\xi)$ (1.39) at point z^k for parameter ξ. Otherwise, $\widetilde{F}'_k$ is just an approximation, see section 1.2.3.
Moreover, equation (1.45) is a generalization of the predictor-corrector approach, see equation (1.31). This means, that solving the subproblem $P(z^k, A_k, \widetilde{H}_k; \xi_{k+1})$ corresponds to one step of the predictor-corrector approach as shown in figure 1.8.

1.3.3. Strong Regularity

The concept of strong regularity [33] has an important role in the proofs of the upcoming contraction estimates.

Definition 1. *Let $\xi_k \in \mathcal{P}$ such that the set of KKT points $Z^*(\xi_k)$ of $P_T(\xi_k)$ is nonempty. Let $\bar{z}^k \in Z^*(\xi_k)$ be a given KKT point of $P_T(\xi_k)$. Problem $P_T(\xi_k)$ is said to be strongly regular at $\bar{z}^k$ if there exist neighborhoods $\mathcal{B}(0, \bar{r}_z)$ and $\mathcal{B}\left(\bar{z}^k, \bar{r}_z\right)$ such that the mapping $z_k^*(\mathfrak{z}) = \mathcal{B}\left(\bar{z}^k, \bar{r}_z\right) \cap L^{-1}(\mathfrak{z}; \bar{z}^k, \xi_k)$ is single valued and Lipschitz continuous in $\mathcal{B}(0, \bar{r}_\mathfrak{z})$ with Lipschitz constant $0 < \gamma < \infty$, i.e.*

$$\|z_k^*(\mathfrak{z}) - z_k^*(\mathfrak{z}')\| \leq \gamma \|\mathfrak{z} - \mathfrak{z}'\|, \ \mathfrak{z}, \mathfrak{z}' \in \mathcal{B}(0, \bar{r}_\mathfrak{z}). \tag{1.48}$$

Note that the constants $\gamma, \bar{r}_z$, and $\bar{r}_\mathfrak{z}$ are global and independent of the individual iteration step k in Algorithm 4. From the definitions of L^{-1}, L, and strong regularity follows the existence of a unique $z_k^*(\mathfrak{z})$ such that:

$$\mathfrak{z} \in L(z_k^*(\mathfrak{z}); \bar{z}^k, \xi_k) = F(\bar{z}^k) + F'(\bar{z}^k)(z_k^*(\mathfrak{z}) - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z_k^*(\mathfrak{z})). \tag{1.49}$$

and $\|\bar{z}^k - z_k^*\| < \bar{r}_\mathfrak{z}$. This can be reformulated to:

$$z_k^*(\mathfrak{z}) = \underbrace{\left(F'(\bar{z}^k) + \mathcal{N}_K\right)^{-1}}_{=J_k} \left(\underbrace{F'(\bar{z}^k)\bar{z}^k - F(\bar{z}^k) - C\xi_k}_{=v^k} + \mathfrak{z} \right) \tag{1.50}$$

Note that if J_k is single-valued and Lipschitz continuous around $v^k = F'(\bar{z}^k)\bar{z}^k - F(\bar{z}^k) - C\xi_k$, strong regularity of $P_T(\xi_k)$ at $\bar{z}^k$ must hold. This characterization of strong regularity will be used in upcoming lemmata and theorems.

Lemma 1.2. *Suppose that Assumptions 1 and 2 are satisfied. Suppose further that problem $P_T(\xi_k)$ is strongly regular at $\bar{z}^k$ for a given $\bar{z}^k \in Z^*(\xi_k)$. Then there exist neighborhoods $\mathcal{B}(\xi_k, r_\xi)$ and $\mathcal{B}\left(\bar{z}^k, r_z\right)$ such that $Z^*(\xi_{k+1})$ is nonempty for all $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$ and $Z^*(\xi_{k+1}) \cap \mathcal{B}\left(\bar{z}^k, r_z\right)$ contains only one point $\bar{z}^{k+1}$. Moreover, there exists a constant $0 < \sigma < \infty$ such that:*

$$\|\bar{z}^{k+1} - \bar{z}^k\| \leq \sigma \|\xi_{k+1} - \xi_k\| \tag{1.51}$$

Proof. See [19, pages 9-10]. □

Remark 1.14. This lemma states that if Algorithm 4 in step k has an approximation z^k close enough to the exact solution $\bar{z}^k$ and the measured change is small enough, the subproblem in the next step $k+1$ has exactly one KKT point in the neighborhood of $\bar{z}^k$. Furthermore, the distance betweene two consecutive optimal solutions is related to the change of the environment. Note that $\|\xi_{k+1} - \xi_k\|$ is assumed to be small.

The corresponding mapping to J_k for $\widetilde{F}'$ is:

$$\widetilde{J}_k = \left(\widetilde{F}_k' + \mathcal{N}_K\right)^{-1}. \tag{1.52}$$

It is possible to formulate and proof Lemma with a similar proposition as Lemma 1.2 but for $\widetilde{J}$.

Lemma 1.3. *Suppose that Assumptions 1 and 2, and Assumption 3.a are satisfied. Then there exist neighborhoods $\mathcal{B}(\xi_k, r_\xi)$ and $\mathcal{B}\left(\bar{z}^k, r_z\right)$ such that for any $z^k \in \mathcal{B}\left(\bar{z}^k, r_z\right)$ and $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$ the the mapping $\widetilde{J}_k$ is single valued in a neighborhood $\mathcal{B}\left(\tilde{v}^k, r_v\right)$, where $\tilde{v}^k = \widetilde{F}_k'\bar{z}^k - F(\bar{z}^k) - C\xi_k$. Moreover $\widetilde{J}_k(\cdot)$ is Lipschitz continuous on $\mathcal{B}\left(\tilde{v}^k, r_v\right)$ with Lipschitz constant $\beta = \frac{\gamma}{1-\gamma\kappa} > 0$.*

Proof. This proof is an extended and more detailed version of the proof shown in [19, pages 10-12] and consists of two parts. Firstly, a contradiction proofs that $\widetilde{J}_k$ is single-valued. Secondly, the Lipschitz continuity is shown.

Assume that for $v \in \mathcal{B}\left(\tilde{v}^k, r_v\right)$ exists z and $z' \neq z$ with $z, z' \in \widetilde{J}(v)$. From the definition of $\widetilde{J}$ (1.52) follows

$$\begin{aligned} v &\in \widetilde{F}_k' z + \mathcal{N}_K(z), \\ v &\in \widetilde{F}_k' z' + \mathcal{N}_K(z'). \end{aligned} \tag{1.53}$$

It is possible to define $\mathfrak{z}$ and $\mathfrak{z}'$ such that

$$\begin{aligned} \mathfrak{z} &= v - \overbrace{\left[\widetilde{F}_k' \bar{z}^k - F(\bar{z}^k) - C(\xi_k)\right]}^{=\tilde{v}^k} + \left[F'(\bar{z}^k) - \widetilde{F}_k'\right](z - \bar{z}^k), \\ \mathfrak{z}' &= v - \underbrace{\left[\widetilde{F}_k' \bar{z}^k - F(\bar{z}^k) - C(\xi_k)\right]}_{=\tilde{v}^k} + \left[F'(\bar{z}^k) - \widetilde{F}_k'\right](z' - \bar{z}^k). \end{aligned} \tag{1.54}$$

Plugging this into equation (1.53) and rearranging it leads to:

$$\begin{aligned} \mathfrak{z} &\in \widetilde{F}_k' z + \mathcal{N}_K(z) - \left[\widetilde{F}_k' \bar{z}^k - F(\bar{z}^k) - C(\xi_k)\right] + \left[F'(\bar{z}^k) - \widetilde{F}_k'\right](z - \bar{z}^k), \\ \mathfrak{z}' &\in \widetilde{F}_k' z' + \mathcal{N}_K(z') - \left[\widetilde{F}_k' \bar{z}^k - F(\bar{z}^k) - C(\xi_k)\right] + \left[F'(\bar{z}^k) - \widetilde{F}_k'\right](z' - \bar{z}^k). \end{aligned} \tag{1.55}$$

Some terms cancel each other out and (1.55) can be simplified to:

$$\begin{aligned} \mathfrak{z} &\in F(\bar{z}^k) + F'(\bar{z}^k)(z - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z), \\ \mathfrak{z}' &\in F(\bar{z}^k) + F'(\bar{z}^k)(z' - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z'). \end{aligned} \tag{1.56}$$

Furthermore, definition (1.54), $v \in \mathcal{B}\left(\tilde{v}^k, r_v\right)$ and Assumption 3.a can be used to estimate the norms of $\mathfrak{z}$ and $\mathfrak{z}'$:

$$\begin{aligned} \|\mathfrak{z}\| &\leq \left\|v - \tilde{v}^k\right\| + \left\|\left[F'(\bar{z}^k) - \widetilde{F}_k'\right](z - \bar{z}^k)\right\| && (1.57) \\ &\leq r_v + \left\|F'(\bar{z}^k) - \widetilde{F}_k'\right\| \left\|z - \bar{z}^k\right\| && (1.58) \\ &\leq r_v + \kappa \left\|z - \bar{z}^k\right\|. && (1.59) \end{aligned}$$

The constants r_v and r_z can be shrunk sufficiently small such that $\|\mathfrak{z}\| \leq \bar{r}_{\mathfrak{z}}$ and $\|\mathfrak{z}'\| \leq \bar{r}_{\mathfrak{z}}$ holds. Then the strong regularity assumption is applicable. Equation (1.56) and the strong regularity assumption, the definition of $\mathfrak{z}, \mathfrak{z}'$ (1.54) and Assumption 3.a give the following estimates:

$$\begin{aligned} \|z - z'\| &\leq \gamma \|\mathfrak{z} - \mathfrak{z}'\| && (1.60) \\ &\leq \gamma \left\|F'(\bar{z}^k) - \widetilde{F}_k'\right\| \|z - z'\| && (1.61) \\ &\leq \gamma\kappa \|z - z'\|. && (1.62) \end{aligned}$$

Assumption 3.a provides the following estimate:

$$\gamma\kappa \leq \frac{1}{2} < 1. \tag{1.63}$$

Equations (1.62) and (1.63) contradict the assumption $z \neq z'$ and therefore the single-valuedness of $\widetilde{J}(\cdot)$ is proven. The Lipschitz continuity is proven directly and the ansatz

is quite similar to the first part. Let now $v, v' \in \mathcal{B}\left(\tilde{v}^k, r_v\right)$, $z = \widetilde{J}_k(v)$ and $z' = \widetilde{J}_k(v')$. Similar to (1.54) we define:

$$\begin{aligned} \eta &= v - \left[\widetilde{F}'_k - F(\bar{z}^k) - C\xi_k\right] + \left[F'(\bar{z}^k) - \widetilde{F}'_k\right](z - \bar{z}^k), \\ \eta' &= v' - \left[\widetilde{F}'_k - F(\bar{z}^k) - C\xi_k\right] + \left[F'(\bar{z}^k) - \widetilde{F}'_k\right](z' - \bar{z}^k). \end{aligned} \tag{1.64}$$

Similar to equations (1.53) and (1.56), this leads to:

$$\begin{aligned} \eta &\in F(\bar{z}^k) + F'(\bar{z}^k)(z - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z), \\ \eta' &\in F(\bar{z}^k) + F'(\bar{z}^k)(z' - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z'). \end{aligned} \tag{1.65}$$

Now, equation (1.65) and strong regularity, equation (1.64) and Assumption 3.a are applied in this order to get:

$$\begin{aligned} \|z - z'\| &\leq \gamma \|\eta - \eta'\| && (1.66) \\ &\leq \gamma \left(\|v - v'\| + \left\|\left[F'(\bar{z}^k) - \widetilde{F}'_k\right](z - z')\right\|\right) && (1.67) \\ &\leq \gamma \left(\|v - v'\| + \kappa \|z - z'\|\right). && (1.68) \end{aligned}$$

With equation (1.63) this leads to:

$$\|z - z'\| \leq \underbrace{\frac{\gamma}{1 - \gamma\kappa}}_{=\beta>0} \|v - v'\| \tag{1.69}$$

and shows the Lipschitz continuity with constant γ. □

Remark 1.15. It was mentioned earlier that for a KKT point z^{k+1} of the problem $P(z^k, A_k, \widetilde{H}_k, \xi_{k+1})$ (1.32) equation (1.45) holds by definition. This equation can be reformulated in the following way:

$$\begin{aligned} 0 &\in F(z^k) + \widetilde{F}'_k(z^{k+1} - z^k) + C\xi_{k+1} + \mathcal{N}_K(z^{k+1}), \\ \widetilde{F}'_k z^k - F(z^k) - C\xi_{k+1} &\in \left(\widetilde{F}'_k + \mathcal{N}_K\right)(z^{k+1}). \end{aligned} \tag{1.70}$$

If z^{k+1} is in $\mathcal{B}\left(\bar{z}^k, r_z\right)$ with Lemma 1.3 can be concluded that problem $P(z^k, A_k, \widetilde{H}_k, \xi_{k+1})$ (1.32) is uniquely solvable. This can then be expressed with $\widetilde{J}_k$ as:

$$z^{k+1} = \widetilde{J}_k\left(\widetilde{F}'_k z^k - F(z^k) - C\xi_{k+1}\right). \tag{1.71}$$

This KKT point of the k-th subproblem will be the next iterate z^{k+1} in Algorithm 4. Lemma 1.3 gives uniqueness in the neighborhood of z^k. It also guarantees the existence of the next iterate z^{k+1} if the actual iterate z^k is close enough to the exact solution $\bar{z}^k$. It remains to show that the new iterate z^{k+1} is again close enough to the exact solution $\bar{z}^{k+1}$.

Remark 1.16. An equation similar to (1.71) holds for the exact solution $\bar{z}^{k+1}$ and corresponding Lagrange multiplier of the original problem $P_T(\xi_{k+1})$. The KKT conditions holds and can be written as:

$$0 = F(\bar{z}^k) + C\xi_{k+1} + \hat{v}_{k+1}, \hat{v}_{k+1} \in \mathcal{N}_K(\bar{z}^{k+1}). \tag{1.72}$$

This and (1.52) leads to the following:

$$\begin{aligned} \bar{z}^{k+1} &= \widetilde{J}_k\left(\widetilde{F}'_k \bar{z}^{k+1} + v_n\right), \\ \bar{z}^{k+1} &= \widetilde{J}_k\left(\widetilde{F}'_k \bar{z}^{k+1} - F(\bar{z}^{k+1}) - C\xi_{k+1}\right). \end{aligned} \tag{1.73}$$

1.3.4. Contraction Estimate for APCSCP

Now it is possible to formulate the main theorem of this thesis.

Theorem 1.4. *Suppose that Assumptions 1 and 2 are satisfied for some $\xi_0 \in \mathcal{P}$. Then, for $k \geq 0$ and $\bar{z}^k \in Z^*(\xi_k)$, if $P_T(\xi_k)$ is strongly regular at $\bar{z}^k$ then there exist neighborhoods $\mathcal{B}\left(\bar{z}^k, r_z\right)$ and $\mathcal{B}\left(\xi_k, r_\xi\right)$ such that:*

a) The set of KKT points $Z^*(\xi_{k+1})$ of $P_T(\xi_{k+1})$ is nonempty for any $\xi_{k+1} \in \mathcal{B}\left(\xi_k, r_\xi\right)$

b) If, in addition, Assumption 3.a is satisfied then subproblem $P(z^k, A_k, \widetilde{H}_k, \xi_{k+1})$ is uniquely solvable in the neighborhood $\mathcal{B}\left(\bar{z}^k, r_z\right)$.

c) Moreover, if Assumption 3.b holds as well the sequence $\left\{z^k\right\}_{k\geq 0}$ generated by Algorithm 4, where $\xi_{k+1} \in \mathcal{B}\left(\xi_k, r_\xi\right)$, guarantees

$$\begin{aligned}\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq &\left(\alpha + p_1\|z^k - \bar{z}^k\|\right)\|z^k - \bar{z}^k\| \\ &+ \left(p_2 + p_3\left\|\xi_{k+1} - \xi_k\right\|\right)\|\xi_{k+1} - \xi_k\|\end{aligned} \tag{1.74}$$

where $0 \leq \alpha < 1$, $0 \leq p_i < \infty$, $i = 1, 2, 3$ and $p_2 > 0$ are given constants and $\bar{z}^{k+1} \in Z^*(\xi_{k+1})$.

Proof. This theorem is proven by induction. For $k = 0$ Assumption 2 gives us the existence of at least one KKT point and Lemma 1.3 the uniqueness of the solution and with the calculations above equation (1.74) follows. The induction steps for some $k \in \mathbb{N}$ are

a) Lemma 1.2 is applicable and the statement follows from that.

b) When assumption 3.a holds, Lemma 1.3 can be applied and from that the uniqueness follows.

c) With equations (1.71) and (1.73), Lemma 1.3, addition of a tautology[3], and the triangle equation follows:

$$\begin{aligned}\left\|z^{k+1} - \bar{z}^{k+1}\right\| =& \left\|\widetilde{J}_k\left(\widetilde{F}'_k z^k - F(z^k) - C\xi_{k+1}\right) - \widetilde{J}_k\left(\widetilde{F}'_k \bar{z}^{k+1} - F(\bar{z}^{k+1}) - C\xi_{k+1}\right)\right\| \\ \leq& \beta\left\|\widetilde{F}'_k\left(z^k - \bar{z}^{k+1}\right) - F(z^k) + F(\bar{z}^{k+1})\right\| \\ =& \beta\left\|\widetilde{F}'_k\left(z^k - \bar{z}^{k+1}\right) - F(z^k) + F(\bar{z}^{k+1}) + F(\bar{z}^k) - F(\bar{z}^k) + \widetilde{F}'_k \bar{z}^k - \widetilde{F}'_k \bar{z}^k\right\| \\ \leq& \beta\left\|\widetilde{F}'_k\left(z^k - \bar{z}^k\right) - F(z^k) + F(\bar{z}^k) + \left[F'(\bar{z}^k) - F'(\bar{z}^k)\right]\left(z^k - \bar{z}^k\right)\right\| + \\ & \beta\left\|F(\bar{z}^{k+1}) - F(\bar{z}^k) - \widetilde{F}'_k\left(\bar{z}^{k+1} - \bar{z}^k\right) + \left[F'(\bar{z}^k) - F'(\bar{z}^k)\right]\left(z^k - \bar{z}^k\right)\right\|.\end{aligned} \tag{1.75}$$

Now in contrast to [19] the fundamental theorem of calculus can be applied, reordering

[3] in this case adding sophisticated 0

and the inequalities in Assumption 3 lead to:

$$\begin{aligned}\left\|z^{k+1}-\bar{z}^{k+1}\right\| \leq & \beta\left\|\left[\widetilde{F}_k'-F'(\bar{z}^k)\right]\left(z^k-\bar{z}^k\right)-\int_0^1\left[F'\left(\bar{z}^k+t(z^k-\bar{z}^k)\right)-F'(\bar{z}^k)\right](z^k-\bar{z}^k)\mathrm{d}t\right\| \\ & +\beta\left\|\left[\widetilde{F}_k'-F'(\bar{z}^k)\right]\left(\bar{z}^{k+1}-\bar{z}^k\right)-\int_0^1\left[F'\left(\bar{z}^k+t(\bar{z}^{k+1}-\bar{z}^k)\right)-F'(\bar{z}^k)\right]\left(\bar{z}^{k+1}-\bar{z}^k\right)\mathrm{d}t\right\| \\ \leq & \beta\left\|\left(\widetilde{F}_k'-F'(\bar{z}^k)\right)-\int_0^1 F'\left(\bar{z}^k+t(z^k-\bar{z}^k)\right)-F'(\bar{z}^k)\mathrm{d}t\right\|\left\|z^k-\bar{z}^k\right\| \\ & +\beta\left\|\left(\widetilde{F}_k'-F'(\bar{z}^k)\right)-\int_0^1 F'\left(\bar{z}^k+t(\bar{z}^{k+1}-\bar{z}^k)\right)-F'(\bar{z}^k)\mathrm{d}t\right\|\left\|\bar{z}^{k+1}-\bar{z}^k\right\| \\ \leq & \beta\left(\kappa+\tfrac{\omega}{2}\|z^k-\bar{z}^k\|\right)\|\bar{z}^k-\bar{z}^k\|+\beta\left(\kappa+\tfrac{\omega}{2}\left\|\left(\bar{z}^{k+1}-\bar{z}^k\right)\right\|\right)\left\|\left(\bar{z}^{k+1}-\bar{z}^k\right)\right\|.\end{aligned} \tag{1.76}$$

Note that [19] applies the mean value theorem, which does not provide the needed estimate. Finally, Lemma 1.2 delivers the desired estimate

$$\begin{aligned}\left\|z^{k+1}-\bar{z}^{k+1}\right\| \leq & \left(\beta\kappa+\frac{\beta\omega}{2}\left\|z^k-\bar{z}^k\right\|\right)\left\|\bar{z}^k-\bar{z}^k\right\| \\ & +\left(\beta\kappa\sigma+\frac{\beta\omega\sigma^2}{2}\|\xi_{k+1}-\xi_k\|\right)\|\xi_{k+1}-\xi_k\|\end{aligned} \tag{1.77}$$

with $\alpha=\beta\kappa$, $p_1=\frac{\beta\kappa}{2}$, $p_2=\beta\kappa\sigma$ and $p_3=\frac{\beta\omega\sigma^2}{2}$. Note that due to Assumption 3.a and the definition of β in Lemma 1.3 holds $\alpha=\frac{\gamma\kappa}{1-\gamma\kappa}<1$, $p_1=\frac{\gamma\omega}{2(1-\gamma\kappa)}\geq 0$, $p_2=\frac{\gamma\kappa\sigma}{1-\gamma\kappa}>0$ and $p_3=\frac{\gamma\omega\sigma^2}{2(1-\gamma\kappa)}$.

□

Remark 1.17. The parameter distance $\|\xi_{k+1}-\xi_k\|$ is assumed to be small, but it is not guaranteed that $\alpha+p_1\left\|z^k-\bar{z}^k\right\|<1$ holds. Therefore, a strict contraction, as in Theorem 1.1, is not shown yet. The following corollary gives us an upper bound for $\left\|z^k-\bar{z}^k\right\|\ \forall k\geq 0$.

Corollary 1.18. If Theorem 1.4 holds, there exists a positive number r_z with $0<r_z<\bar{r}_z=\frac{1-\alpha}{p_1}$. If the initial point z^0 in Algorithm 4 is chosen such that $\|z^0-\bar{z}^0\|\leq r_z$ and $\bar{z}^0\in Z^*(\xi_0)$ holds, for any $k\geq 0$ follows:

$$\left\|z^{k+1}-\bar{z}^{k+1}\right\|\leq r_z. \tag{1.78}$$

Provided that $\bar{z}^{k+1}\in Z^*(\xi_{k+1})$ holds and the parameter distance $\|\xi_{k+1}-\xi_k\|$ is less than $r_\xi r_\xi<\bar{r}_\xi$ with:

$$\bar{r}_\xi=\begin{cases}\frac{\sqrt{p_2^2+4p_3r_z(1-\alpha-p_1r_z)}-p_2}{2p_3} & \text{if } p_3>0\\ \frac{r_z(1-\alpha-p_1r_z)}{p_2} & \text{if } p_3=0\end{cases}. \tag{1.79}$$

Consequently, the sequence $\{\left\|z^k-\bar{z}^k\right\|\}_{k\geq 0}$, between the approximate KKT point z^k and the exact KKT point $\bar{z}^k$ of $P(\xi_k)$ is bounded by r_z.

Proof. See [19, page 14]. It is shown by choosing z small enough and applying equation (1.74). □

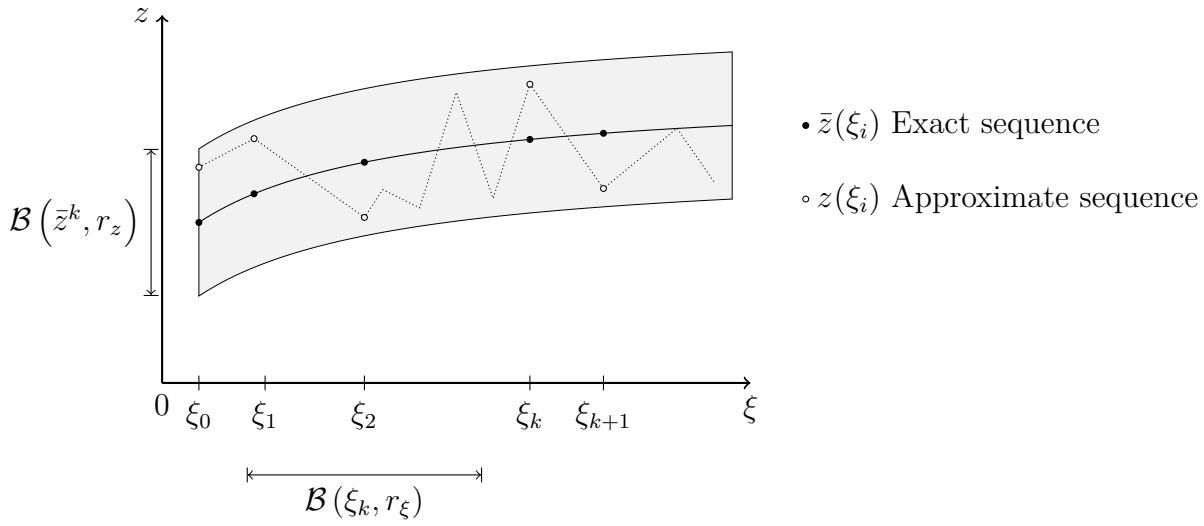


Figure 1.10.: Visualization of the obtained approximated KKT sequence $\{z^k\}_{k\leq 0}$ obtained by Algorithm 4 and the area where the estimates hold.

Remark 1.19. With this corollary we have shown that the sequence of approximated solutions $\{z^k\}_{k\geq 0}$ stays close to the sequence of exact solutions $\{\bar{z}^k\}_{k\geq 0}$ when the parameter change $\|\xi_{k+1} - \xi_k\|$ is less than r_ξ in every time step $k \geq 0$ and the initial approximate solution z^0 is already close to the exact solution $\bar{z}^0$. Although the obtained sequence may not converge to the exact sequence, it will always stay close to it. This is visualized in figure 1.10.

1.3.5. Contraction Estimate for PCSCP

In this section, a special case of the Algorithm 4 is examined. It is called *Predictor-Corrector SCP* (PCSCP) algorithm as $A_k = g'(x^k)$ holds. This has the consequence that $m^k = 0\ \forall k \geq 0$ and the subproblem $P(z^k, A_k, \widetilde{H}_k; \xi)$ (1.32) simplifies to $P(x^k, \widetilde{H}_k; \xi)$ given as

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x + \frac{1}{2}(x - x^k)^T \widetilde{H}_k (x - x^k) \\ \text{s.t.} \quad & g(x^k) + g'(x^k)(x - x^k) + M\xi = 0, \\ & x \in \Omega. \end{aligned} \tag{1.80}$$

Note that the Lagrange multiplier y^k has no influence on the subproblem if $\widetilde{H}_k$ is chosen independently of it. Assumption 3.a is about the approximation quality and simplifies to Assumption 3.a.

Assumption 3. *For a given $\bar{z}^k \in Z^*(\xi_k),\ k \leq 0$ holds:*

a') There exists a constant $0 \leq \widetilde{\kappa} < \frac{1}{2\gamma}$ such that:

$$\left\|\nabla_x^2 \mathcal{L}(\bar{z}^k) - \widetilde{H}_k\right\| \leq \widetilde{\kappa},\ \forall k \geq 0. \tag{1.81}$$

In [19, pages 15-16] is shown that this assumption induces in this setting Assumption 3.a. Similar to Theorem 1.4 and Corollary 1.18 a contraction theorem can be formulated.

Theorem 1.5. *Suppose that Assumptions 1 and 2 are satisfied for some $\xi_0 \in \mathcal{P}$. Then, for $k \geq 0$ and $\bar{z}^k \in Z^*(\xi_k)$, if $P_T(\xi_k)$ is strongly regular at $\bar{z}^k$, there exist neighborhoods $\mathcal{B}\left(\bar{z}^k, r_z\right)$ and $\mathcal{B}\left(\xi_k, r_\xi\right)$ such that:*

a) The set of KKT points $Z^*(\xi_{k+1})$ of $P_T(\xi_{k+1})$ is nonempty for any $\xi_{k+1} \in \mathcal{B}\left(\xi_k, r_\xi\right)$.

b) If, in addition, Assumption 3.a' is satisfied then subproblem $P(x^k, \widetilde{H}_k; \xi_{k+1})$ is uniquely solvable in the neighborhood $\mathcal{B}\left(\bar{z}^k, r_z\right)$.

c) Moreover, if Assumption 3.b holds as well then the sequence $\{z^k\}_{k\geq 0}$ generated by the PCSCP, where $\xi_{k+1} \in \mathcal{B}\left(\xi_k, r_\xi\right)$, guarantees

$$\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq (\widetilde{\alpha} + \widetilde{p}_1)\left\|z^k - \bar{z}^k\right\| + (\widetilde{p}_2 + \widetilde{p}_3\left\|\xi_{k+1} - \xi_k\right\|)\left\|\xi_{k+1} - \xi_k\right\| \tag{1.82}$$

where $0 \leq \widetilde{\alpha} < 1$, $0 \leq \widetilde{p}_i, i = 1, 2, 3$ and $\widetilde{p}_2 > 0$ are given constants and $\bar{z}[k+1] \in Z^*(\xi_{k+1})$.

d) If the initial point z^0 in the PCSCP is chosen such that $\|z^0 - \bar{z}^0\| \leq \widetilde{r}_z$, where $\bar{z}^0 \in Z^*(\xi_0)$ and $0 < \widetilde{r}_z < \bar{\widetilde{r}}_z = \frac{1-\widetilde{\alpha}}{\widetilde{p}_1}$, then

$$\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq \widetilde{r}_z \tag{1.83}$$

provided that $\|\xi_{k+1} - \xi_k\| \leq \widetilde{r}_\xi$ with $0 < \widetilde{r}_\xi \leq \bar{\widetilde{r}}_\xi$,

$$\bar{\widetilde{r}}_\xi = \begin{cases} \frac{\sqrt{\widetilde{p}_2^2 + 4\widetilde{p}_3\widetilde{r}_z(1-\widetilde{\alpha}-\widetilde{p}_1\widetilde{r}_z)} - \widetilde{p}_2}{2\widetilde{p}_3} & \text{if } \widetilde{p}_3 > 0 \\ \frac{\widetilde{r}_z(1-\widetilde{\alpha}-\widetilde{p}_1\widetilde{r}_z)}{\widetilde{p}_2} & \text{if } \widetilde{p}_3 = 0 \end{cases} \tag{1.84}$$

Proof. See [19, pages 15-16]. This theorem is shown by reducing it to Theorem 1.4 and Lemmas 1.2 and 1.3. □

Remark 1.20. Even though the exact Jacobian of the equality constraint ($A = g'(x^k)$) is used in every time step, there is no substantial better error bound given. The calculation of $g'(x^k)$ is time consuming for the most models.

1.3.6. Example

Setting

The following example optimization problem illustrates the idea of the APCSCP method:

$$\begin{aligned} \min_{x\in\mathbb{R}^2} \quad & -x_1 \\ \text{s.t.} \quad & (x_1)^2 + 3x_2 + \frac{7}{4} - 3\xi = 0, \\ & (x_1)^2 - 3(x_2)^2 + 1 \leq 0, \\ & x \geq 0. \end{aligned} \tag{1.85}$$

Where $\mathcal{P} = \{\xi \in \mathbb{R} | \xi > \frac{7}{6}\}$ is the set of possible parameters. For $f \in \mathscr{C}^\infty\left[\mathbb{R}^2, \mathbb{R}\right]$, $f(x) = -x_1$, $M = -3 \in \mathbb{R}^{1\times 1}$ and $g \in \mathscr{C}^\infty\left[\mathbb{R}^2, \mathbb{R}\right]$, $g(x) = (x_1)^2 + 3x_2 + \frac{7}{4}$ and as feasible set

$\Omega = \left\{x \in \mathbb{R}^2 | x \geq 0, \|(x_1, 1)^T\| \leq 3x_2\right\}$ we obtain a problem in the form of problem $P_T(\xi)$ (1.9). Note that Ω is convex and closed. As the goal function is unbounded in $\mathbb{R}^2$, at least one of the three given inequalities must hold with equality. If $x_2 = 0$ holds, the inequality does not hold at all. If $x_1 = 0$ holds, the goal function evaluates to 0. In the case $(x_1)^2 - 3(x_2)^2 + 1 = 0$, the stationary point $\bar{x}_\xi$ must also fulfill the equality constraint and therefore holds:

$$\bar{x}^\xi = \left(\sqrt{3\left(\xi - \sqrt{\xi}\right) - \frac{1}{4}}, \sqrt{\xi} - \frac{1}{2}\right). \tag{1.86}$$

It is easy to see that in this case $f(\bar{x}^\xi) = -\bar{x}_1^\xi < 0$ is true. Furthermore, at $\bar{x}^\xi$ the strong second order sufficient condition holds. For a series $\{\xi_k\}_{k\geq 0}$ with $\|\xi_{k+1} - \xi_k\| < r_\xi$, the APCSCP and PCSCP algorithms are applicable. This problem can also be solved with real-time SQP methods [14, pages 44 ff.]. In the following, the corresponding subproblems are formulated and then the methods are compared on the sequence $\{1.2 + 0.2k\}_{k\geq 0}$ for $k = 0, \ldots, 9$. The initial point x^0 is chosen as the correct solution $\bar{x}^{\xi_0}$ in all used methods.

The APCSCP subproblem in the kth time step is then:

$$\begin{aligned} \min_{x\in\mathbb{R}^2} \quad & -x_1 + \left(\begin{pmatrix} 2x_1^k \\ 3 \end{pmatrix} - \begin{pmatrix} 2x_1^0 \\ 3 \end{pmatrix}\right)^T y^k (x - x^k) \\ \text{s.t.} \quad & \begin{pmatrix} 2x_1^0 \\ 3 \end{pmatrix}(x - x^k) + (x_1^k)^2 + 3x_2^k + \frac{7}{4} - 3\xi_k, \\ & x \in \Omega. \end{aligned} \tag{1.87}$$

In this case, A_k is approximated with $g'(x^0)$ and $\widetilde{H}_k = 0$. If A_k is chosen as $g'(x^k)$ the PCSCP subproblem is:

$$\begin{aligned} \min_{x\in\mathbb{R}^2} \quad & -x_1 \\ \text{s.t.} \quad & 2x_1^k x_1 - (x_1^k)^2 + 3x_2 + 1.75 - 3\xi = 0, \\ & x \in \Omega. \end{aligned} \tag{1.88}$$

The equality constraint can also be written as $g(x^k) + g'(x^k)(x - x^k) + M\xi_k$. Note that both SCP-based algorithms have the same feasible set condition as the original problem. If the problem is solved with the real-time SQP method the corresponding subproblem is:

$$\begin{aligned} \min_{\Delta_{SQP}x} \quad & -\Delta_{SQP}x_1 + \frac{1}{2}\Delta_{SQP}x^T \widetilde{H}_k^{SQP} \Delta_{SQP}x \\ \text{s.t.} \quad & (x_1^k)^2 + 3x_2^k + \frac{7}{4} - 3\xi_k + \begin{pmatrix} 2x_1^k \\ 3 \end{pmatrix} \Delta_{SQP}x = 0, \\ & (x_1^k)^2 - 3(x^k)^2 + 1 + \begin{pmatrix} 2x_1^k \\ -6x_2^k \end{pmatrix} s \leq 0. \end{aligned} \tag{1.89}$$

This problem only provides an update direction $\Delta_{SQP}x$. The new iterate is given as $x^{k+1} = x^k + \Delta_{SQP}\bar{x}$. In this case, the equality constraint $g^{SQP}(x, \xi) = g(x) + M\xi$ is linearized. For the solution $\Delta_{SQP}\bar{x}$, this can be interpreted as $g^{SQP}(x^k) + g^{SQP'}(\Delta_{SQP}\bar{x}) = g(x^k) + g'(x^k)(x^{k+1} - x^k) + M\xi_k$. It has the same structure as in the PCSCP method. The

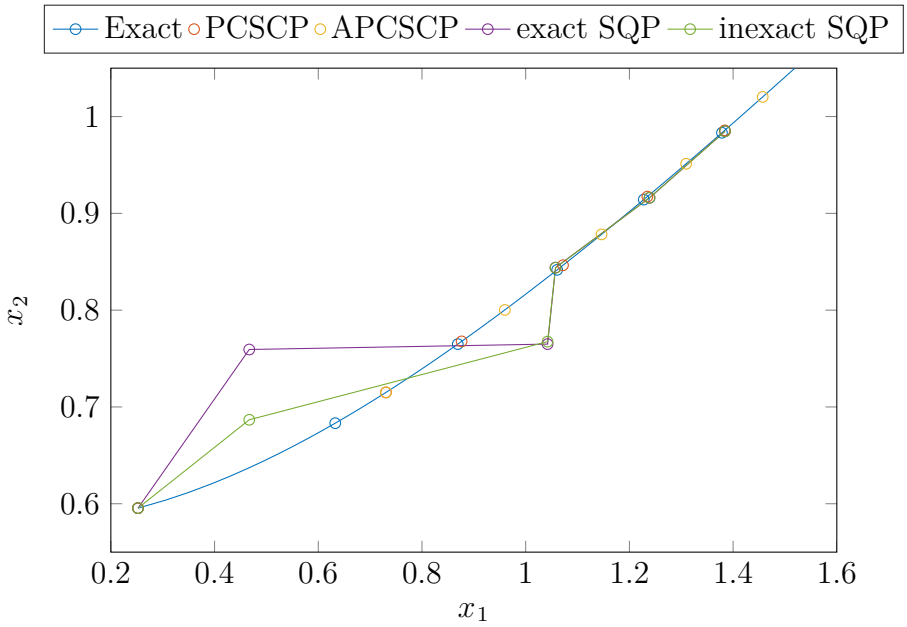


Figure 1.11.: Visualization of the obtained trajectories. In this figure only the results of the first six time steps are shown. As visible in figure 1.12 the further results have similar distances to the exact solution as the last two results in this figure.

goal function has the form $\nabla_x f(x^k)\Delta_{SQP}x + \frac{1}{2}\Delta_{SQP}x^T\widetilde{H}_k^{SQP}\Delta_{SQP}x$ and is a linearization of f. Similarly to the SCP-based methods, $\widetilde{H}_k^{SQP}$ approximates the Hessian of the Lagrangian. In the following experiment, the exact SQP method and the inexact SQP with $\widetilde{H}_k = 0$ are compared to the SCP-based algorithms. The non-negativity constraints are omitted in this case, as they are not in the active set (see figure 1.11) in this particular example.

Results

The results are visualized in figures 1.11 to 1.13. Note that the first time step of APCSCP and PCSCP leads to the same results. This is because in time step $k = 0$ the approximation $A_0 = g'(x^0)$ is exact and $m^0 = 0$. As the feasible set constraint was not linearized in the SCP-based methods, the inequality constraints are not violated, see figures 1.11 and 1.13. In this example, the tracking error of the PCSCP method has the same order as the errors of both SQP methods. In many real-time control problems the multiple shooting approach (section 1.1.3) leads to a high dimensional and complex structured equality constraint g. In this case the APCSCP algorithm requires only a directional derivative $g'(x^k)^T y^k$, which can be calculated significantly faster than the whole Jacobian; see Remark 1.11 and section 2.2.1. The disadvantage of the APCSCP method is, that the tracking error is larger than in the other methods, but as shown in Theorem 1.4 and Corollary 1.18 it is bounded for all time steps.

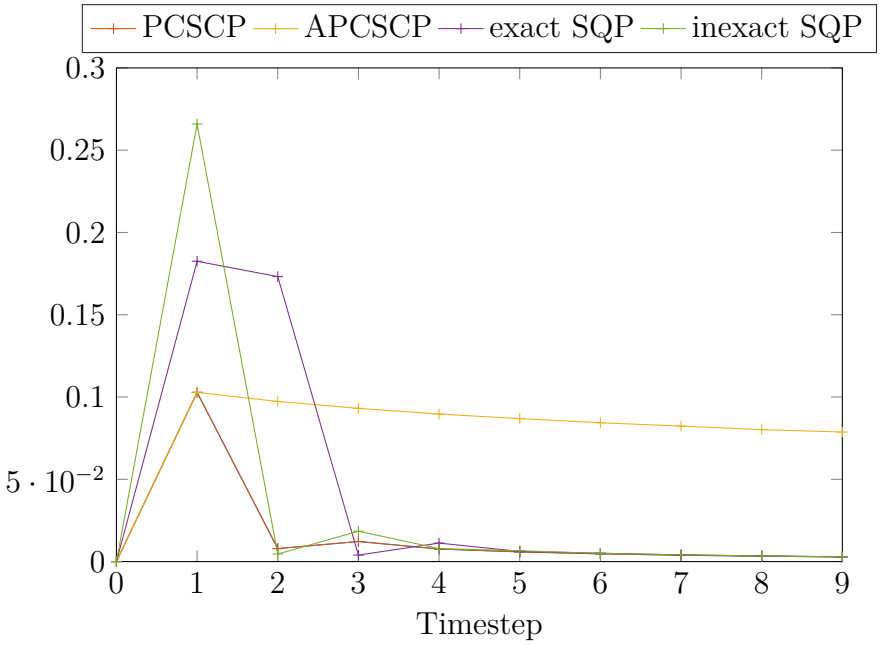


Figure 1.12.: Visualization of the tracking error. In the first time step, APCSCP and PCSCP method have the same result. Later, the derivative information of the equality constraint is used in PCSCP to reduce the tracking error.

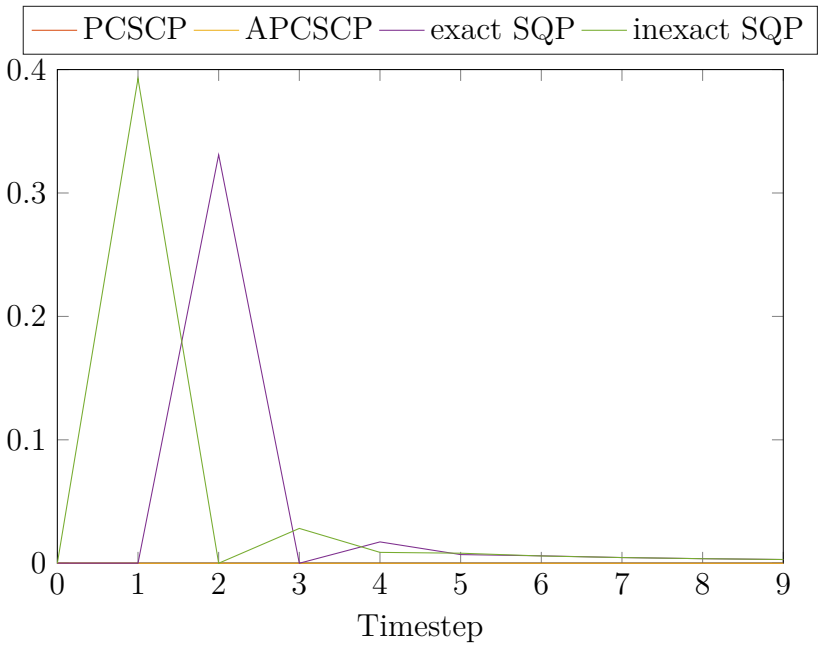


Figure 1.13.: Visualization of the violation of the constraint $(x_1)^2 - 3(x_2)^2 + 1 \leq 0$. As one can see in Figure 1.11, the non-negativity constraint always holds. In both SCP based algorithms the feasible set is preserved, all solutions are valid.

2. Hydro Power Plant

2.1. Model

For more than 3500 years humanity has been able to use the power of flowing water. At first water wheels were used to propel machines, such as mills or pumping stations. Since the invention of the generator in the 19th century, the generation of electrical energy has become more and more important, and is the main source in countries such as Brasil or Norway nowadays [29].
To reduce the effects of global warming, the emissions of carbon dioxide must be reduced. In Germany, power generation is responsible for more than a third of all carbon dioxide emissions [26]. This is by far the largest category, followed by traffic, which causes a fifth of all emissions. Renewable energy sources offer potential for enormous emission reduction. However, the transition to a sustainable energy market faces some technical challenges. The majority of renewable sources, such as solar or wind energy, depend highly on the weather and are therefore not controllable. If the share of these sources in the market is increased, it will become harder to provide enough energy e.g. in windless nights. Hydro power is one of the few controllable energy sources, as reservoirs can store water and the water flow changes slowly. If more energy is produced than consumed, the stability of the energy grid is also endangered. There are very few options to store electrical power efficiently. The most prominent one is battery storage using electrochemical processes, which is better suited for small amounts of power. Another option is pump-storage hydro plants, which convert electrical energy into potential energy.
Pump-storage hydro plants require a special topology, land consumption and the building costs are high, and the impact on the local ecosystem is enormous. Therefore, performance and profit should be maximized while additional requirements such as flood prevention should be assured.

The previously discussed algorithms can be applied here, as the flow of a river is changing very slowly. Furthermore is the real-time iteration with a standard solver like `IPOpt` (see section 2.2.1) time consuming, such that the application of fast approximative solvers is sensible.
The model in this work is based on the work in [19, 16, 29, 17] using the model from the Hydro Power Valley (HPV) benchmark, created in the project "Hierarchical and Distributed Model Predictive Control". In contrast, the plant in this thesis is simplified and does not contain river reaches. The model of ducts is replaced by an continuous differentiable model. The structure of the goal function in [19], where the Algorithm 4 is applied, is simple, and this thesis tests its performance on more complex functions where the energy generation is modeled properly.

2.1.1. Setting

Several *lakes* L are connected with *ducts* U, *pumps* P or *turbines* T. There are several *flows* q leading water into or out of the system. Pumps and turbines are controllable. The goal is to keep the lakes' water heights h at given levels and to maximize the plant's profit. Let $H_k \in \mathbb{R}^{n_{state}+n_{control}}$ denote the state of the whole hydro power plant at time step t_k. For the plant visualized in figure 2.1, this is

$$H_k = (h_{L_0,k}, \dots, h_{L_{3b},k}, q_{U_3,k}, q_{P_{10},k}, q_{T_{02},k}, q_{T_{12},k}, q_{T_{23},k}, q_{in,0,k}, q_{in,1,k}, q_{out,1,k})^T \in \mathbb{R}^{13}. \quad (2.1)$$

Flow data

The Bavarian Environmental Agency provides the flow data of every river in Bavaria [2]. The selected gauge for the numerical experiments was number 13 005 701, located in the centre of Munich at river chainage 145.90 of the river Isar. The flow is measured every 15 minutes. For smaller time steps, the data was linearly interpolated. To generate flow forecasts, mean filtering was applied to the exact data.This location was chosen because at this point the river is already a major river, providing enough water for a large hydro power plant and the river is still so small that intense rains during tempests or upriver plants lead to a significant flow change. Moreover, the riverbanks south of Munich are often steep and high and therefore, a theoretically appropriate location for a pumped hydro storage power plant.

Power prices

A pumped hydro storage plant takes advantage of varying power prices. It stores cheap electrical energy as potential energy and reconverts it during load peaks when the power price is high. In Western Europe, electrical power is a good which can be traded. The European Energy Exchange offers two spot market platforms. In day-ahead auctions, power is sold on a hourly basis. The intra-day auction is based on a quarter-hourly basis. The purpose of the intra-day auction is to compensate for short-term load peaks. The price data was provided by the chair of energy systems at the TUM Mechanical Engineering Department. Let ϖ_k^{da} denote the day-ahead auction price and ϖ_k^{id} the intra-day auction price in [€/MW h] at time point k.

Control strategy

First, a power production plan ϱ for the next day is computed, based on flow forecasts and day-ahead auction prices. The expected energy from this plan is sold on the day-ahead auction. This plan includes global information (e.g. global expected price and flow extrema). The actual production of this power is controlled in real-time, based on exact flow data and intra-day prices. If the real-time control does not produce exactly the amount of power sold in the day-ahead auction, the difference is bought or sold on the intra-day market. The short-sided real-time control has the freedom to take advantage of varying prices but is not too greedy, as it has to follow the production plan.

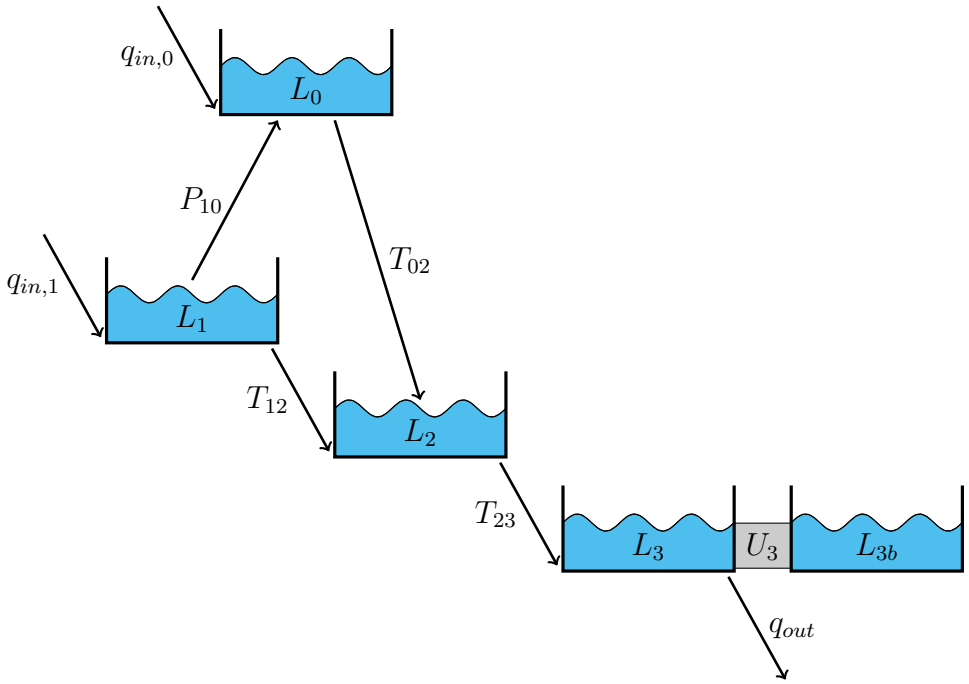


Figure 2.1.: Visualization of the hydro power plant model.

2.1.2. Components

Lake

A lake L has a changing water level $h_L(t)$. The total flow $q_L(t)$ is the sum of all components flowing into the lake. If in total more water flows into the lake than out of it, q_L is positive; otherwise it is negative. We also assume that vertical cross sections are rectangular and that the water surface S_L is constant. The lake height $h_L(t)$ is given by the following ordinary differential equation:

$$\frac{\partial}{\partial t} h_L(t) = \frac{q_L(t)}{S_L}. \tag{2.2}$$

When water flows into the lake, the water level has to increase. When the lake loses water, the height decreases. If lake L_1 has double the surface of lake L_2, double the amount of water must flow into L_1 for the same height increase. The lower bound for the height $h_L(t)$ is zero, which means that the lake is empty. The maximum height depends on the topography; typically, it is the height of the reservoir dam.

Duct

A duct U connects two lakes L_A and L_B with equal surface area, and its cross section area s_U is much smaller than the lake surface. Furthermore, it is assumed that the ducts are connected to the bottom of the lakes. Therefore, the duct is always completely filled with water, which simplifies the model. In [29, 19], Bernoulli's law is used to describe the flow $q_U(t)$ from L_A to L_B as:

$$q_{ber,U}(t) = s_U \operatorname{sgn}(h_{L_B}(t) - h_{L_A}(t) - \Delta\widehat{h}_U)\sqrt{2g\left|h_{L_B}(t) - h_{L_A}(t) - \Delta\widehat{h}_U\right|}, \tag{2.3}$$

where $\Delta\widehat{h}_U \in \mathbb{R}$ is the constant height distance between the bottoms of L_A and L_B and g is the gravity constant ($9.81\,[\mathrm{m/s^2}]$). The term $h_{L_B}(t) - h_{L_A}(t) - \Delta\widehat{h}_U$ describes the height difference between the lake surfaces. If there are no additional flows, the principle of *communicating vessels* states that this height difference will converge to the stable state zero for $t \to \infty$. From this principle we can also follow that, if at a time point t the water level of L_A is higher than the water level of L_B water flows into L_B, i. e. $q_U(t) > 0$. This contradicts equation (2.3) leading to a negative flow $q_U(t)$ in this situation. This thesis obtained a sign-corrected version describing the flow from L_A to L_B:

$$q_{inc,U}(t) = s_U \operatorname{sgn}(\Delta h_U(t))\sqrt{2g\,|\Delta h_U(t)|}. \tag{2.4}$$

The height difference between the water levels is $\Delta h_U(t) = h_{L_A}(t) - h_{L_B}(t) + \Delta\widehat{h}_U$. Equation (2.4) is not continuously differentiable as

$$\frac{\partial}{\partial t} s_U \operatorname{sgn}(\Delta h_U(t))\sqrt{2g|\Delta h_U(t)|} = \frac{\sqrt{g}s_U}{\sqrt{2|\Delta h_U(t)|}}\frac{\partial}{\partial t}\Delta h_U(t) \to \infty, \text{ as } t \to 0 \tag{2.5}$$

if $\frac{\partial}{\partial t}\Delta h_U(t)$ can be bounded. Therefore Assumption 1 does not hold and Algorithm 4 is not applicable. If there are additional flows into or from L_A or L_B, bounding is possible (see equation (2.2)). Note that even if no additional flows are connected, the derivative is piece-wise constant, but also incontinuous in the stable state $\Delta h_U(t) = 0$. This is also visualized in figure 2.2. The following two alternative models for a duct were developed.

Linear Flow As an alternative, it is assumed that q_U depends linearly on $\Delta h_U(t)$:

$$q_{lin,U}(t) = s_U\sqrt{2g}\Delta h_U(t) \tag{2.6}$$

If $\Delta h_U(t) = 0$ holds, $q_{lin,U}(t)$ equals $q_{inc,U}(t)$. As both functions are continuous, the distance to $q_{inc,U}$ is small in neighborhoods of these time points. For large Δh_U the it is unbounded and monotonously increasing. Nevertheless, the height difference Δh_U is bounded as well, as there are upper bounds for the lake heights. The numerical experiments in section 2.3 ran with linear flows as well, but were not successful.

Sigmoid Flow As a second alternative a sigmoid is used to model the flow

$$q_{sig,U}(t) = \widehat{s}_U\left(\frac{2}{1+e^{-\Delta h_U(t)}} - 1\right). \tag{2.7}$$

For $\Delta h_U(t) = 0$, $q_{sig,U}(t)$ is also zero. Therefore, the distance to $q_{inc,U}$ is small in neighborhoods of such time points. Like the Bernoulli's Law based model, this function has its steepest slope at $\Delta h_U(t) = 0$. For large Δh_U this distance is unbounded and increases monotonously as well. In this setting, the flow is bounded by $\pm\widehat{s}_U \in \mathbb{R}$.

Turbines

A turbine T_{AB} connects two lakes L_A and L_B. The amount of water flowing through T_{AB} is $q_{T_{AB}}(t)$ and is controllable ($T \in \mathfrak{U}$ the set of controllable components). The flow has $0\,[\mathrm{m^3/s}]$ as its lower bound. As the turbine has mechanical limits, an upper bound

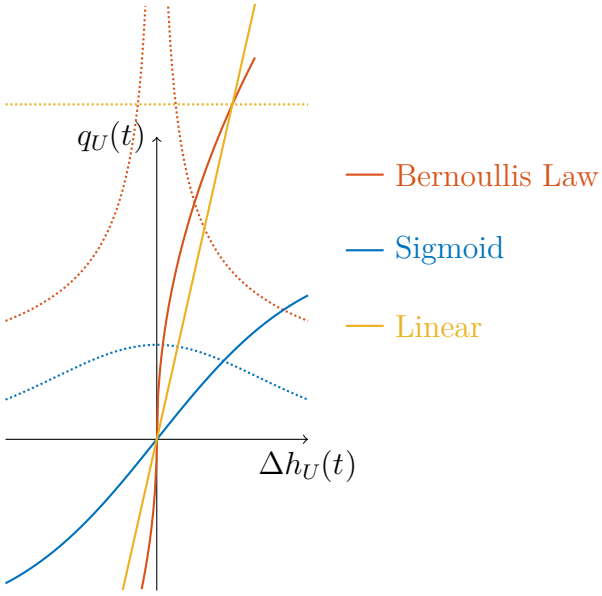


Figure 2.2.: Visualization of three different flow models and their derivatives (dotted).

is necessary. The energy of the flowing water is used to generate electrical power. The amount of electrical power [W] depends on the water flow and the height difference.

$$p_{T_{AB}}(t) = \zeta_{T_{AB}} q_{T_{AB}}(t) \Delta h_{T_{AB}}(t) \tag{2.8}$$

where

$$\zeta_{T_{AB}} = g \underbrace{\rho_{water}}_{\text{density of water}} \overbrace{\eta_{T_{AB}}}^{\text{efficiency}} \tag{2.9}$$

is a constant. For the efficiency $\eta_{T_{AB}}$ it holds that $0 < \eta_{T_{AB}} < 1$ and $\rho_{water} \mathrel{\widehat{=}} 998.204\,[\mathrm{kg/m^3}]$. The height difference

$$\Delta h_{T_{AB}}(t) = h_{L_A}(t) - h_{L_B}(t) + \Delta \widehat{h}_{T_{AB}} \tag{2.10}$$

depends on the water levels of both lakes and on the height difference $\Delta h_{L_A,L_B}$ between the lake bottoms. $\Delta \widehat{h}_{T_{AB}}$ is constant and positive iff L_A is higher than L_B. Equation (2.8) can be explained in the following way. The density of water $\rho_{water} q_{T_{AB}}(t)$ gives the mass of $q_{T_{AB}}(t)$ m^3 of water. The potential energy of this amount of water is therefore $g\rho_{water} q_{T_{AB}}(t) \Delta h_{T_{AB}}(t)$. As power is energy over time and flow is volume over time, $g\rho_{water} \Delta h_{T_{AB}}(t) q_{T_{AB}}(t)$ is the total amount of power. The turbine does not convert the entire amount into electrical energy since, for instance due to friction a certain amount is lost as heat energy. The factor $\eta_{T_{AB}}$ models this phenomenon and depends on the specific design and condition of the turbine.

Generated electrical energy In [19], the system's profit depends just on the prices and the flows in the controllable components. The influence of water heights (see equation (2.8)) is not considered. In this thesis themodel is extended. The amount of generated electrical

energy is given as:

$$E^{T_{AB}}_{[t_k,t_{k+1})} = \int_{t_k}^{t_{k+1}} p_{T_{AB}}(\tau)\mathrm{d}\tau = \int_{t_k}^{t_{k+1}} \zeta_{T_{AB}} q_{T_{AB}}(\tau)\Delta h_{T_{AB}}(\tau)\mathrm{d}\tau. \tag{2.11}$$

measuring the energy in [Ws] or [J]. In section 1.1.3, it is assumed that the control is piece-wise constant and therefore constant in the interval $[t_k, t_{k+1})$. This simplifies the integration to:

$$E^{T_{AB}}_{[t_k,t_{k+1})} = \zeta_{T_{AB}} q_{T_{AB}}(t_k) \int_{t_k}^{t_{k+1}} h_{L_A}(\tau) - h_{L_B}(\tau) + \Delta\widehat{h}_{T_{AB}}\mathrm{d}\tau. \tag{2.12}$$

If a lake L is not connected to a duct, the total flow q_L in $[t_k, t_{k+1})$ is constant as well. Therefore, $h_L(t)$ is linear in this interval. If both lakes have only constant flows, the energy has the form:

$$E^{T_{AB}}_{[t_k,t_{k+1})} = \zeta_{T_{AB}} q_{T_{AB}}(t_k) \left[\frac{h_{L_A}(t_k) + h_{L_A}(t_{k+1}) - h_{L_B}(t_k) - h_{L_B}(t_{k+1})}{2} + \Delta\widehat{h}_{T_{AB}} \right] \underbrace{(t_{k+1} - t_k)}_{\Delta t}. \tag{2.13}$$

Note that for lakes connected to a duct, this corresponds to a basic finite differences integration formula. The energy production of the whole system is denoted as

$$E_{[t_k,t_{k+1})} = \sum_{\mathfrak{u}\in\mathfrak{U}} E^{\mathfrak{u}}_{[t_k,t_{k+1})}. \tag{2.14}$$

The basic rules of integral calculus allow the calculation of arbitrary intervals with the formula given in equation (2.13).

Pump

A pump P_{AB} pumps water with variable flow $q_{P_{AB}}$ from L_A to L_B and converts electrical energy into flow energy:

$$p_{P_{AB}}(t) = \zeta_{P_{AB}} q_{P_{AB}}(t)\Delta h_{P_{AB}}(t), \tag{2.15}$$

where

$$\zeta_{P_{AB}} = g \underbrace{\rho_{water}}_{\text{density of water}} \overbrace{\eta_{P_{AB}}}^{\text{efficiency}} \tag{2.16}$$

is a constant. Now, electrical energy is converted into flow energy, the efficiency must be larger than 0. $\Delta h_{P_{AB}}(t)$ is measured in the same way as in the turbine case (2.1.2), but now L_B is higher than L_A. Hence, the difference is negative and therefore the power generation is also non-positive. The flow is also controllable ($P_{AB} \in \mathfrak{U}$).

Remark 2.1. We see that a pump is a turbine with a negative height difference and has a reciprocal efficiency. In the implementation, pump and turbine were represented by the same class `Cflow`, which had different parameters. Furthermore, equation (2.13) also holds for pumps and leads to non positive results, indicating that power is consumed and not produced.

Remark 2.2. A component, which combines a turbine and a pump can not be modeled by setting the lower bound to a negative value, because in the negative flow direction the efficiency is then larger than one and this violates the "law of conservation of energy". It is possible to model this device in both a separate pump and turbine component with no further restrictions. As both efficiencies cannot be chosen as 1, the elevation of a given amount of water costs more power than the drop of the same amount produces. Hence an optimal solution will set the flow of one component to 0 at any time point.

External flows

An incoming flow $Q_{in,i}$ brings water into the system. The flow $q_{in,i}(t)$ is given but not controllable, and like the controls, assumed to be piece-wise constant. An outgoing flow $q_{out,j}$ removes water from the system.The flow is controllable ($Q_{out,j} \in \mathfrak{U}$), but it does not generate any electrical energy, i. e. $E^{Q_{out,j}}_{[t_k,t_{k+1})} = 0 \; \forall t_k, t_{k+1} > 0$.

2.1.3. Goal Function

Global goal function

A hydro plant control has to satisfy the following needs:

1. *Keeping water heights at a desired level.*
2. *Maximizing profit.*

The goal function for the offline problem has the ansatz

$$f(x) = \sum_{k=0}^{k_{max}} \left(\sum_{x_i \in H_k} s_i (x_i - \iota_i)^2 - \kappa \vartheta E_{[t_k,t_{k+1})} \varpi_k^{da} \right) \tag{2.17}$$

where $s_i \in \{0,1\}$ selects if ξ_i is considered for optimization and $\iota_i \in \mathbb{R}, s_i = 0 \Rightarrow \iota_i = 0$ is a desired value. $\kappa \in \mathbb{R}$ balances the two different needs.

1. *Keeping water heights at a desired level.* This is modeled in the minuend. Iff a lake should be kept at a desired water level, the corresponding s_i is set to 1 and the corresponding ι_i to the desired water level.
2. *Maximizing profit.* This need is modeled in the subtrahend. Note that maximizing the profit is equivalent to minimizing the negative profit. The factor $\vartheta = \frac{1}{3600*10^6}$ converts the unit [Ws] into [MWh] and is necessary as the prices are given in [€/MWh].

Real-time goal function

The real-time goal function has to meet the following requirements:

1. *Keeping water heights at a desired level.*
2. *Maximizing profit.*
3. *Sticking to the planned power production.*

The ansatz of the goal function at time point k is similar to the global goal function f defined in equation (2.17):

$$f(x) = \sum_{j=k}^{k+T} \left(\sum_{x_i \in H_j} s_i (x_i - \iota_i)^2 - \kappa \left(\vartheta E_{[t_j, t_{j+1})} - \varrho_j \right) \varpi_j^{id} \right), \tag{2.18}$$

where $\iota_L, \kappa \in \mathbb{R}$ and $\iota_L \in \{0, 1\}$ are identical with their correspondents in the global goal function 2.17. The power production plan ϱ_j is the power production given by the global solution.

1. *Keeping water heights at a desired level.* Identical with 2.1.3.
2. *Maximizing profit.* Here the intra-day prices are used, see 2.1.1.
3. *Sticking to the planned power production.* Based on the power production plan the plant has to produce/consume at time t_j ϱ_j[MWh]. Therefore the profit calculation is shifted by this value.

Duct damping If the model contained any ducts, the numerical experiments showed that it is useful to reduce the duct flow. This is achieved by setting the corresponding s to 1 and the corresponding ι to 0.

2.1.4. Errors

Every 15 minutes a random error is added. The maximum error is $\pm 1\%$ of the actual value. If the time step is smaller than 15 [min], the error added is proportionally smaller, too.

2.2. Implementation

In order to examine the performance of the solvers discussed in chapter 1, Algorithms 2 and 4 were implemented and applied on the models developed in chapters 2 and 3. The developed theory requires the solution of nonlinear problems and integration of differential algebraic equations. A comprehensive discussion of these problems or their implementation from scratch exceeds, by far, the scope of a thesis, hence only a short overview of the used libraries and methods will be given in section 2.2.1. In section 2.2.2 the structure of the newly implemented `C++` code is explained.

2.2.1. Used Libraries

CasADi

`CasADi` [3] is a minimalistic computer algebra system (Cas) implementing automatic differentiation (AD). AD is a technique for evaluating derivatives of computer represented functions. In nonlinear optimization, derivative information is an important factor and AD allows faster development cycles as the required derivatives are calculated automatically. The approach used in `CasADi` delivers directional derivatives of arbitrary differentiable

functions up to machine precision, using not more than four times the number of floating points operations necessary to evaluate the original function. In the *forward* mode, the technique delivers a Jacobian times vector product of a vector valued function. The *reverse* mode provides a vector times Jacobian product. The computational costs are in both modes of the same order as the cost of evaluating the original function. A multiple evaluation leads to complete Jacobians. The computational cost is proportional to the number of independent variables in the forward mode and to the dimension of the function in the reverse mode, see [4]. The computer algebra system (Cas) gives a representation of matrices and executes operations such as addition, multiplication, transposition and accessing elements, rows, or columns.
`CasADi` is a `C++` library and has interfaces to nonlinear problem solvers like `IPOpt` (see section 2.2.1) and integrators such as `cvodes` and `idas`. It offers a fully-featured front-end to `Python` and `MATLAB`. In this thesis, the control algorithms were implemented in `C++` only.

IPOpt

`IPOpt`, as described in [42], is an iterative interior point solver. The computationally complex task of identifying active constraints is avoided. The outer loop of `IPOpt` is given in Algorithm 5 and consists of solving a barrier problem. For the following restricted optimization problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t. } g(x) = 0 \\ &x_i \geq 0 \ \forall i \in \mathcal{I} \subseteq \{1,\dots,n\} \end{aligned} \quad , \tag{2.19}$$

the corresponding barrier problem is given as

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) - \varrho \sum_{i\in\mathcal{I}} \ln x_i \\ &\text{s.t. } g(x) = 0 \end{aligned} \tag{2.20}$$

with *barrier parameter* $\varrho > 0$. Note that this approach is not restricted to non-negativity inequality constraints. Especially the inequality constraints of the from

$$x_{min,i} \leq x_i \leq x_{max,i} \tag{2.21}$$

are possible. To solve the barrier problem, a primal-dual method is used. For this purpose, dual variables are introduced and the KKT system of the barrier method is solved by a Newton-type approach. For $x_i \searrow 0$, the barrier term $-\varrho \ln x_i$ diverges to infinity and therefore the local solution $\bar{x}$ for this problem lies in the interior of the feasible set $\{x \in \mathbb{R}^n | x_i > 0, i \in \mathcal{I}\}$. x_i approaches 0 only, if ϱ approaches 0 in the steps of the outer loop. As the inequality $x_i > 0$ always holds, these methods are also often called *interior point* methods.

Integrator

To calculate the integration, the solvers `cvodes` and `idas` from the `sundials` suite were used. The automatic differentiation framework `CasADi` provides an interface to both solvers and also their derivative information.

Algorithm 5 IPOpt outer loop

1: **loop** $l \in \{0, 1, \ldots, l_{max}\}$
2: Determine error tolerance.
3: Obtain an approximate solution to the barrier problem with barrier parameter ϱ, satisfying the above determined error tolerance. Use the previous solution as starting point.
4: Decrease the barrier parameter ϱ.
5: **end loop**

cvodes This solver uses the "Backward Differentiation Formula" (BDF) method [11] to solve the initial value problem. The method is implicit and at order 1 it is the implicit Euler method. In the majority of examples, the default order 5 was used.

idas This solver uses the "Generalized minimal residual Algorithm" (GMRES) [35] to solve the initial value problem. The maximum Krylov subspace size was set to 10. In contrast to `cvodes`, `idas` is able to solve differential algebraic equations.

2.2.2. Structure of Source Code

The code is designed in an object oriented way and follows the terminology of this thesis. Due to its modular architecture, modification or exchange of used models and solvers is easy. For real-world applications, where exactly one model is solved with a certain solver on given hardware, faster implementations are possible. But if the scope is on rapid prototyping, the object-oriented design is better suited. For time measurements the `chrono` time library is used and for data import the `boost` library is used.
The main class is called `Global_Realtime`. It defines the basic settings such as $T, \Delta t$, number of possible threads, properties of the used solvers and the simulated time interval $t_{k_{max}}$. Then the used model is initialized and a solution for the whole simulated time interval is computed. Afterwards several real-time solvers are tested on the model and the results are saved. The program terminates when the last solver is finished.
The source code is based on three abstract superclasses `Component`, `Model`, and `Solver`. In the following the basic concepts of every superclass are sketched.

Solver

This abstract class represents either a *global solver*, which provides a solution of the model for the whole simulated time interval or a *real-time solver*, which is a implementation of Algorithm 2. It stores solution statistics, for instance if the solution of a subproblem was successful, and timing properties. Furthermore, it prints the results into a `MATLAB` file for analysis.

Global Solution The class `Global_Solver` inherits from `Solver` and solves the problem for the whole simulated time interval. This can be used, for instance, to calculate the

power production plan ϱ. In this case, the only optimization problem is solved directly with `IPOpt`. Note that the term *global* refers here and in the following to a solution over the whole simulated interval.

Real-time Solution In this algorithm, a sequence of optimization problems must be solved. The solver `RealtimeIpOpt` (RT IPOpt) solves every problem of Algorithm 2 directly with `IPOpt`. To increase the performance, warmstart options were used and the Lagrange multipliers of the previous problem were given to the solver. This is possible, as the expected change between two consecutive optimization problems is assumed to be small.
The solver `RealtimeAPCSCP` is a implementation of Algorithm 4. For possible choices of the approximations, see appendix A.3. Note that if $A = g'(x^k)$ is chosen this is also an implementation of the PCSCP algorithm, see section 1.3.5.

Model

The central element of the source code is the abstract class `Model`. Its core member is a list of all involved components and a vector `V`, which represents x. The class provides the solver with goal function, equality and inequality constraints. The inequality constraints are the boundary values of the involved components and the equality constraint is obtained by multiple shooting, see section 1.1.3. Moreover, this class obtains the differential algebraic equation describing the dynamics from the list of components and stores the used integrator. This class is abstract and has no knowledge of the individual components. This is implemented with an iteration through the list of components, because there the required information is stored. This leads to a more complex source code in this class, but has the advantage, that the assembly of the optimization problem is separated from individual components and the order of the components in `V`, which allows any combination of components.
Note that the goal function is an abstract function which must be implemented in the subclass as it depends on the individual model. In the constructor of every subclass all components are initialized, connected, and added to the list of components. In the following section, different hydro plants are tested (see sections 2.3.1 and 2.3.2). To reduce the coding effort, an additional abstract class `HydroModel`, which inherits from `Model` is useful to bundle functions specific to hydro models. In chapter 3 this is omitted, as only one specific setting tested.
As mentioned in section 1.1.3, the order in x (1.7) can be arbitrarily permuted, as long as it is done consistently. In this implementation, `V` is ordered by component and then by shooting nodenumber. This simplifies accesses on `V` in the evaluation of goal function and multiple shooting as then only consecutive slices of `V` are needed. This is easier to represent in `CasADi` and computationally faster. Note that this is a contrast to (1.7), where x is ordered by shooting node and then by component.

Component

This class represents the components described in sections 2.1.2 and 3.1.1. It provides the model with information about differential or algebraic equations, the lower and upper

bounds and initial values. The abstract superclass `Component` stores these bounds, results from previous iterations, desired values, and the place, where the component is located in `V`. Furthermore, the update process between two time steps in the real-time iteration is implemented here. Random errors can be loaded from a file to ensure that every real-time solver faces the same errors.
Depending on the properties of the modeled component, a component may have knowledge of other components. For instance, the component `lake` (section 2.1.2) needs a list of all components flowing into or out of it, because the total flow is part of the differential equation describing the behaviour of the lake. In this implementation, all information about the component is stored in the component and not administered by the model. A flow is either `Cflow` representing a pump or turbine, or an external flow `ExtFlow`. In this case an additional abstract class `Flow` is useful, which is inherited from `Component` and implements all general flow properties.

2.3. Numerical Results

2.3.1. Single Lake

Exact Model

As a starter, the single lake example in section 1.1.2 is simulated. The inflow q_{in} is given as the flow data of the 3rd of June 2015, see section 2.1.1 and figure 2.4. In this setting, it is assumed that the sensor measures the height exactly and that the system behaves exactly as modeled. The time step is chosen as $\Delta t = 7.5\,[\text{min}] = 450\,[\text{s}]$ and the horizon T is set to $2\,[\text{h}]$. The lake has a surface of $80\,000\,[\text{m}^2]$ and a maximum height of $30\,[\text{m}]$. The initial lake height is $13\,[\text{m}]$ and the desired height is $15\,[\text{m}]$. Both SCP-based solvers approximate the Lagrangian with $\widetilde{H} = 0$. In the APCSPC solver the Jacobian of the equality constraint is updated every fifth step.

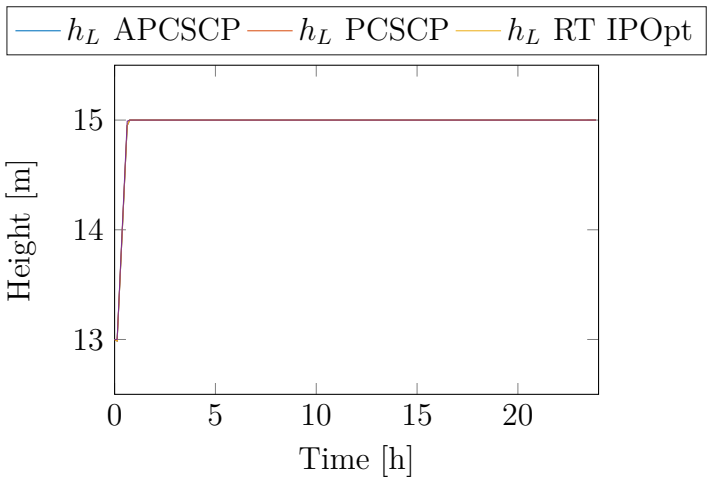


Figure 2.3.: Visualization of the heights in the single lake exact model example.

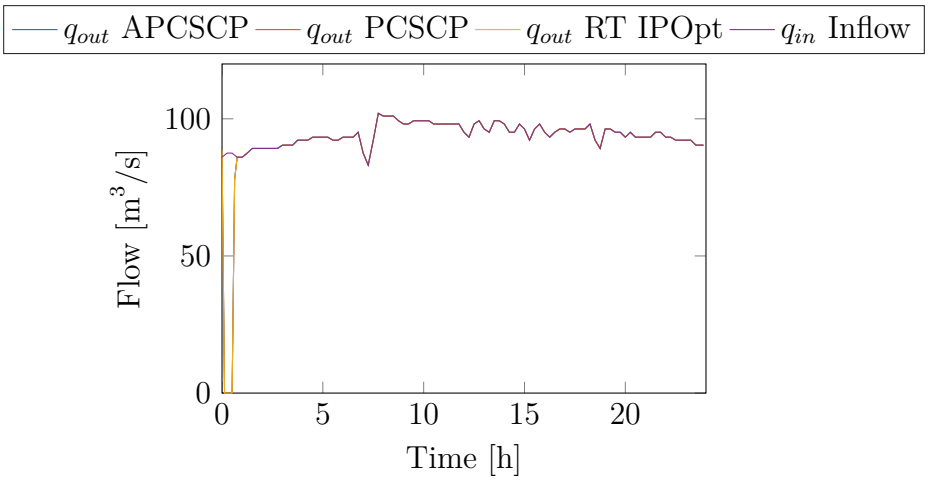


Figure 2.4.: Visualization of the different flows in the single lake exact model example. Note that all three real-time solvers have the same result.

The lake height is visualized in figure 2.3. In this simple example all solvers are able to obtain the same solution. The obtained solutions of all three real-time solvers are identical. The solution time is visualized in figure 2.5. Here the SCP-based solvers deliver the solution in about 10 % of the time the exact real-time solver needs.

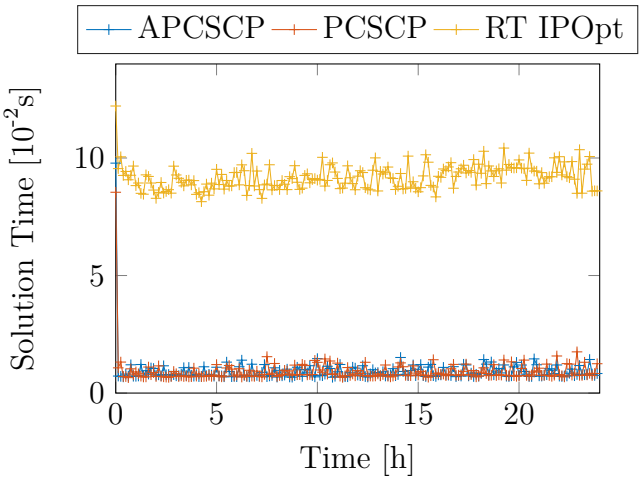


Figure 2.5.: Visualization of the solution time in the single lake exact model example.

Faulty Model

In contrast to the previous setting, the exactness assumption is removed. As described in section 2.1.4, in every time step, an error is added to the lake height and the inflow. This error represents two things. Firstly, the sensor measurement may be inaccurate. Secondly, the model is inaccurate, as e.g. rain is not modeled. The result is visualized in figures 2.6 and 2.7.

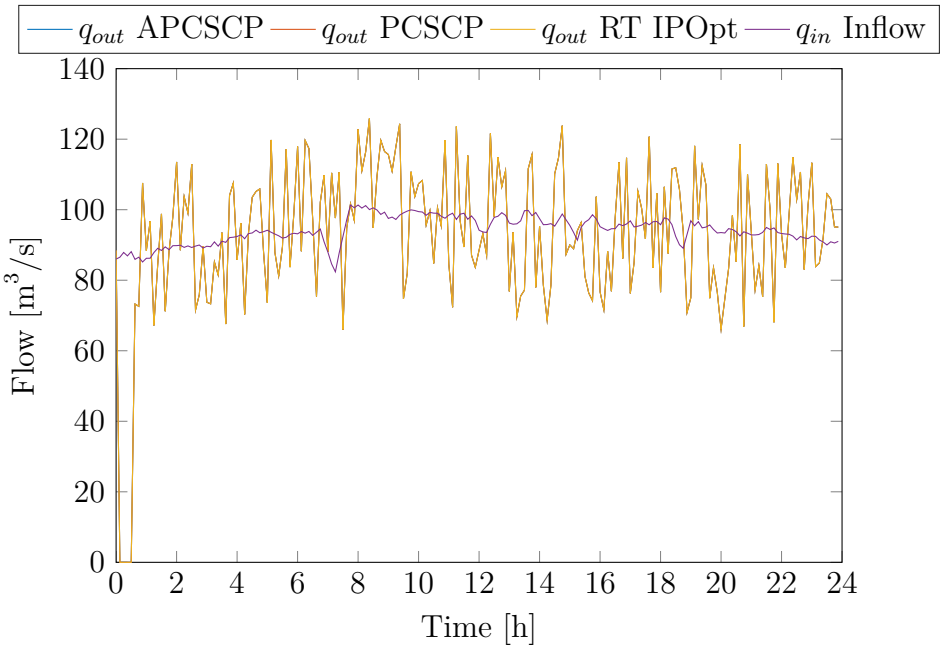


Figure 2.6.: Visualization of the different flows in the single lake inexact model example.

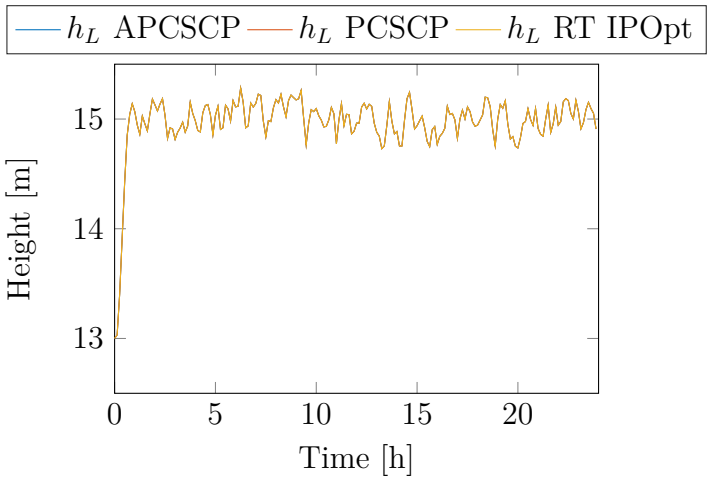


Figure 2.7.: Visualization of the heights in the single lake inexact model example.

As in the previous example, the solutions are all identical. Due to the unknown external error, it is not possible to match the desired height exactly. After the desired height is reached, the system is able to keep the distance to the desired height smaller than 0.3 [m], see figure 2.7. In this example the deviation from the desired lake height large, but the control operates with a time step Δt of 7.5 [min]. In figure 2.6 the obtained flows are visualized. Note that the obtained flow under- and over-estimates the actual flow. Due to the real-time iteration, the overestimations cancel out and the lake height stays close to the desired height.
The solution time of all algorithms are visualized in figure 2.8 and increased just slightly. Furthermore, the generated graphs are less smooth than the corresponding graphs in the

exact model. This is a direct consequence of the error, which leads to increased deviation from the desired values.

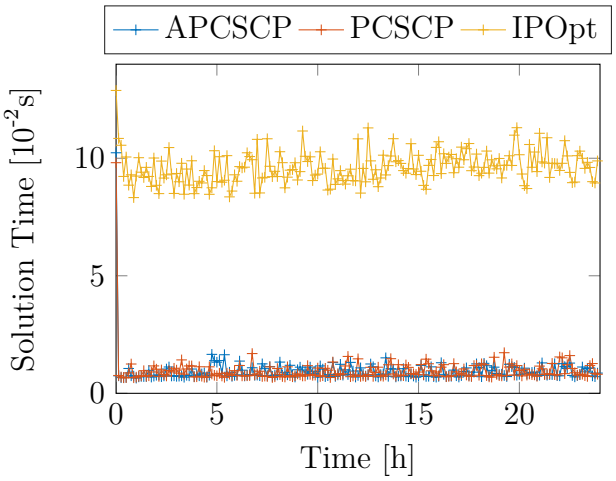


Figure 2.8.: Visualization of the timing in the single lake inexact model example.

2.3.2. Stabilization of a Hydro Plant

In this example, the plant visualized in figure 2.1 is simulated. At first, the offline global solution is calculated. The goal is to keep the lake heights at a desired level and to optimize the profit; the balancing parameter is chosen as $\kappa = 10^{-4}$. Note that the deviation from the desired lake heights shall be in the magnitude of 0.1 [m] and the profits are in the magnitude of 10^3[€]. This choice puts the main focus on the stabilization, but the impact of the price is still visible. The features of all components are listed in tables 2.3 and 2.4 and section 2.3.2. The storage lake L_0 has the smallest surface, as elevated area is typically limited. The mountainside allows larger maximal heights. The prices are the day-ahead prices from the 3rd of June 2015, visualized in figure 2.9.

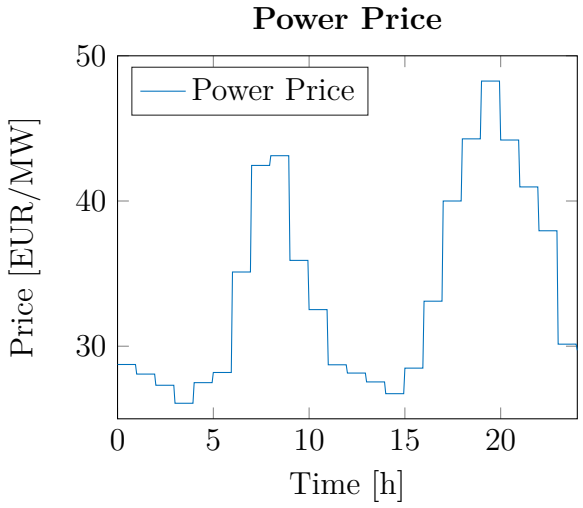


Figure 2.9.: Visualization of the power price used in the offline step in the example described in section 2.3.2. The price changes every hour.

In the real-time iteration, the goal function is a semi-norm. The goal is to follow the previously obtained optimal lake heights. To prevent an overexpensive usage of the pump P_{10}, this flow should also follow the optimal solution. The price information is only implicitly contained in the optimal lake heights, see figure 2.10. As the profit is not optimized, the balancing parameter κ is obsolete and set to 1. In the real-time setting, the predictions of both inflows $q_{in,0}$ and $q_{in,1}$ are faulty, see section 2.1.4. The step size in all calculations was set to $\Delta t = \frac{15}{4}\text{min} = 225\,\text{s}$ and the horizon of the subproblems was set to 2 hours. In the real-time settings, 22 hours were simulated.
In order to compare the results with [19], the setting here was chosen to be as close as possible. The authors use the `cvodes` integrator to solve a differential algebraic equation. The version of `cvodes` used in this implementation was not able to solve any algebraic differential equation and the `idas` solver from the `Sundials` package was used as integrator. Additionally is, as we have seen in section 2.1.2, their duct model in the stable state not continuously differentiable. Due to these problems, the duct flow was modeled as sigmoid
In the real-time iteration, four different algorithms were tested. Solving every subproblem of Algorithm 2 directly with `IPOpt`. Applying PCSCP with $\widetilde{H} = 0$. Algorithm 4 was tested with two different update strategies for A. The first one (APCSCP A5) calculates the exact Jacobian in every fifth step. In APCSCP A768 the exact Jacobian is just calculated in the initial step. Both algorithms use $\widetilde{H} = 0$. The choice of these algorithms corresponds to the numerical experiment in [19].
The result of the global solver (i.e. for 24 hours) consisted of 4219 variables with 2304 equality constraints and the solution took 9.6 minutes. The obtained lake heights are visualized in figure 2.10.

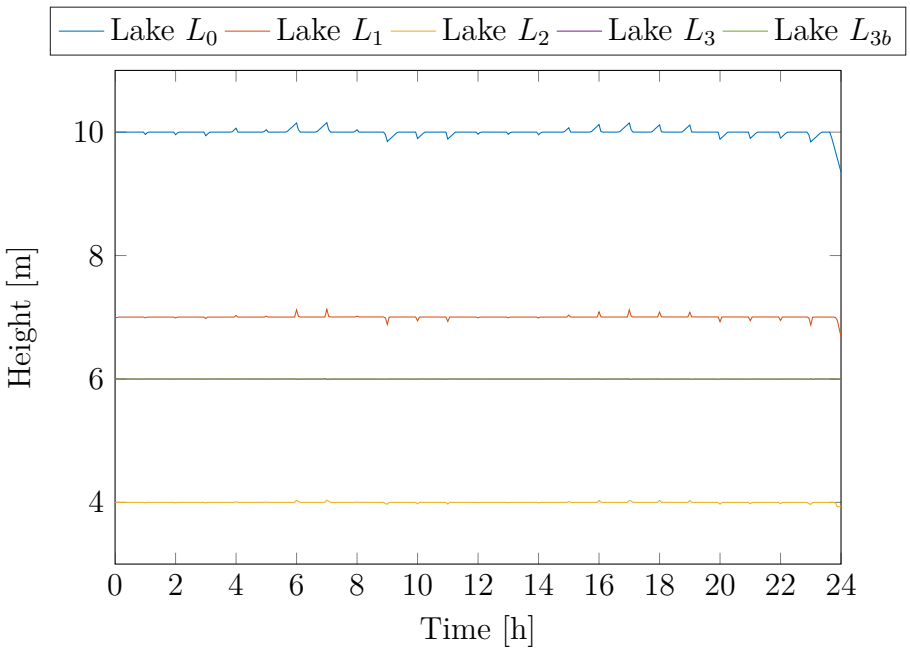


Figure 2.10.: Visualization of lake heights in the global solution of the example described in section 2.3.2.

The desired heights are matched closely. In the neighborhood of changing prices, water is stored, to take advantage of the price difference. If prices are increasing, water is stored

in the lakes and released in the next hour, when the price is higher. If prices are falling, the water level decreases first. The higher the price difference, the larger the deviation. The difference in the deviations between the lakes have two different reasons. Firstly, L_0 has a smaller surface than L_1. Secondly, turbine T_{02} is more efficient and has the largest height difference. L_3 does not have any deviations, as its outflow q_{out} does not produce any energy. L_{3b} has the same height as L_3, with small deviations in the phases where T_{23} has larger flow changes. In the end, the water level of L_0 and L_1 significantly decreases. This occurs, as the terminal lake height was not fixed with constraints. The flows in duct, pump, and turbines are visualized in figure 2.11.

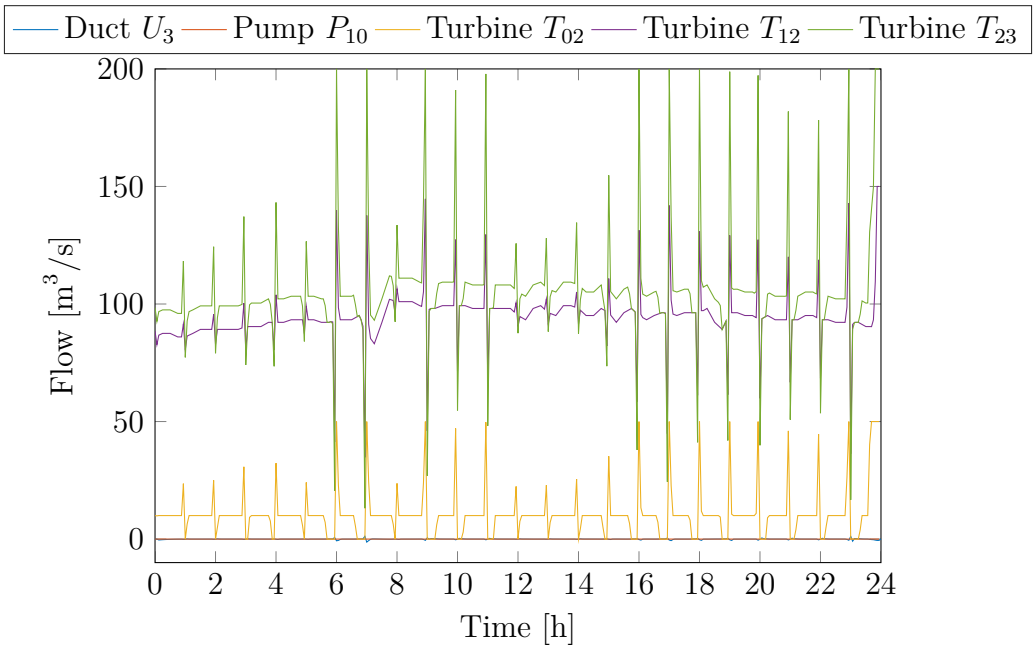


Figure 2.11.: Visualization of ducts, pump and turbine flows in the global solution of the example described in section 2.3.2.

The goal was to keep the lake heights at desired levels and to maximize the profit. The pump is not active, as the focus of the goal function is more on the stabilization of lake heights. The flows in the turbines change strongly in intervals, where the price is about to change. The higher the price difference, the higher is the flow difference. In the last few intervals, more water is released as the resulting changes in the lake heights are beyond the considered time frame. To complete the image, the external flows are visualized in figure 2.12. The inflows are given. Flow $q_{in,0}$ is constant, and $q_{in,1}$ is the same real world flow data as in section 2.3.1. The outflow can be set freely. In this example, it follows the flow in T_{23} to keep the total flow of L_3 zero.

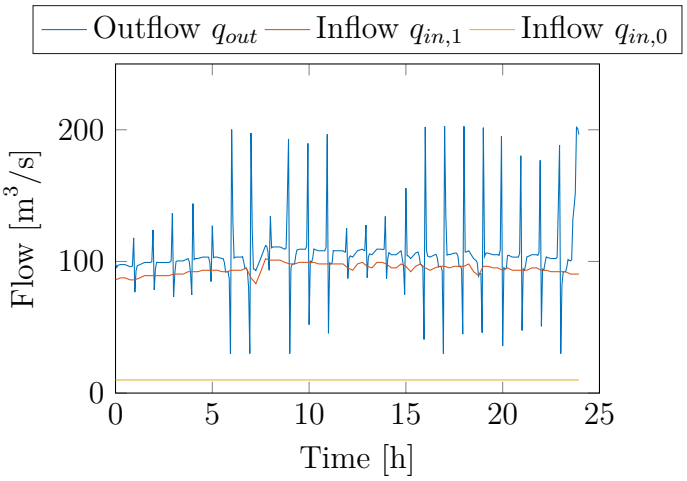


Figure 2.12.: Visualization of the external flows in the global solution of the example described in section 2.3.2.

As the results of the three SCP-based algorithms are all similar, only APCSCP A5 is discussed in detail, see figures 2.13 to 2.15. Each subproblem consisted of 347 variables with 192 equality constraints. The lake heights obtained in the real-time iteration are visualized in figure 2.13.

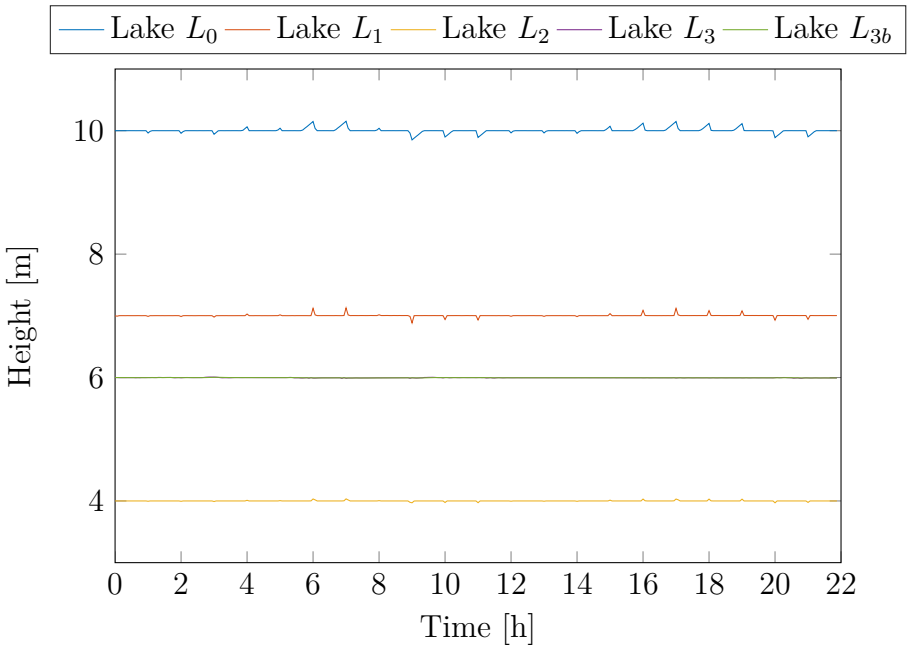


Figure 2.13.: Visualization of lake heights obtained by APCSCP A5 for the example described in section 2.3.2.

The heights of lakes L_1 and L_2 match closely to the solution obtained by the global solver. Note that there is no drop in the end, because the simulated time frame in this example is shorter than in the global solution step. This is because the real-time iteration in this example considers the next two hours in every time step. In real-world applications, a new

global solution for the next day is used to obtain the control between 22 and 24 [h]. The flows in duct, pump, and turbines are visualized in figure 2.14.

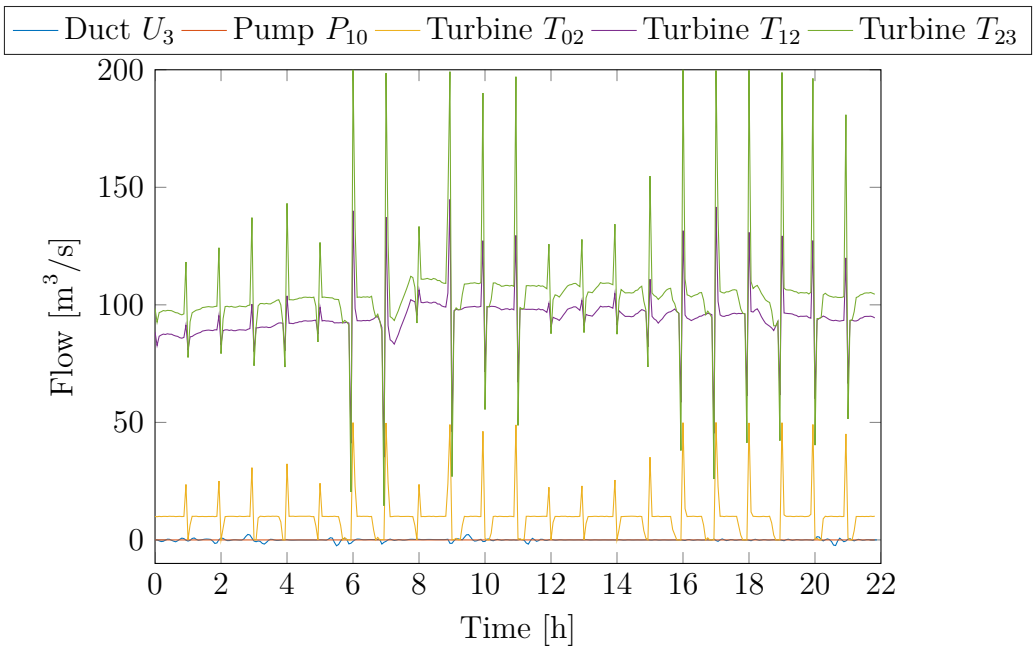


Figure 2.14.: Visualization of duct, pump, and turbine flows obtained by APCSCP A5 for the example described in section 2.3.2.

The result of APCSCP A5 follows the global solution shown in figure 2.11. In this example, the goal is to follow the lake heights from the global solution and the pump flow. The pump flow has to follow the global solution as the real-time solver has no knowledge of the power price. Otherwise, the pump would be used to compensate for the faulty inflow and consume a lot of energy. The price information is implicitly contained in the optimal lake heights. In intervals not adjacent to a price change, the outflow follows the sum of both inflows. But the intervals not adjacent to an price change are less smooth. This can be explained, as both inflow predictions are assumed to be faulty. Compared to the global solution, there is more flow in U_3. This occurs when the changes in T_{23} and q_{out} are high. The external flows are shown in figure 2.15.

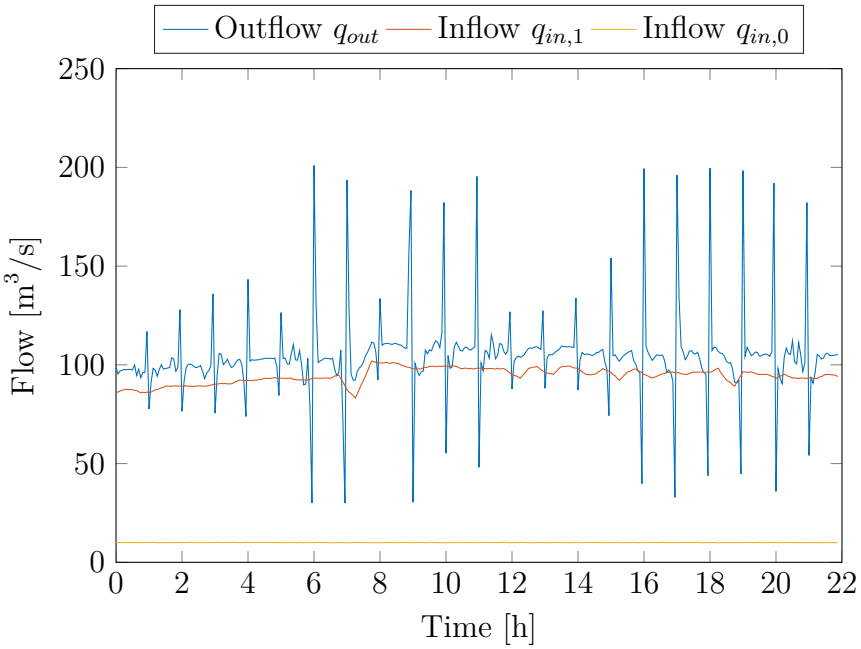


Figure 2.15.: Visualization of the external flows obtained by APCSCP A5 for the example described in section 2.3.2.

The error added to $Q_{in,1}$ is 1 percent of the predicted flow every 15 minutes. The outflow follows the global solution. Similar to the controlled flows are the parts between to price changes less smooth, see figures 2.12 and 2.14.
The calculation times in the SCP-based algorithms are significantly shorter than in the solution of the original problem, see figures 2.16 and 2.17 and table 2.1. These results are comparable to the results in [19, Table 5.1]. In this implementation, the runtime of PCSCP is better, as the derivative information of the used model can be calculated faster. If the calculation time is required to be small, APCSCP A768 is on the inside track. But in this actual example, the real-time IPOpt solver is still fast enough to be applicable and provides better results. In [19] one APCSCP iteration took 5.782% of the time compared to the solution of the full nonlinear problem. In APCSCP A768 60.341% of the total time is spent for the solution of the subproblem. The setup of the nonlinear program solver is included in this implementation, which is a contrast to [19], where the solution takes only 3.28% of the time.

Avg. calculation times compared to RT IPOpt			
RT IPOpt	PCSCP	APCSCP A5	APCSCP A768
1.553 s	0.107 s	0.087 s	0.087 s
100%	6.883%	5.597%	5.611%

Table 2.1.: Average calculation times and relative to the real-time IPOpt solver for the example described in section 2.3.2.

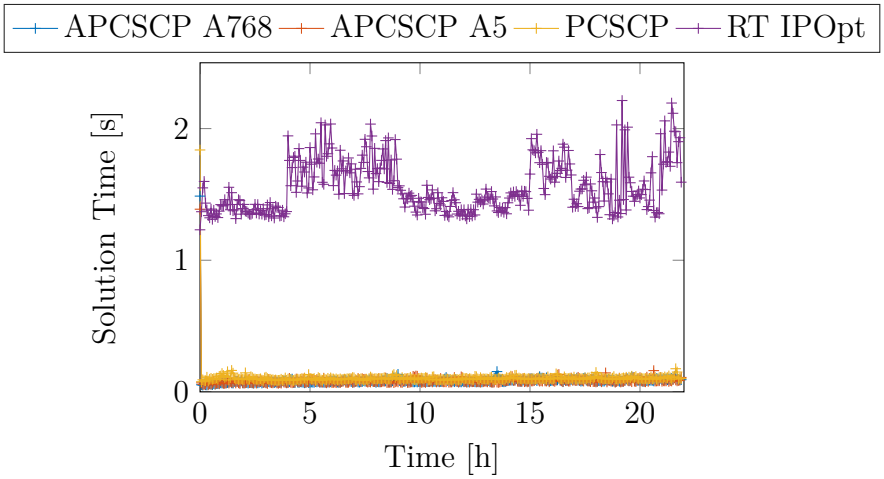


Figure 2.16.: Solution times for the example described in section 2.3.2.

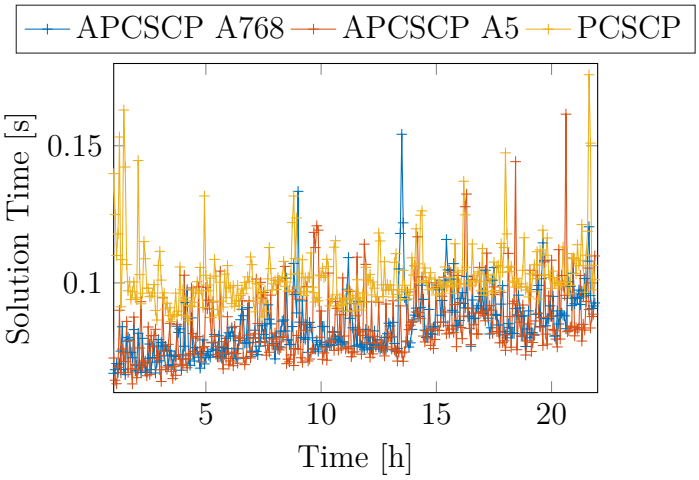


Figure 2.17.: Details of the solution times for the example described in section 2.3.2.

The SCP-based algorithms approximatively solve the problem, see figure 2.18 and table 2.2. In the first time step, all algorithms take about 1.5 seconds to find an optimal solution. In this step, all four algorithms solve the same problem with the same solver (IPOpt). In the lower plot of figure 2.18, a more detailed view to the three SCP-based algorithms solution times is given. The PCSCP algorithm calculates an exact Jacobian in every time step. APCSCP A5 updates A in every fifth iteration. In APCSCP A768, A is only updated

Objective function evaluations				
Algorithm	Minimum	Maximum	Mean	Median
IPOpt	0.0	0.0	0.0	0.0
PCSCP	0.0	0.0785	0.0574	0.0578
APCSCP A5	0.0	0.0785	0.0574	0.0578
APCSCP A768	0.0	0.0785	0.0574	0.0578

Table 2.2.: Average goal function evaluations. In every time step the optimal goal function value is zero. The average difference between the goal function evaluations of two SCP-based results is less than 10^{-6}.

at the first iteration. This can be seen in the solution times as well. Every fifth step of APCSCP A5 is close to the PCSCP times and the other steps are close to the APCSCP A768 steps. A further evaluation can be found in table 2.1.

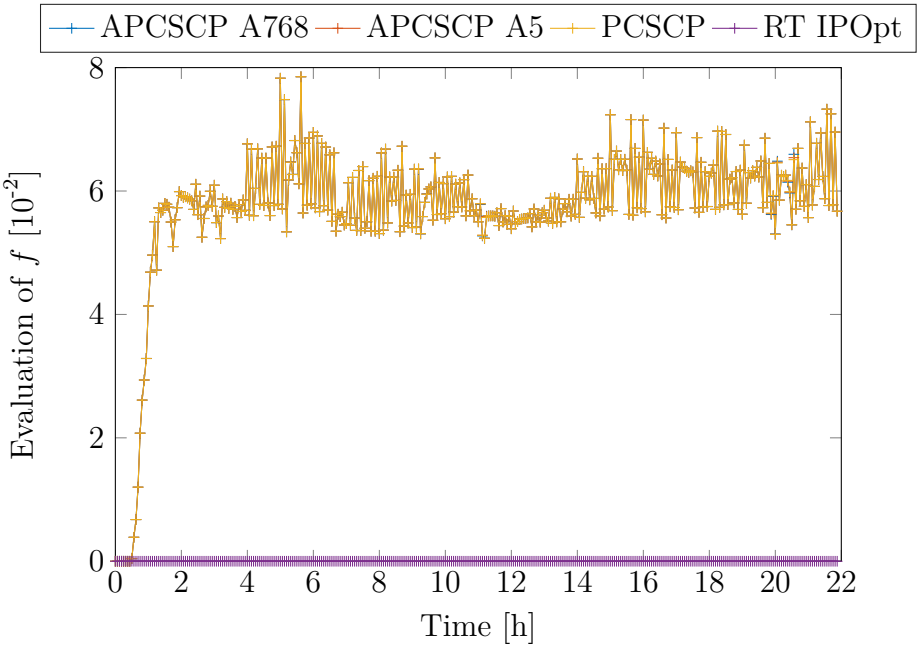


Figure 2.18.: Goal function values of the optimal solution for the example in section 2.3.2.

The approximation quality is visualized in figure 2.18 and table 2.2. As the goal function in this setting is a seminorm, the optimal value for IPOpt is 0. The real-time `IPOpt` solver solves the original problem exactly. The APCSCP curves are covered by the PCSPC curve, because the results are similar. The first goal function value is 0 in all algorithms, as the initial problem is solved exactly. The result of the contraction estimates (Theorem 1.4) is visible. There is an approximation error, but it is bounded by a constant.

Lakes				
Name	Surface [m^2]	Initial height [m]	Desired height [m]	Maximal height [m]
L_0	80,000	10	10	30
L_1	100,000	7	7	10
L_2	150,000	4	4	15
L_3	300,000	6	6	7.5
L_{3b}	300,000	6	6	7.5

Table 2.3.: Properties of the lakes visualized in figure 2.1.

Controlled Flows					
Name	Efficiency	Minimal flow [$m^3\,s^{-1}$]	Initial flow [$m^3\,s^{-1}$]	Maximal flow [$m^3\,s^{-1}$]	Height difference [m]
T_{02}	0.85	0	10	50	150
T_{12}	0.75	0	88.46	150	25
T_{23}	0.75	0	98.46	200	12
P_{10}	1.2	0	0	30	-125
q_{out}	0	30	98.46	600	0

Table 2.4.: Properties of the controlled flows as visualized in figure 2.1. The outflow can be controlled, but does not produce any power.

Duct					
Name	From	To	Max flow [$m^3\,s^{-1}$]	Parameter	Height difference [m]
U_3	L_3	L_{3b}	8	400	0

Table 2.5.: Properties of the duct as visualized in figure 2.1. A sigmoid flow is assumed.

2.3.3. Pump Storage Hydro Plant

The focus of the previous example was on the stabilization of lake heights and error compensation. Moreover, the pump was not used and the real-time iteration had no information of the power price. Therefore, the plant is not able to store and release energy to compensate for short term events. In the following example, the same model[1] as in the previous example section 2.3.2 is used, but the goal function differs. In the offline step, the goal is to keep the lower lakes L_1, L_2, L_3 and L_{3b} at the desired heights, see table 2.3. The balancing parameter $\kappa = 10^{-5}$ puts the main focus of the four lower lakes on stabilization of their heights, especially when T_{02} is completely open. Lake L_0 does not have a desired height and can be set freely to optimize the profit on the day-ahead market. In the real-time part, the horizon T is set to 1.5 [h] and the goal is to follow the obtained heights of lakes L_0, L_1, L_2 and L_3 and to optimize the profit on the intra-day market, see section 2.1.1. No external error is added. The intra-day power prices were linearly interpolated (see figure 2.19) to reduce the change in the goal function between two time steps. In reality intra-day prices are constant for 15 minutes. The day-ahead prices are the same as in the example described in section 2.3.2, see figure 2.9.

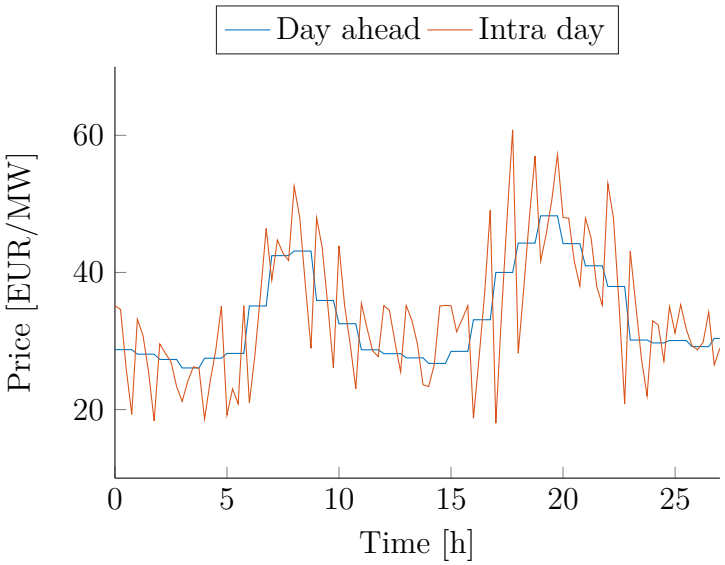


Figure 2.19.: Visualization the power prices used in section 2.3.3. The day-ahead prices are used in the offline and the intra-day prices in the real-time step.

The result of the global solution is visualized in figures 2.20 to 2.22. The result of the global solver (i.e. for 27 hours) consisted of 4747 variables with 2592 equality constraints. In contrast to the previous example (section 2.3.2), the lower lakes are almost constant. This is because the balancing parameter κ is smaller. The pump storage lake L_0 stores water and releases it, when the prices are maximal.

[1] See visualization in figure 2.1 and parameters in tables 2.3 to 2.4.

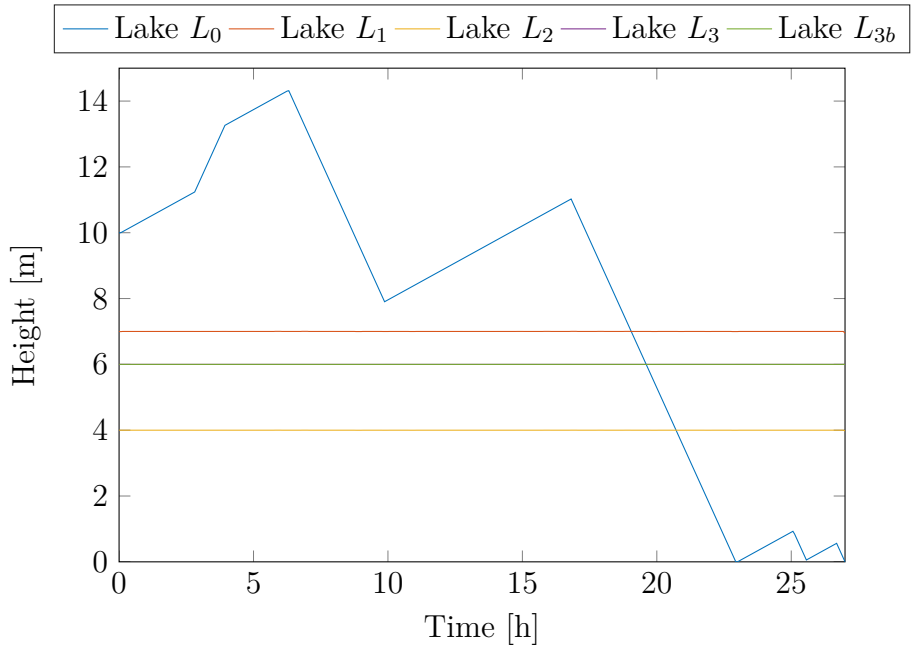


Figure 2.20.: Visualization of the optimal heights in the setting of section 2.3.3.

As the global solution knows the exact inflow data for the whole period, it is able to hold the lakes L_1, L_2, L_3 and L_{3b} exactly at the desired water level. L_0 must not have a desired height, it is completely free for profit maximization. In contrast to the previous example, it is also the only lake whose height shows a dependency to the power price. This also explains why it is emptied in the end.

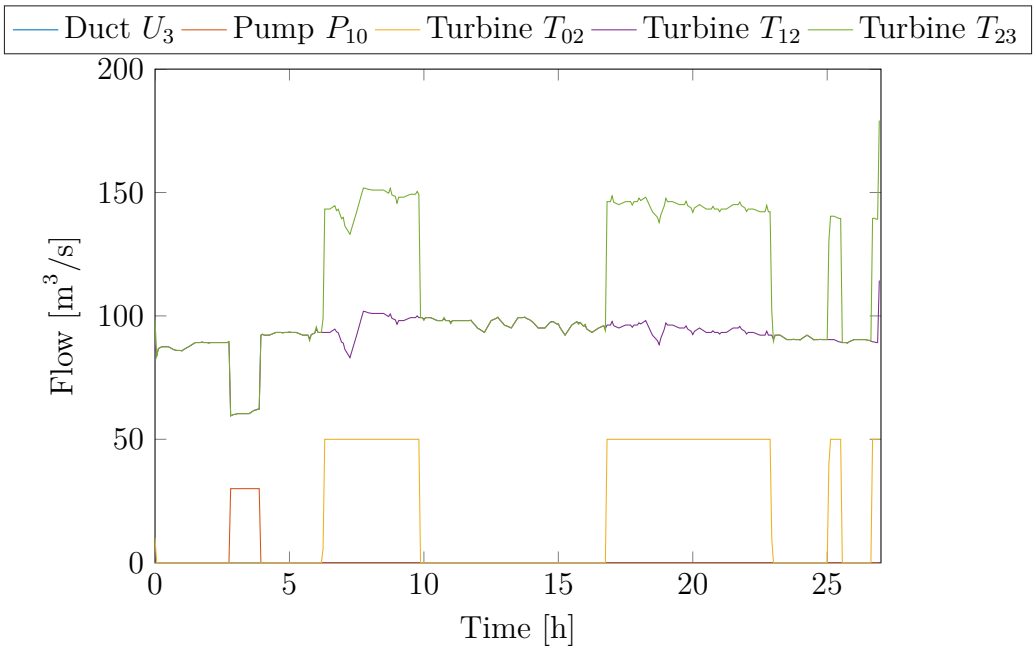


Figure 2.21.: Visualization of the duct, pump, and turbine flows the global solver in the setting of section 2.3.3.

The controlled flows have two different patterns. T_{12} and T_{23} are chosen, such that the total flow in L_1 resp. L_2 is zero. Pump P_{10} and T_{02} operate always on their lower or upper bound. Although P_{10} and T_{02} have a high efficiency (see table 2.4), the operation of the pump is in this setting only profitable between hour 3 and 4. The external flows are visualized in figure 2.22.

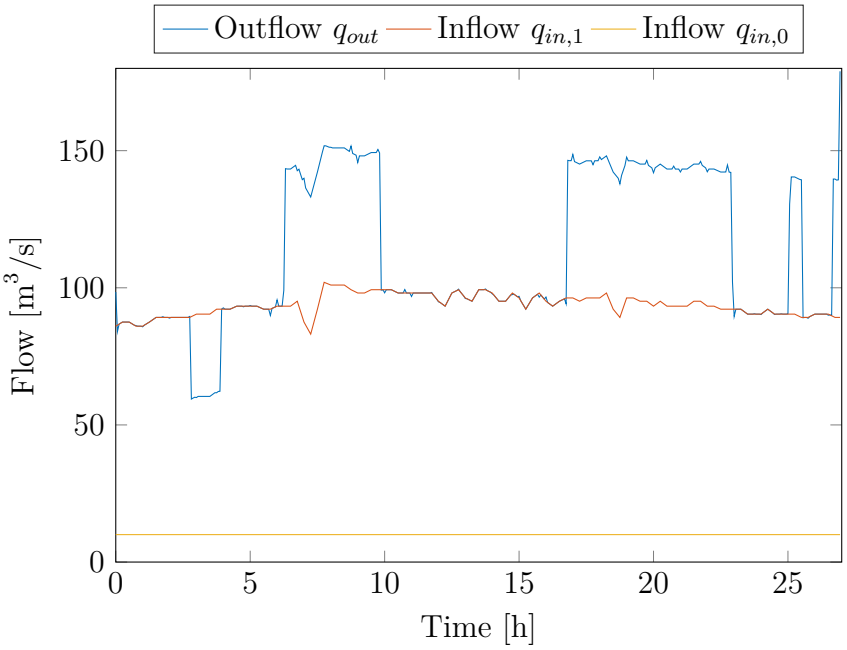
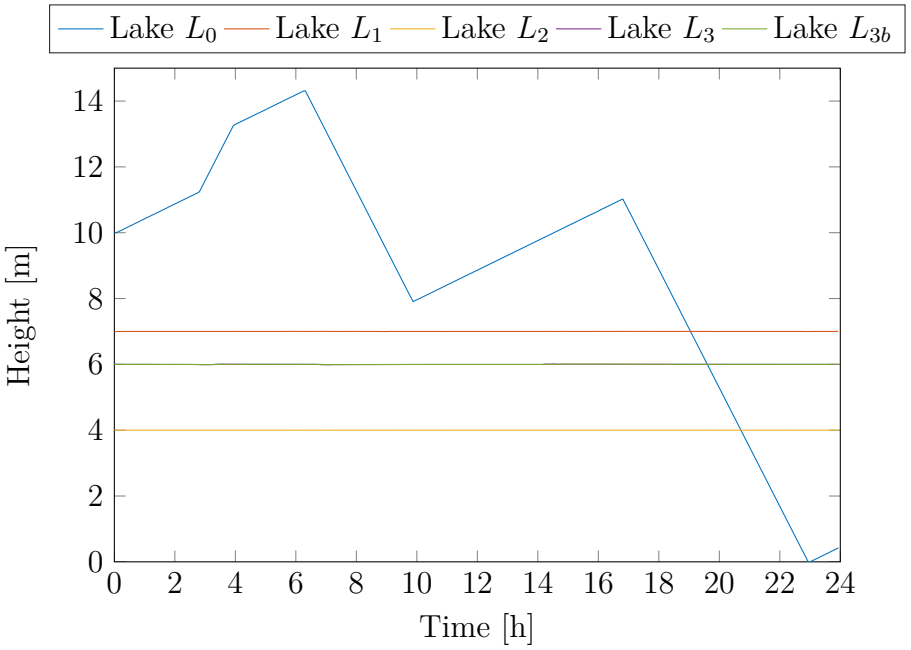


Figure 2.22.: Visualization of the global optimal external flows using the setting of section 2.3.3.

The flow q_{out} is controllable and follows T_{23} to keep the total flow of L_3 at 0. This keeps L_3 and L_{3b} at their desired height.
Like in the previous example, the results of the all real-time solvers are similar. In the following, the result of the APCSCP solver with no further update of A (APCSCP A864) is discussed in detail. Each subproblem consisted of 171 variables with 96 equality constraints. The lake heights are visualized in figure 2.23. We see, that the real-time solvers are able to keep the lakes close to the precomputed optimal heights.



.

Figure 2.23.: Visualization of the heights obtained by APCSCP A768 in the setting of section 2.3.3.

The flows of duct, pump and turbines are visualized in figure 2.24. They also follow the results in the global offline solution. Note that the flow of the duct is increased. This can be explained by the faulty inflow and the fact that in the real-time iteration neither the height of L_{3b} nor q_{U_3} is optimized.

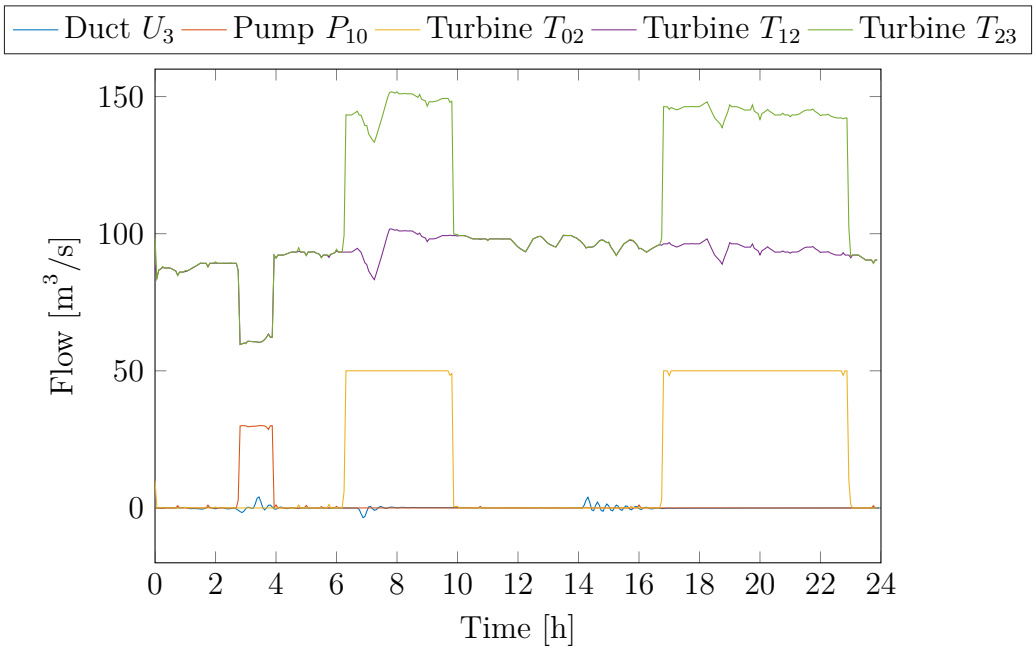


Figure 2.24.: Visualization of the duct, pump and turbine flows obtained by APCSCP A768 in the setting of section 2.3.3.

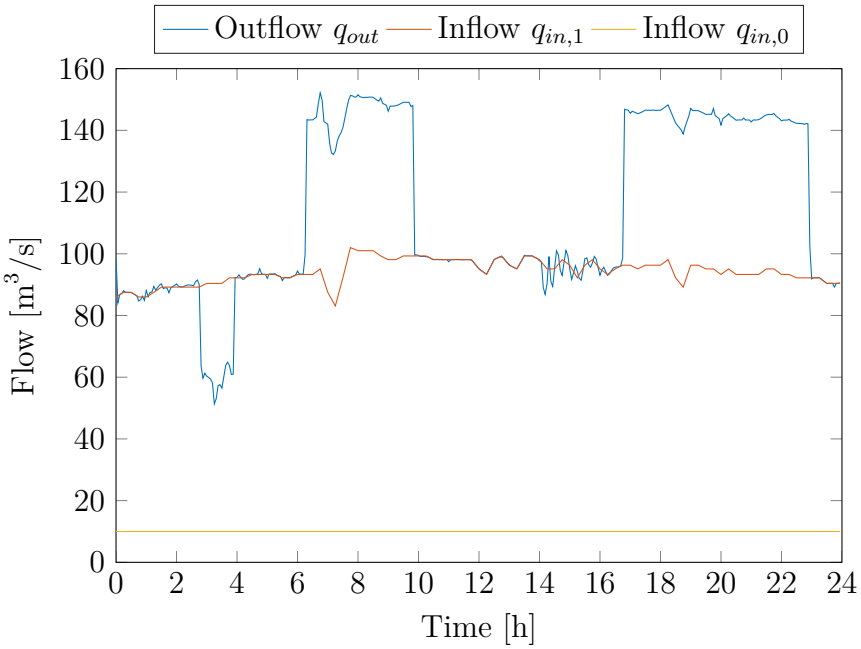


Figure 2.25.: Visualization of the external flows obtained by APCSCP A768 in the setting of section 2.3.3.

The outflow follows the flow of T_{23}. In contrast to the global solution, the flows in the duct U_3 is increased. The other flows follow the global solution. As in the previous example (section 2.3.2). The small deviations from the the precomputed flows can be explained by the changed power price and the increased duct flow, which is compensated with $q_{out,1}$ to keep the height of lake L_3 at the desired level.

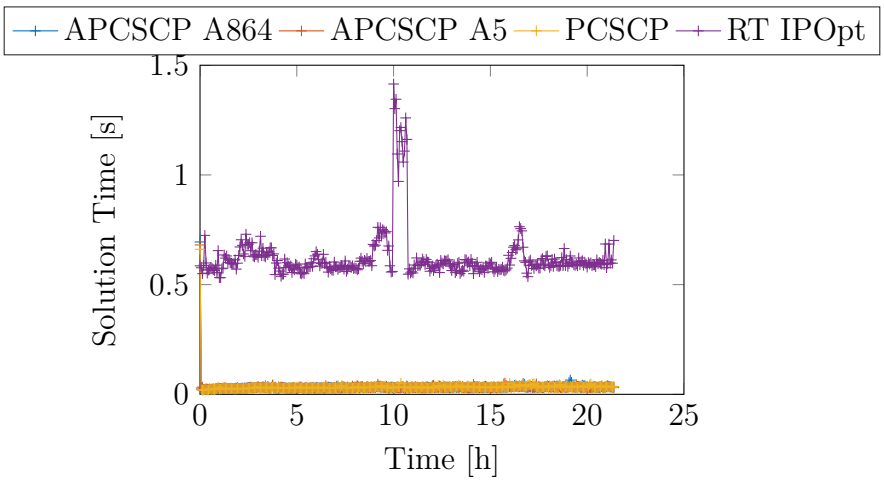


Figure 2.26.: Solution times for the model described in section 2.3.3.

Avg. calculation times compared to IPOpt			
IPOpt	PCSCP	APCSCP A5	APCSCP A768
0.6244 s	0.0338 s	0.0332 s	0.0336 s
100%	5.42%	5.32%	0.539%

Table 2.6.: Average calculation times and relative to the IPOpt solver in the setting of section 2.3.3.

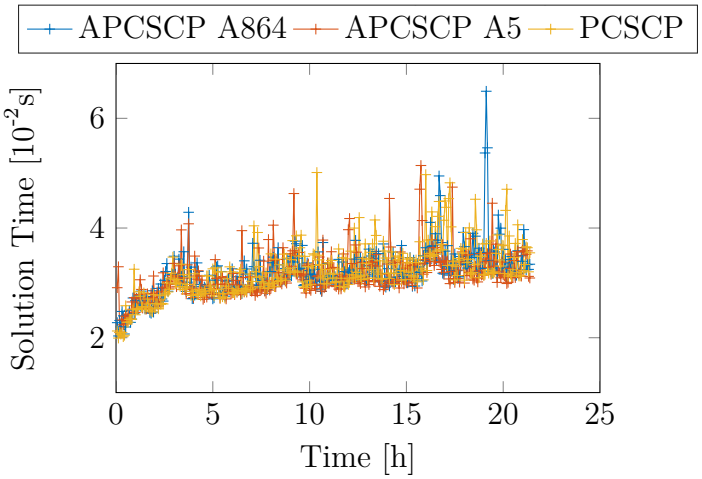


Figure 2.27.: Details of the solution times for the model described in section 2.3.3.

The evaluation times are compared in figures 2.26 and 2.27 and table 2.6 and we see that the SCP-based solvers are faster than IPOpt. Although the solution time for IPOpt always less than the time step $\Delta t = 225\,\mathrm{s}$ some subproblems can not be solved within 500 iterations. For the performance analysis, only the results before exceeding the first maximum iteration were used. In contrast to the previous example, the solution times of APCSCP and PCSCP are on the same level. After L_0 is emptied for the first time, the solution time of IPOpt increases shortly.

Objective function evaluations				
Algorithm	Minimum	Maximum	Mean	Median
IPOpt	-0.0013	-0.0001	-0.0004	-0.0004
PCSCP	0.0	0.0477	0.0238	0.0245
APCSCP A5	0.0	0.0477	0.0238	0.0245
APCSCP A768	0.0	0.0477	0.0238	0.0245

Table 2.7.: Average goal function evaluations in the pump storage setting described in section 2.3.3.

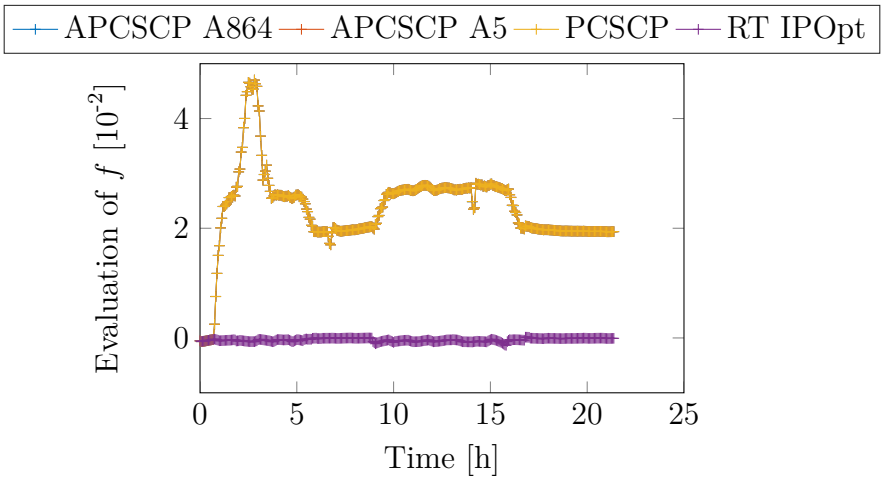


Figure 2.28.: Goal function values of the optimal solution of the example in section 2.3.3.

The objective function evaluations are visualized in figure 2.28. As expected, the SCP solvers are not optimal but stay close to the exact solution. The lowest level is achieved when turbine T_{02} is open. As the actual flow is larger than predicted, it is possible to produce more power and the optimal goal function value is negative. The APCSCP curves are covered by the PCSCP curve. The average distance between the goal function evaluations of two results obtained by a SCP-based solver is less than 10^{-5}.

2.3.4. Summary

For stabilization tasks, as described in section 2.3.2, the SCP-based solvers significantly reduce the computational cost, compensate for measurement and model errors, and the approximation error is acceptable. If the goal function in the real-time iteration is more complex, as simulated in section 2.3.3, the solution is possible, but measurement errors easily lead to a premature termination. Furthermore, also the intra-day prices had to be linearly interpolated to reduce the change in the goal function.
In order to increase the stability and robustness of the solvers, the step size can be reduced if the average solution time is smaller than Δt, like in this example. In a real-world application, a trade-off between the goal function in section 2.3.2 and section 2.3.3 should be made to balance the optimization of the profit and the stabilization of the storage lake L_0.

3. Automated Drive

3.1. Model

A survey conducted by the U.S. National Highway Traffic Safety Administration [37] from 2005 to 2007 found out that in about 94% of the investigated crashes, the critical reason, which is the last event in the crash causal chain, was assigned to the driver. In this category, the main reasons were recognition and decision errors. This measure was not intended to be interpreted as the cause of the crash or the fault of the driver. However, an older study from the late 1970s [39] came to the conclusion, that more than 90% of accidents are caused by humans. The main causes were improper lookout, speeding, improper evasive action and internal distraction. Therefore, cars with assisted or autonomous driving may help to lower the risk of human faults. Aside from the safety aspects, the development of assisted driving features satisfies the needs of customers. The daily use of a car fully requires the driver's attention. With the development of area-covering internet connection there is an increasing number of possibilities to spend this attention on more interesting things than being stuck in a traffic jam or parking manoeuvres.
In the following, a traffic jam assistant on a straight road is modeled as a receding horizon control. The safety-related requirements for a traffic jam pilot have three components. Firstly, it must be ensured that a collision with the predecessor can be avoided at any time. Secondly, it must be ensured that the vehicle stays in its lane, and thirdly, it must be ensured, that the driver has enough time to take over control. These requirements must be fulfilled in every time step and are therefore modeled as constraints. The goal function gives an optimal trajectory for the system, but, as in many autonomous drive applications, being close to an optimal solution is sufficient. For instance, if the distance between the predecessor is 5 cm smaller than the optimal distance, the result is suboptimal but, as long as the minimum safety distance holds, good enough. Therefore, the main disadvantage of approximative solvers such as APCSCP and PCSCP is not valid here.
In the hydro model, the optimal control of a turbine had two different patterns: following the inflow to keep the preceding lake stable, or operating at one of its bounds to dam water up or produce as much energy as possible. Especially when the focus was more on profit optimization than on height stabilization, this lead to problems for all real-time solvers. The duct brought in cyclic dependencies, which also caused numerical problems. When more than one duct is part of the system the duct flow must be heavily damped in the goal function, otherwise all real-time solvers diverge. As we saw before in section 2.3.2, the approximative solvers were more robust regarding measurement errors when the goal function had a semi-norm-like structure.
As in a traffic jam the car velocities are by definition small, the single track model [27, p. 722 ff.], sometimes also called bicycle or simple car model, is a good approximation. The behaviour of the car is simplified to the behaviour of a single track vehicle. Effects caused by powertrain, rear axis, or wheelspin are neglected. In [32], the bicycle model is used

for lane change and return manoeuvres. Furthermore, they show that this model leads to a control behaving like a human driver. In [1], a single track model is used for online trajectory replanning in a trajectory tracking task. [20] shows that the simple model fails in the path-following task on high speeds at an obstacle-avoidance manoeuvre task, but for smaller velocities it is an attractive alternative to more complex but computational intensive models. Additionally, it is possible to extend the single track model to solve more complex tasks. If the single track model is extended with a model of tire forces, it can be used to perform obstacle avoidance on snowy low curvature roads [40]. The single track model with nonlinear tire forces, neglected load transfer and coupled lateral and longitudinal slip allows to operate RC cars in a race setting, i.e. at friction and velocity limits, see [28]. In the following, the single track model form [27] is explained and extended with a suitable goal function and the safety distance formulation.

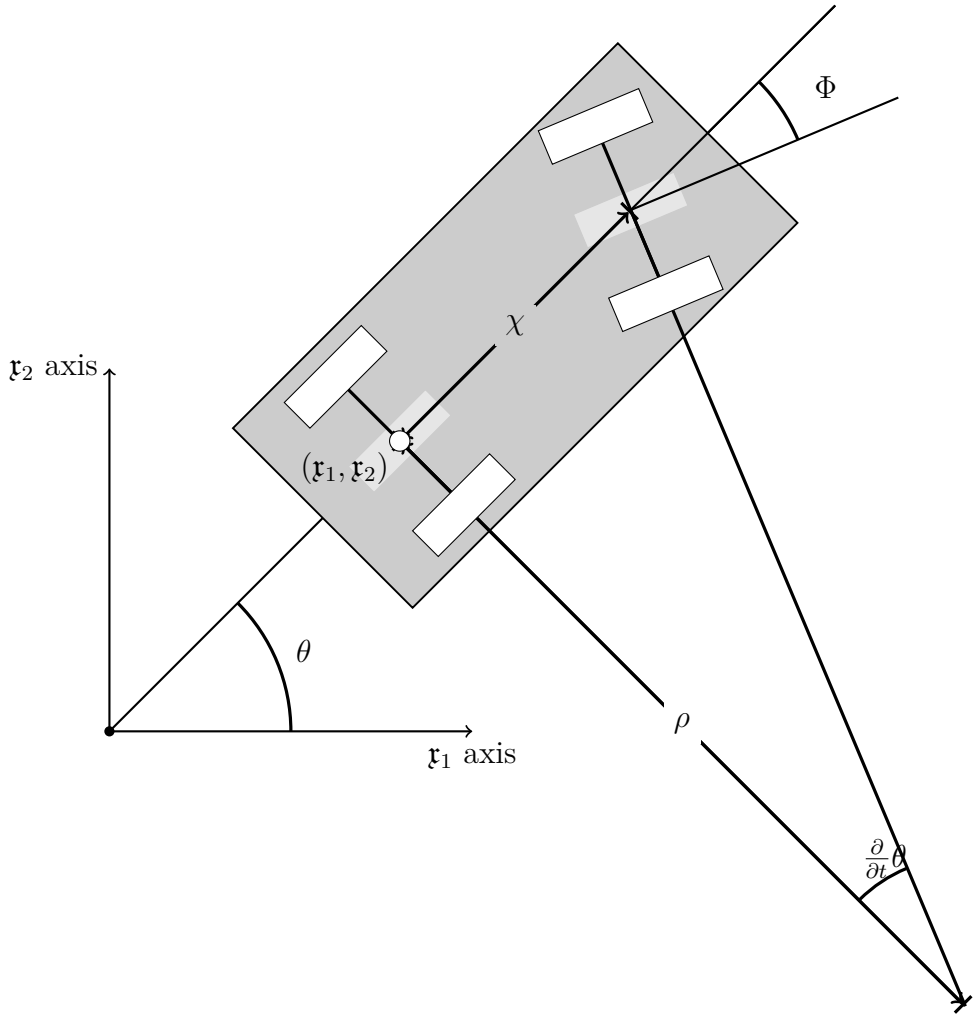


Figure 3.1.: Visualization of the simple car model. To give the impression of a car, there are two tires drawn per axis. The model, however, assumes that there is only a narrow tire per axis in the center. This is indicated with the brighter areas. Note that the car width has no effect on the behaviour of the model.

3.1.1. Components

In the following, the car *position* $(\mathfrak{x}_1, \mathfrak{x}_2)$ is the center of the rear axis, see figure 3.1. The road is straight and follows the $\mathfrak{x}_1$ axis. The *orientation* of the car θ is the angle between the $\mathfrak{x}_1$ axis and the direction the car would drive if there was no steering. The distance between front and rear axis is called *wheelbase* χ. The *velocity* is v. The *acceleration* a and the *steering* Φ are the controllable variables. In this model, the state of a car at time step k is visualized in figure 3.1 and can be described as:

$$C_k = (\mathfrak{x}_{1k}, \mathfrak{x}_{2k}, \theta_k, v_k, \Phi_k, a_k)^T \in \mathbb{R}^6. \tag{3.1}$$

As new innovations typically enter the market in the top-of-the-range cars of premium manufacturers, a BMW 7 Series sedan [25] on a German motorway was chosen as reference to determine specific values.

Acceleration

The acceleration can be set freely in every time step. In the implementation, the maximum acceleration a_{max} was set to $3.4\,[\mathrm{m/s^2}]$ and a_{min} to $-4.5\,[\mathrm{m/s^2}]$, see [25].

Velocity

The time derivative of the velocity is the acceleration:

$$\frac{\partial}{\partial t} v = a. \tag{3.2}$$

As this model is only suitable for slow velocities, v_{max} was set to $30\,[\mathrm{km/h}] = 8.3333\,[\mathrm{m/s}]$. Driving backwards is not allowed on motorways, therefore v_{min} was set to $0\,[\mathrm{m/s}]$.

Steering

It is assumed that the steering angle can be set freely in every time step. In the implementation, the maximum steering angles $\Phi_{min,max}$ were set to $\pm\frac{\pi}{8}$. A regular car has a larger maximum steering angle, but in the straight line setting this choice is sufficient and makes the assumption more plausible.

Orientation

In a small time interval, the orientation changes according to the segment covered on the turning circle. The dependency between turning radius ρ and the steering angle $\Phi \neq 0$ is visualized in figure 3.1 and given as:

$$\rho = \frac{\chi}{\tan \Phi}. \tag{3.3}$$

The orientation change over time is assumed to be :

$$\frac{\partial}{\partial t} \theta = \frac{\tan \Phi}{\chi} v. \tag{3.4}$$

The larger the turning radius, the smaller the orientation change. The faster we drive on a constant turning cycle, the faster the orientation changes. If $\Phi \to 0$, the turning direction change approaches 0 as well and ρ diverges to $\pm\infty$. No steering means driving on a straight line in the direction of the current orientation, which can be interpreted as driving on an circle with infinite radius. The maximum orientation angles $\theta_{min,max}$ were set to $\pm\frac{\pi}{8}$. The length of the wheelbase in the reference car is 3.21 [m].

Position

The directions a car can move in depend on its current position, because the rear axis cannot move sideways. This is especially a problem in parking situations, but the dependency of the car's orientation must be modeled in the lane assistant situation as well. For a small time interval, the car position moves approximately in the direction the rear wheels are pointing at. This means that:

$$\frac{\frac{\partial}{\partial t}\mathfrak{x}_2}{\frac{\partial}{\partial t}\mathfrak{x}_1} = \tan\theta = \frac{\sin\theta}{\cos\theta} \tag{3.5}$$

holds. Note that in the chosen setting, $\frac{\partial}{\partial t}\mathfrak{x}_1$ is always larger than 0 as the street follows this axis and turns will lead to a violation of the lane keeping constraint. Therefore, the orientation is restricted to $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$. The velocity of the length is the total position change, i.e. $v = \|\frac{\partial}{\partial t}\mathfrak{x}_1 + \frac{\partial}{\partial t}\mathfrak{x}_2\|$, therefore the position dynamics are:

$$\begin{aligned} \frac{\partial}{\partial t}\mathfrak{x}_1 &= v\cos(\theta), \\ \frac{\partial}{\partial t}\mathfrak{x}_2 &= v\sin(\theta). \end{aligned} \tag{3.6}$$

As the road is assumed to be straight and the lane width on German motorways is 3.15 [m] and the reference car has a width of 2.16 [m] [25], the bounds in $\mathfrak{x}_2$ direction were set to

$$\mathfrak{x}_{2min,max} = \pm \underbrace{\frac{1}{2}}_{\text{safety factor}} \frac{3.15 - 2.16}{2}[\text{m}]. \tag{3.7}$$

The additional safety factor was introduced to guarantee a larger distance to the neighbouring lanes. The upper bound for $\mathfrak{x}_1$ must ensure the safety distance d_{safety} to the preceding vehicle and therefore depends on the actual velocity v_k and the measured position of the predecessor $(\widehat{\mathfrak{x}_1}, \widehat{\mathfrak{x}_2})_k$. In a *worst case* assumption, it is assumed that the predecessor will stay at this position for all upcoming time steps.

Safety distance In this thesis, the safety distance:

$$d_{safety}(v_{act}) = \underbrace{3v_{act}\Delta t}_{\text{reaction distance}} + \underbrace{\frac{1}{2}\frac{v_{act}^2}{|a_{min}|}}_{\text{breaking distance}} + \underbrace{0.2}_{\text{guaranteed distance}} [\text{m}] \tag{3.8}$$

consists of three parts. The first one is the distance travelled until the braking of the predecessor is recognized. In the following, it is assumed that at most 3 time steps are

necessary for the system to detect that the preceding vehicle is braking. In this time, the current velocity v_{act} is not reduced. The second part is the braking distance. It is assumed, that the brakes provide a constant deceleration in the whole braking process. The third part is an additional safety distance of 0.2 m to compensate further measurement errors and ensure a minimum when the velocity is close to zero.

3.1.2. Goal Function

In this thesis, the goal function was designed such that a traffic jam assistant satisfies the following goals:

1. *Following the predecessor* in an adequate distance.
2. *Cancelling errors.* Engine, steering and sensors do not work precisely and this model is also approximative. The obtained error should not sum up over time.
3. *Driving comfortably.* To ensure a pleasant and smooth ride for the passengers the lateral and longitudinal accelerations should be reduced.
4. *Reducing driver interaction.* The problem should be fast enough to solve in as many consecutive time steps as possible.

The goal function at time point k measured in the global time frame is chosen as

$$f_k(x) = \sqrt{\sum_{j=k}^{k+T} \sum_{x_i \in C_j} \left(w_i s_i (x_i - \iota_i(x_i))\right)^2} \tag{3.9}$$

with $w, \iota \in \mathbb{R}^{6(T+1)}, w \geq 0, s \in \{0,1\}^{6(T+1)}$ and $s_i = 0 \Rightarrow \iota = 0$.

Remark 3.1. Note that $f_k(x)$ can also be written in the form

$$\left\| \mathrm{diag}(w)\mathrm{diag}(s)\left(x - \iota(x)\right)\right\|. \tag{3.10}$$

The goal function can be seen as a distance to an optimal point. If ι does not depend on x, f_k would be a semi norm.

The parameters w, s and ι were chosen to meet the above mentioned requirements in 3.1.2.

1. *Following the predecessor* in an adequate distance. This is a requirement on the $\mathfrak{x}_1$ position. The optimal position for the car depends on position and velocity of the predecessor just like the maximal position. In the numerical experiments the optimal position is the double safety distance behind the predecessor position. In this case, other cars can switch safely into the lane. The parameters corresponding to $\mathfrak{x}_1$ values are set to

$$s = 1, w = 1 \text{ and } \iota = 2d_{safety}(v_k). \tag{3.11}$$

2. *Cancelling errors.* The car should always drive in the middle of the road. As the distances in the $\mathfrak{x}_2$ direction are relatively smaller the corresponding parameters are set to

$$s = 1, w = 20 \text{ and } \iota = 0. \tag{3.12}$$

3. *Driving comfortably.* The reduction of steering improves the ride's smoothness, but the system must be able to stop safely at any time. At time point k, the next control for the upcoming interval is calculated. By minimizing the steering only for the next time step $(k+1)$, the smoothness of the ride is ensured and in every time step a new emergency braking control is calculated. The weight corresponding to $\Phi(t_{k+1})$ is set to 3 and all other weights are set 0.

4. *Reducing driver interaction.* The control for the next time step must be calculated prior this time step. To achieve this, the horizon T must be selected carefully.

3.1.3. Errors

In every second a random uniformly distributed error is added to the $\mathfrak{x}_1$- and $\mathfrak{x}_2$-position. The values were:

Component	Max. Error
$\mathfrak{x}_1$ position	20 [cm]
$\mathfrak{x}_2$ position	2 [cm]

If there is more than one step per second, the error added is proportional to the time interval.

3.2. Numerical Results

3.2.1. Traffic Jam Assistant

In this example the car has to follow a predecessor driving with the following velocities, see figure 3.2.

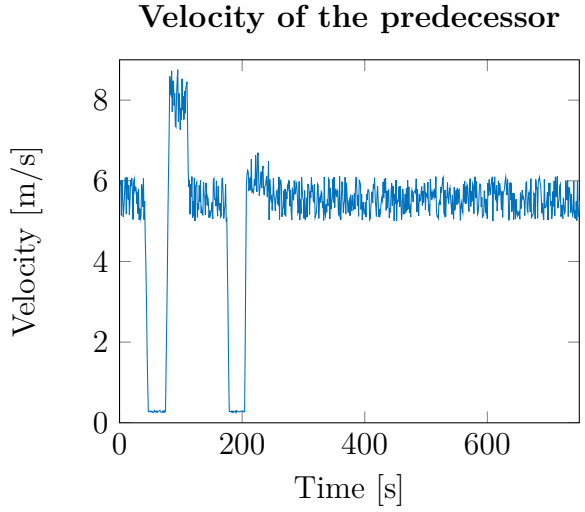


Figure 3.2.: Visualization of the preceding vehicle's velocity. In the beginning, there are two stop and go phases, afterwards the longtime behaviour is tested.

The goal function and its parameters were discussed in section 3.1.2. As in [31], the driver's take-over time was measured between 10 to 15 s when the car drove faster than 100 km h^{-1}, the horizon was set to $T = 12$ [s] and the step size to $\Delta t = 0.5$ [s]. The position of the preceding vehicle is not known in advance. In every time step, the sensor set provides a new position of the preceding car. The simulated time is 750 s, therefore 1500 subproblems must be solved. In this setting, the offline calculation of an global solution with predicted data is not sensible. As the board computers do not have the necessary computational power, there is not enough time and an adequate prediction method for the behaviour of the predecessor for a longer time span does not exist.
Similar to the experiments in chapter 2, the tested solvers were real-time IPOpt, PCSCP and APCSCP with $\widetilde{H} = 0$ and for APCSCP various update steps for an exact calculation of A were used. In the following the results of APCSCP A3, are discussed. Here the A is calculated in every third step. Like in chapter 2, the RT IPOpt had some problems solving all subproblems in less than 200 iterations. In step 409 (i.e. $t = 204.5$ [s]), this occurs for the first time. In the following figures and the performance analysis, the result of this solver is only ploted to this time step. The SCP-based were able to simulate the whole time interval.

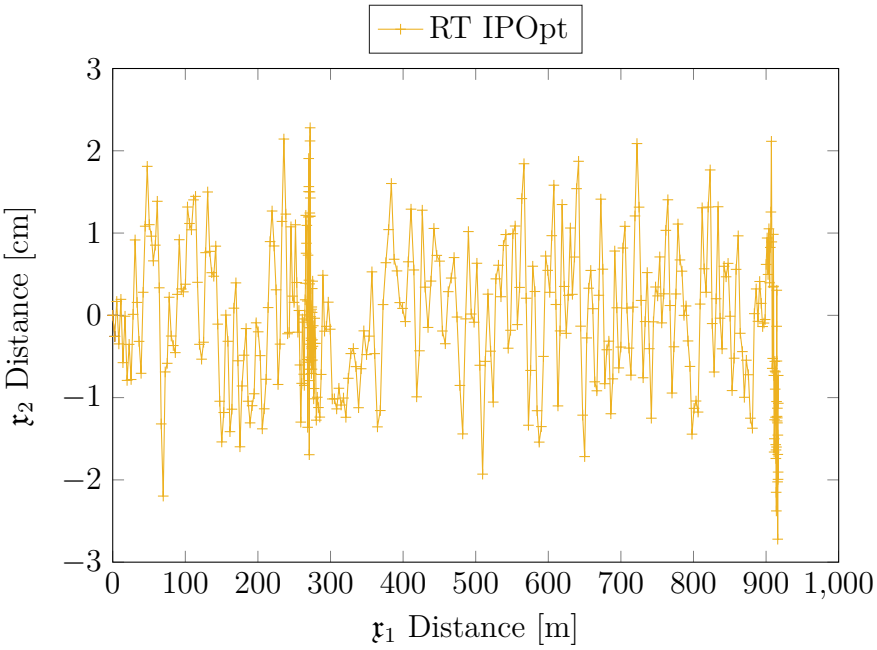


Figure 3.3.: Visualization of the route obtained by real-time IPOpt.

In figures 3.3 and 3.4 the $\mathfrak{x}_1$ and $\mathfrak{x}_2$ positions of the RT IPOpt and APCSCP A3 are visualized for the first 409 time steps. Both solvers are able to keep the distance to the lane center less than 3 cm. Both solvers obtain safe trajectories with marginal differences.

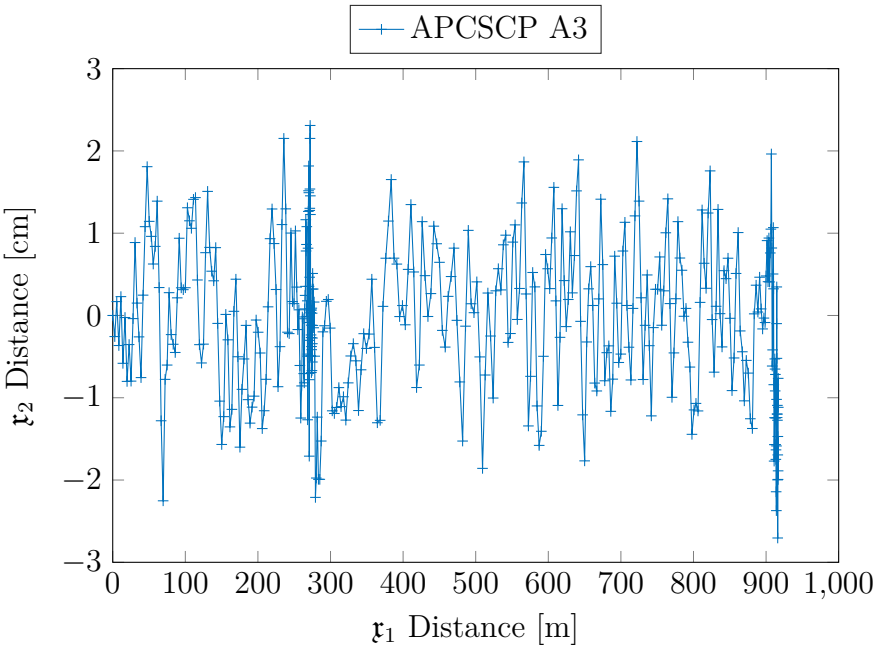


Figure 3.4.: Visualization of the route obtained by APCSCP A3.

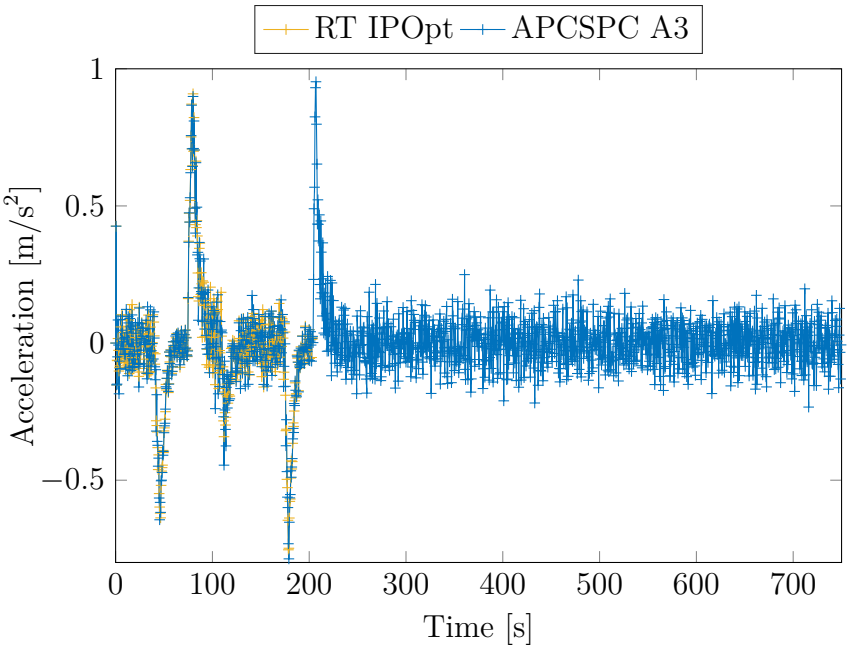


Figure 3.5.: Comparison of the obtained accelerations a.

In figure 3.5 the car's acceleration is visualized. The car's velocity, which depends on the acceleration is visualized in figure 3.6. Both solvers are able to keep the accelerations less than $\pm 1\,[\mathrm{m/s^2}]$. In constant velocity phases, the absolute acceleration is less than $0.3\,[\mathrm{m/s^2}]$. Small braking and accelerating manoeuvers compensate the error in $\mathfrak{x}_1$ direction and larger accelerations only occur when it is required by the behaviour of the predecessor. If the absolute value of the acceleration is close to zero, the quality of the ride is ensured.

Furthermore, has the car in every time step a control for the next T seconds. This control is an emergency braking manoeuvre, and is executed, if, for instance, the next subproblem of the real-time iteration is not solvable. Note that the velocity profile is smoother compared to the predecessor, see figure 3.2.

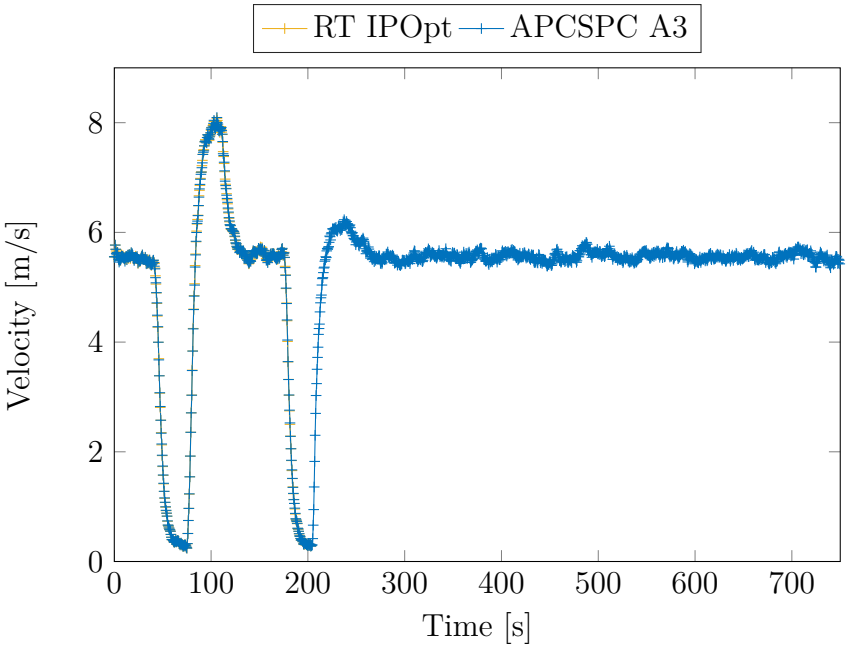


Figure 3.6.: Comparison of the obtained velocities v.

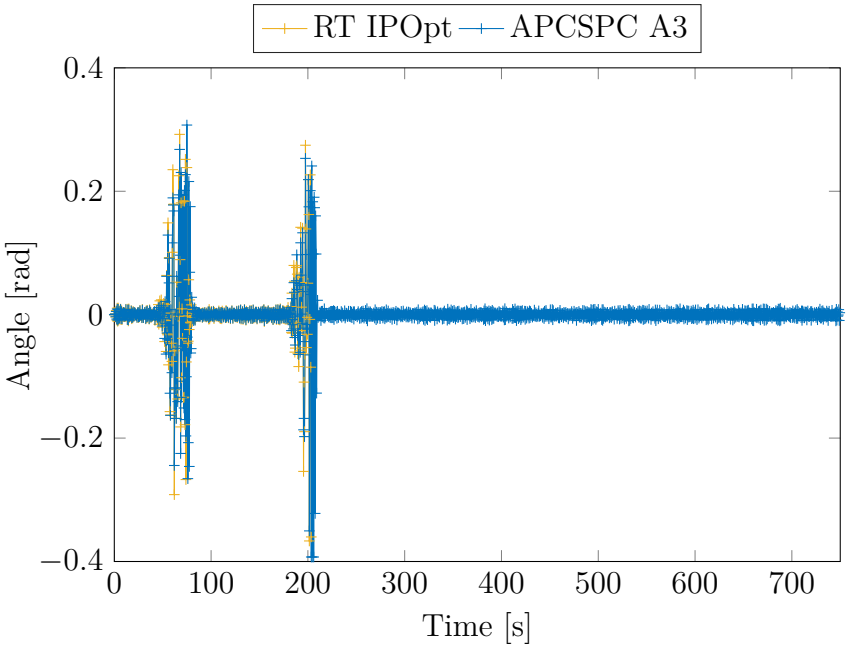


Figure 3.7.: Comparison of the obtained steering angles Φ.

Figure 3.7 visualizes the car's steering angle. The orientation of the car (see figure 3.8)

depends on the steering Φ. The change in the orientation is always less than 5° and in constant velocity phases it is less than 0.5°. In the low velocity phases, the orientation and steering angles differ the most from the optimal value 0. When the car drives at a low velocity, a larger steering angle is required for the same impact on the orientation, see equation (3.4). Changes in the orientation are necessary to compensate for the error in $\mathfrak{x}_2$ direction, which does not depend on the velocity.

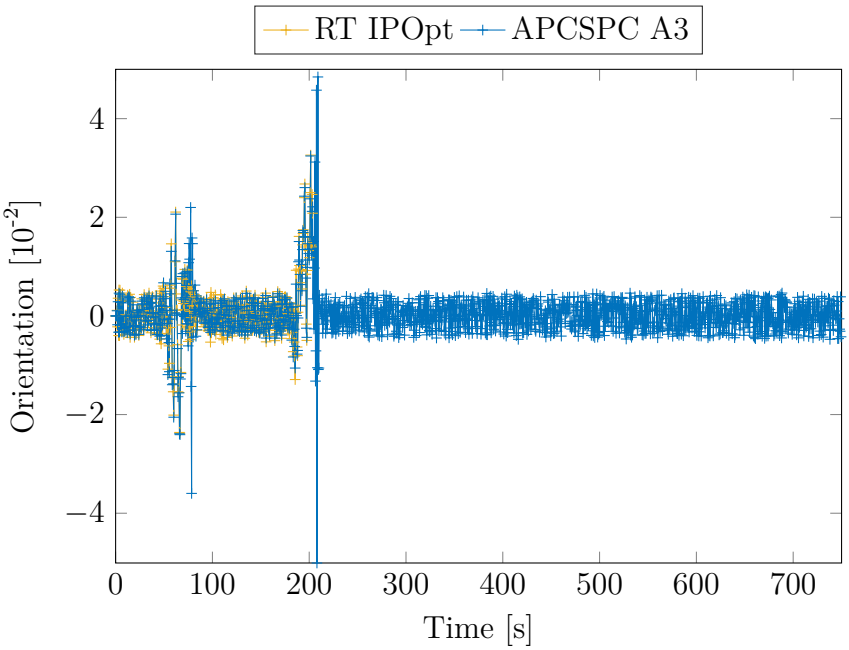


Figure 3.8.: Comparison of the obtained orientation θ.

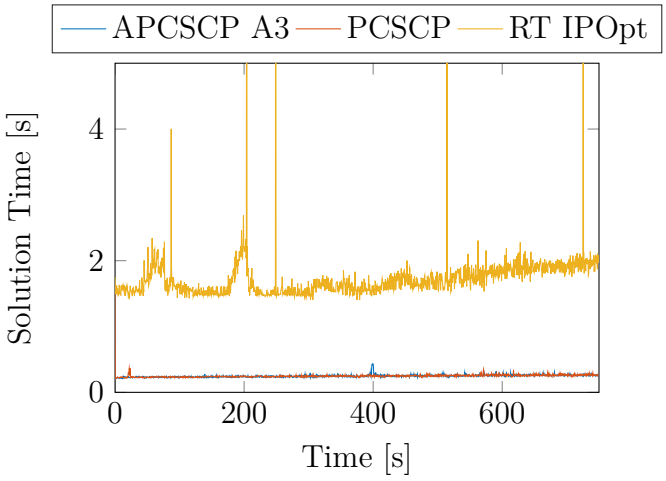


Figure 3.9.: Solution times for the model described in section 3.2.1.

Calculation times compared to IPOpt		
RT IPOpt	PCSCP	APCSCP A3
1.665 s	0.2341 s	0.2324 s
100%	14.06%	13.96%

Table 3.1.: Average absolute calculation times and relative to the IPOpt solver in the setting of section 3.2.1.

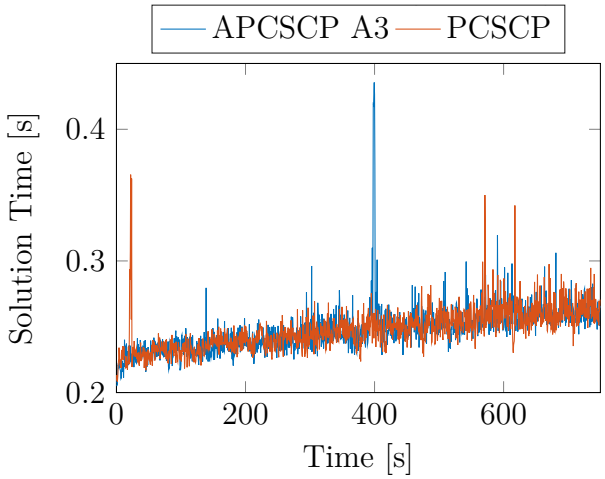


Figure 3.10.: Details of the solution times for the model described in section 3.2.1.

The timing properties of the tested solvers are visualized in figures 3.9 and 3.10 and table 3.1. The RT IPOpt outliers are cut of; the maximum solution duration there is around 15 [s]. The solution times of the SCP-based solvers are always less than $0.5\,[\mathrm{s}] = \Delta t$ and the solution time of RT IPOpt is always greater than $1.5\,[\mathrm{s}] = 3\Delta t$. This means that the SCP-based solvers fullfil the timing requirement and can be applied in a real-world application. Without a massive performance gain this is not possible for the RT IPOpt. As A is updated in every third step, APCSCP A5 faster, but has also larger outliers.

Objective function evaluations				
Algorithm	Minimum	Maximum	Mean	Median
RT IPOpt	0.8977	75.2078	37.8219	46.0293
Diff. of PCSCP	-1.0435	0.7191	0.0412	0.0214
relative	-2.1874%	4.3665%	0.1766%	0.1895%
Diff. of APCSCP A3	-0.6796	0.8078	0.0449	0.0418
relative	-2.7028%	3.6972%	0.1875%	0.2054%

Table 3.2.: Average goal function evaluations for the example described in section 3.2.1. The line "Diff. of PCSCP" means difference of PCSCP to the exact solution obtained by RT IPOpt.

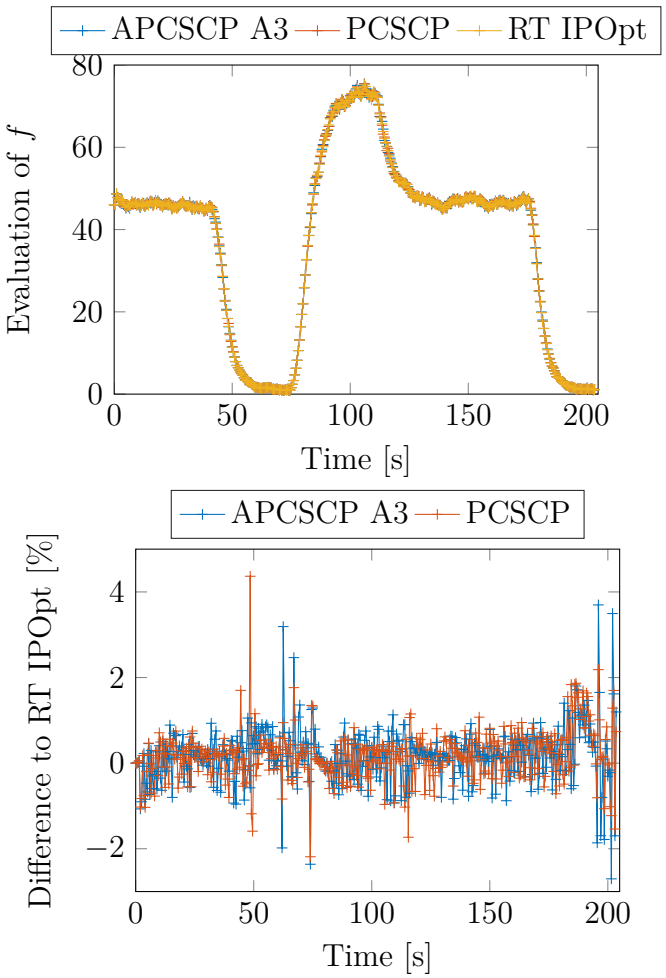


Figure 3.11.: Goal function evaluations for the model described in section 3.2.1.

The approximation quality of the tested SCP-based solvers is visualized in figure 3.11 and table 3.2. As the upper plot in figure 3.11 shows, have all tested solvers similar goal function evaluations. The relative approximation error of the SCP-based solvers is visualized in the lower plot pf figure 3.11. The approximation error is always less than 5% and in constant velocity phases, it is smaller than 2%. As discussed before, the occupants

will hardly notice a difference, see figures 3.5 to 3.8. Note that low velocity phases are associated with a lower goal function value, as the car velocity is the main dependency of the safety distance, see equation (3.8).

3.2.2. Parallelization

Section 1.1.3 mentions that the multiple shooting procedure can be parallelized. Since several calculations are done simultaneously, the data in shared variables must be protected from corruption such as overwriting. This protection may lead to performance reduction or to increased memory consumption. The number of possible threads must therefore be chosen carefully, as an increased number of threads does not automatically lead to faster results. In the following experiment, the first 200 seconds from the traffic jam assistant example described in section 3.2.1 ran with a varying number of possible threads. To implement the parallelization the GNU Offloading and Multi Processing Runtime Library including an implementation of the OpenMP API (`libgomp`) was used. The code ran on Linux Mint 17.3 equipped with the Intel i7-4700MQ CPU offering 4 cores at 2.4 GHz, 8 threads and having 6 MiB cache and 8 GiB DDR3 RAM. The used solver is PCSCP. All runs produced exactly the same controls and states but with different timing properties, see figures 3.12 and 3.13. Besides the operation system, no further applications were running, that may have allocated CPU time. All programs were executed sequentially starting with the single thread version and with a pause between two programs to reduce the effects of overheating.

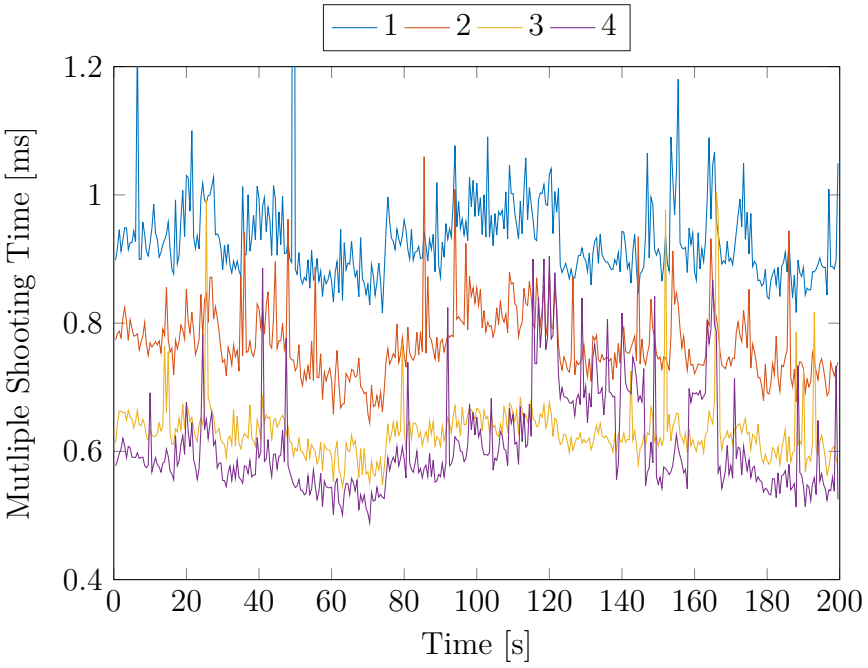


Figure 3.12.: Visualization of the solution times for the multiple shooting procedure run with different number of threads.

The results are visualized in figures 3.12 and 3.13. As one might expect, the version with four threads is the fastest. If the thread number is larger than the number of physical

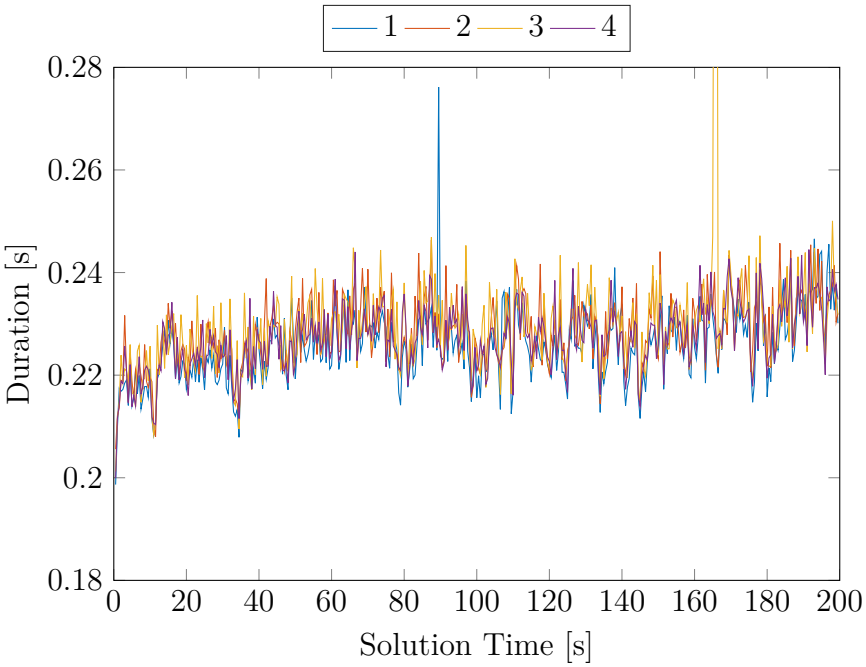


Figure 3.13.: Visualization of the durations for one iteration of the PCSCP solver.

cores, the performance is reduced. In this case threads share one core and this leads to a larger number of memory operations. The run with 3 threads is slightly slower, but one core is free for other background tasks (e.g. of the operating system). The solution of the subproblem is about 25 times more time consuming, than the integration with multiple shooting. For this system the effect of parallelized programming is very small. The outlier of thread 3 solution is at 0.40 [s]. In this thesis, the computational cost of the integration was always distinctly less than the cost of the NLP solver. Therefore, the iteration time of the real-time algorithm is not significantly decreased. If a model contains more complex differential algebraic equations and the integration becomes the bottleneck, the coding overhead will pay off.

3.2.3. Time Discretization

The choice of Δt is crucial to most receding horizon controls. If Δt is chosen too large, the subproblems can not be solved successfully, as the occurring error is too large, see figure 1.3. If Δt is chosen too small, it is not possible to solve the subproblems in less than Δt. Furthermore, the reduction of Δt leads to more difficult subproblems, as halving Δt doubles the number of variables when T is constant. To investigate the effects of different time discretizations, the traffic jam assistant described in section 3.2.1 ran with 3 different choices for Δt for the first 200 seconds. Analogously to the parallelization experiment in section 3.2.2, all other settings were not changed.
The result of the APCSCP A3 solver is visualized in table 3.3. For the choice $\Delta t = 0.5$ [s] all subproblems are solved successfully and faster than Δt. The choice $\Delta t = 1$ [s] leads to a faster solution, but the results are less smooth. If Δt is chosen as 0.25 [s]. All subproblems solve successfully, but the average solution time is considerably larger than 0.25 s and therefore not applicable in real-time. In this example APCSCP A3 is used as solver.

Time Discretization		
Δt	Avg. Duration	variables
1 [s]	0.0113 s	236
0.5 [s]	0.1131 s	472
0.25 [s]	2.3542 s	944

Table 3.3.: The effects of time discretization.

3.2.4. Summary

In contrast to the examples in chapter 2, the goal function required the controls to be as close to 0 as possible. This prevents a control to jump from the lower to the upper boundary between two time steps. This might be one of the main causes for the robostness of this system to external errors. APCSCP and PCSCP are an option for tasks in the automated driving field. They solve the subproblems considerably faster than an exact solver and ensure the feasibility of safety-related constraints. The approximation is sufficiently close to the optimum to control the system successfully and is small enough to be not noticed by the occupants.

The use of a coordinate system, which is based on the rear axis, could lead to better results, especially when the road is bendy. In this example, the orientation was set to $\pm\theta_{max} = \frac{\pi}{8}$, which makes sense when both, the $\mathfrak{x}_1$ axis and the street, point in the same direction. But if, for instance, the street has a 90° turn, the actual model is not applicable. If the single track model is extended for fast velocity drive, the use of SCP-based solvers becomes more interesting, as a more complex model leads to a more complex equality constraint. On the other hand, the model should fulfill the assumptions of APCSCP to bind the approximation error. When a safe trajectory for a more difficult driving task, like lane change or driving without a predecessor, is given, these solvers can also be used to follow that trajectory.

Bibliography

[1] Muhammad Awais Abbas, J. Mikael Eklund, and Ruth Milman. Real-time analysis for nonlinear model predictive control of autonomous vehicles. In *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4, 2012. ISBN 9781467314336. doi: 10.1109/CCECE.2012.6335014. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6335014.

[2] Bavarian Environmental Agency. Master data Gauge München/Isar. http://www.gkd.bayern.de/fluesse/abfluss/stationen/stammdaten/index.php?thema=gkd&rubrik=fluesse&produkt=abfluss&msnr=16005701&sp=en, 02 2016.

[3] Joel Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. Phd thesis, Arenberg Doctoral School, KU Leuven, Leuven, 2013. URL https://lirias.kuleuven.be/bitstream/123456789/418048/1/thesis_final2.pdf.

[4] Joel Andersson, Johan Akesson, and Moritz Diehl. Dynamic optimization with CasADi. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 681–686, Maui, Hawaii, 2012. IEEE. doi: 10.1109/CDC.2012.6426534. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6426534.

[5] Hans Georg Bock and Karl Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *9th IFAC World Congress Budapest*, pages 242–247. Pergamon Press, 1984. URL http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf.

[6] Hans Georg Bock, Moritz Diehl, Daniel B. Leineweber, and Johannes P. Schlöder. A Direct Multiple Shooting Method for Real-Time Optimization of Nonlinear DAE Processes. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, volume 26, pages 245–267. Birkhäuser Basel, Basel, 2000. ISBN 978-3-0348-8407-5. doi: 10.1007/978-3-0348-8407-5_14. URL http://link.springer.com/10.1007/978-3-0348-8407-5_14.

[7] Joseph Frédéric Bonnans and Alexander Shapiro. Optimization Problems with Perturbations: A Guided Tour. *SIAM Review*, 40(2):228–264, jan 1998. ISSN 0036-1445. doi: 10.1137/S0036144596302644. URL http://dx.doi.org/10.1137/S0036144596302644.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 7 edition, 2004. ISBN 978-0-521-83378-3. URL http://www.cambridge.org/gb/academic/subjects/statistics-probability/optimization-or-and-risk/convex-optimization?format=HB.

[9] Eduardo F. Camacho and Carlos Bordons Alba. *Model Predictive Control*. Springer, London, 2 edition, 2007. doi: 10.1007/978-0-85729-398-5. URL `http://link.springer.com/book/10.1007/978-0-85729-398-5`.

[10] Rafael Correa and Hector C. Ramirez. A Global Algorithm for Nonlinear Semidefinite Programming. *SIAM Journal on Optimization*, 15(1):303–318, jan 2004. ISSN 1052-6234. doi: 10.1137/S1052623402417298. URL `http://dx.doi.org/10.1137/S1052623402417298`.

[11] Charles Francis Curtiss and Joseph Oakland Hirschfelder. Integration of Stiff Equations. *Proceedings of the National Academy of Sciences of the United States of America*, 38(3):235–243, 1952. ISSN 0027-8424. doi: 10.1073/pnas.38.3.235. URL `http://www.pnas.org/content/38/3/235`.

[12] Frederik Debrouwere, Wannes Van Loock, Goele Pipeleers, Quoc Tran Dinh, Moritz Diehl, Joris De Schutter, and Jan Swevers. Time-Optimal Path Following for Robots With Convex-Concave Constraints Using Sequential Convex Programming. *IEEE Transactions on Robotics*, 29(6):1485–1495, dec 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2277565. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6634254`.

[13] Peter Deuflhard. *Newton Methods for Nonlinear Problems*, volume 35 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1 edition, 2011. ISBN 978-3-642-23898-7. doi: 10.1007/978-3-642-23899-4. URL `http://link.springer.com/10.1007/978-3-642-23899-4`.

[14] Moritz Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Heidelberg University Faculty of Mathematics and Computer Science, 2001. URL `http://www.ub.uni-heidelberg.de/archiv/1659`.

[15] Moritz Diehl, Hans Georg Bock, and Johannes P. Schlöder. A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005. ISSN 0363-0129. doi: 10.1137/S0363012902400713. URL `http://epubs.siam.org/doi/abs/10.1137/S0363012902400713`.

[16] Quoc Tran Dinh. *Sequential Convex Programming and Decomposition Approaches for Nonlinear Optimization*. PhD thesis, KU Leuven, 2012. URL `https://lirias.kuleuven.be/handle/123456789/359872`.

[17] Quoc Tran Dinh, Carlo Savorgnan, and Moritz Diehl. Real-time sequential convex programming for nonlinear model predictive control and application to a hydropower plant. In *IEEE Conference on Decision and Control and European Control Conference*, pages 5905–5910, Orlando, FL, dec 2011. IEEE. ISBN 978-1-61284-801-3. doi: 10.1109/CDC.2011.6160919. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6160919`.

[18] Quoc Tran Dinh, Carlo Savorgnan, and Moritz Diehl. Real-Time Sequential Convex Programming for Optimal Control Applications. *ArXiv e-prints*, page 10, 2011. URL `http://adsabs.harvard.edu/abs/2011arXiv1105.3427D`.

[19] Quoc Tran Dinh, Carlo Savorgnan, and Moritz Diehl. Adjoint-Based Predictor-Corrector Sequential Convex Programming for Parametric Nonlinear Optimization. *SIAM Journal on Optimization*, 22(4):1258–1284, 2012. ISSN 1052-6234. doi: 10.1137/110844349. URL http://epubs.siam.org/doi/abs/10.1137/110844349.

[20] Paolo Falcone, Hongtei Eric Tseng, Francesco Borrelli, Jahan Asgari, and Davor Hrovat. MPC-based yaw and lateral stabilisation via active front steering and braking. *Vehicle System Dynamics*, 46(March 2015):611–628, 2009. ISSN 0042-3114. doi: 10.1080/00423110802018297. URL http://www.tandfonline.com/doi/abs/10.1080/00423110802018297.

[21] Bassem Fares, Dominikus Noll, and Pierre Apkarian. Robust control via sequential semidefinite programming. *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, 3(6):1791–1820, 2002. ISSN 0191-2216. doi: 10.1109/CDC.2002.1184257. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1184257.

[22] Roland W. Freund and Florian Jarre. A sensitivity result for semidefinite programs. *Operations Research Letters*, 32(2):126–132, 2004. ISSN 01676377. doi: 10.1016/S0167-6377(03)00069-5. URL http://www.sciencedirect.com/science/article/pii/S0167637703000695.

[23] Roland W. Freund, Florian Jarre, and Christoph H. Vogelbusch. Nonlinear semidefinite programming: sensitivity, convergence, and an application in passive reduced-order modeling. *Mathematical Programming*, 109(2-3):581–611, 2007. ISSN 0025-5610. doi: 10.1007/s10107-006-0028-x. URL http://link.springer.com/10.1007/s10107-006-0028-x.

[24] Andreas Griewank and Andrea Walther. On constrained optimization by adjoint based quasi-Newton methods. *Optimization Methods and Software*, 17(January 2015):869–889, 2002. ISSN 1055-6788. doi: 10.1080/1055678021000060829. URL http://dx.doi.org/10.1080/1055678021000060829.

[25] BMW Group. BMW 7er Limousine: Technische Daten. http://www.bmw.de/de/neufahrzeuge/7er/limousine/2015/technische-daten.html, 02 2016.

[26] Martin Ittershagen and Stephan Gabriel Haufe. UBA-Emissionsdaten 2014 zeigen Trendwende beim Klimaschutz, 2015. URL https://www.umweltbundesamt.de/sites/default/files/medien/381/dokumente/pi_2015_31_03_uba-emissionsdaten_2014_zeigen_trendwende_beim_klimaschutz.pdf.

[27] Stephen Michael Lavalle. *Planning Algorithms*. Cambridge University Press, 2006. ISBN 9780521862059. URL http://planning.cs.uiuc.edu/.

[28] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015. ISSN 10991514. doi: 10.1002/oca.2123. URL http://onlinelibrary.wiley.com/doi/10.1002/oca.2123/abstract;jsessionid=A365926EF43204B80C7CBD979392A269.f03t01.

[29] José María Maestre, Miguel A. Ridao, Attila Kozma, Carlo Savorgnan, Moritz Diehl, Minh Dang Doan, Anna Sadowska, Tamas Keviczky, Bart De Schutter, Holger Scheu, Wolfgang Marquardt, Felipe Valencia, and Jairo Jose Espinosa. A comparison of distributed MPC schemes on a hydro-power plant benchmark. *Optimal Control Applications and Methods*, 36(3):306–332, 2015. ISSN 10991514. doi: 10.1002/oca.2154. URL `http://dx.doi.org/10.1002/oca.2154`.

[30] Olvi Leon Mangasarian and Stanley Fromovitz. The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications*, 17(1):37–47, 1967. ISSN 0022247X. doi: 10.1016/0022-247X(67)90163-1. URL `http://www.sciencedirect.com/science/article/pii/0022247X67901631`.

[31] Natasha Merat, A. Hamish Jamson, Frank C.H. Lai, Michael Daly, and Oliver M.J. Carsten. Transition to manual: Driver behaviour when resuming control from a highly automated vehicle. *Transportation Research Part F: Traffic Psychology and Behaviour*, 27(PB):274–282, nov 2014. ISSN 13698478. doi: 10.1016/j.trf.2014.09.005. URL `http://dx.doi.org/10.1016/j.trf.2014.09.005`.

[32] Razvan C. Rafaila and Gheorghe Livint. Predictive control of autonomous steering for ground vehicles. In *Advanced Topics in Electrical Engineering (ATEE), 2015 9th International Symposium on*, pages 543–547. IEEE, 2015. ISBN 978-1-4799-7514-3. doi: 10.1109/ATEE.2015.7133880. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7133880`.

[33] Stephen M. Robinson. Strongly regular nonsmooth generalized equations. *Mathematical Programming*, 5(1):43–62, 1980. URL `http://www.jstor.org/stable/3689393`.

[34] Stephen M. Robinson. Generalized equations and their solutions, part II: Applications to nonlinear programming. In Monique Guignard, editor, *Mathematical Programming Study*, volume 19, chapter 2, pages 200–221. Springer Berlin Heidelberg, 1982. doi: 10.1007/BFb0120989. URL `http://www.springerlink.com/index/10.1007/BFb0120989`.

[35] Youcef Saad and Martin H Schultz. Gmres: a Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986. doi: 10.1137/0907058. URL `http://dx.doi.org/10.1137/0907058`.

[36] Sebastian Schlenkrich, Andreas Griewank, and Andrea Walther. On the local convergence of adjoint Broyden methods. *Mathematical Programming*, 121(2):221–247, 2010. ISSN 14364646. doi: 10.1007/s10107-008-0232-y. URL `http://link.springer.com/article/10.1007/s10107-008-0232-y`.

[37] Santokh Singh. Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey, 2015. URL `http://www-nrd.nhtsa.dot.gov/pubs/812115.pdf`.

[38] Michael Stingl, Michal Kočvara, and Günter Leugering. A Sequential Convex Semidefinite Programming Algorithm with an Application to Multiple-Load Free Material Optimization. *Dx.Doi.Org*, 20(1):130–155, 2009. ISSN 1052-6234. doi: 10.1137/070711281. URL `http://dx.doi.org/10.1137/070711281`.

[39] John R. Treat, Nicholas. S. Tumbas, Stephen T. McDonald, David Shinar, R. D. Hume, Richard E. Mayer, R. L. Stansifer, and N. John Castellan. Causes of Road Accidents, 1979. ISSN 0959-8138. URL `http://www.bmj.com/cgi/doi/10.1136/bmj.1.4975.1098`.

[40] Valerio Turri, Ashwin Carvalho, Hongtei Eric Tseng, Karl Henrik Johansson, and Francesco Borrelli. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 378–383, The Hague, 2013. IEEE. ISBN 9781479929146. doi: 10.1109/ITSC.2013.6728261. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6728261`.

[41] Michael Ulbrich and Stefan Ulbrich. *Nichtlineare Optimierung*. Birkhäuser, Basel, 1 edition, 2014. ISBN 978-3-0346-0142-9. doi: 10.1007/978-3-0346-0654-7. URL `http://link.springer.com/book/10.1007/978-3-0346-0654-7`.

[42] Andreas Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, 2002. URL `https://www.research.ibm.com/people/a/andreasw/papers/thesis.pdf`.

[43] Victor M. Zavala and Mihai Anitescu. Real-Time Optimization as a Generalized Equation. *SIAM J. Control Optim.*, 48(8):5444–5454, 2010. ISSN 0363-0129. doi: 10.1137/090762634. URL `http://dx.doi.org/10.1137/090762634`.

A. Appendix

A.1. Notation

In the following some notations and definitions will be explained.

A.1.1. Sets

Ball

A *ball* with centre $x \in \mathbb{R}^n, n \in \mathbb{N}$ with radius $0 < r \in \mathbb{R}$ is denoted as

$$\mathcal{B}(x, r) = \{\widehat{x} \in \mathbb{R}^n \,|\, \|x - \widehat{x}\| < r\}. \tag{A.1}$$

where $\|\cdot\|$ is the Euclidean norm. Note that a ball is convex. $\bar{\mathcal{B}}(x, r)$ is the closure of $\mathcal{B}(x, r)$.

Affine hull

The *affine hull* of a set $\Omega \in \mathbb{R}^n$ is defined as

$$\mathrm{aff}(\Omega) = \left\{ \sum_{i=1}^{k} \lambda_i x^i \,\middle|\, x^i \in \Omega, \sum_{i=1}^{k} \lambda_i = 1,\ k \in \mathbb{N},\ \lambda_i \in \mathbb{R} \right\}. \tag{A.2}$$

The affine hull is the set of all possible affine combinations of points in Ω. Moreover, it is the smallest affine set containing Ω.

Relative interior

The *relative interior* of a set $\Omega \subseteq \mathbb{R}^n$ is defined as

$$\mathrm{ri}(\Omega) = \{x \in \Omega \,|\, \mathcal{B}(x, r) \cap \mathrm{aff}(C) \subseteq \Omega, r > 0\}. \tag{A.3}$$

Normal cone

The *normal cone* of a convex set Ω at point $x \in \mathbb{R}^n$ is defined as:

$$\mathcal{N}_\Omega(x) = \begin{cases} \left\{u \in \mathbb{R}^n | u^T(x - v) \geq 0, v \in \Omega\right\}, & \text{if } x \in \Omega \\ \emptyset, & \text{otherwise} \end{cases} \tag{A.4}$$

Remark A.1. Note that, for $\widehat{\varepsilon} > 0, \Omega \subseteq \mathbb{R}^n, x \in \mathbb{R}^n$ with $\mathcal{B}(x, \widehat{\varepsilon}) \subseteq \Omega$ the normal cone $\mathcal{N}_\Omega(x)$ is always $\{0\}$. In fact, in this setting 0 is in the normal cone iff $x \in \Omega$. As $\mathcal{B}(x, \widehat{\varepsilon}) = x + \mathcal{B}(0, \varepsilon), \varepsilon > 0$ holds, the following equation is true:

$$\left\{u \in \mathbb{R}^n | u^T v \leq 0, v \in \mathcal{B}(0, \varepsilon)\right\} \supseteq \mathcal{N}_\Omega(x) \supseteq \{0\}.$$

As $u^T v > 0, v \in \mathcal{B}(0,\varepsilon) \Rightarrow u^T(-v) < 0$ with $-v \in \mathcal{B}(0,\varepsilon)$ is a contradiction, $u^T v = 0, \forall v \in \mathcal{B}(0,\varepsilon)$ must hold. The scalar product is bilinear and $\{0\} = \left\{u \in \mathbb{R}^n | u^T v \geq 0, v \in \mathcal{B}(0,\varepsilon)\right\}$ is therefore true.

A.1.2. Functions

Continuity The set of all continuous functions is denoted as:

$$\mathscr{C}^i[\mathbb{R}^n, \mathbb{R}^m] = \{f : \mathbb{R}^n \mapsto \mathbb{R}^m | f \text{ is } i \text{ times continuous differentiable}\}. \tag{A.5}$$

In some cases this is shortened to $\mathscr{C}^i$.

Derivative Operators

Partial Derivatives The partial derivative of a function f with respect to the variable x is denoted as $\frac{\partial}{\partial x} f$.

Gradient The gradient of a function $f(x) \in \mathscr{C}^1(\mathbb{R}^n, \mathbb{R})$ is denoted as

$$\nabla_x f = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right)^T \tag{A.6}$$

Jacobian The Jacobian of a function $g(x) \in \mathscr{C}^1(\mathbb{R}^n, \mathbb{R}^m)$ is denoted as $g'(x)$.

A.1.3. Matrices

Let $\mathcal{S}^n$ denote the set of all symmetric matrices $M \in \mathbb{R}^{n \times n}$. The set $\mathcal{S}^n_+$ is the set of all symmetric positive semi-definite matrices in $\mathbb{R}^{n \times n}$ and $\mathcal{S}^n_{++}$ is the set of all positive definite matrices in $\mathbb{R}^{n \times n}$. If a matrix M is positive (semi-)definite we also write $(M \succeq 0), M \succ 0$. For negative definiteness the signs $\preceq, \prec$ are used.

Weighted Norm

For a linear operator $M \in \mathcal{S}_{++}$ we can define a norm for x:

$$\|x\|^2_M = x^T M x \tag{A.7}$$

This notation may also hold positive semi-definite operators but leads to a semi-norm.

A.2. Constraint Qualifications

A condition implying the Guignard constraint qualification is called *constraint qualification* (CQ). In the following the CQs which are needed later will be presented.

A.2.1. Slater CQ

A problem satisfies the Slater CQ if

$$\exists x \text{ s.t. } h(x) < 0, g = 0, \tag{A.8}$$

see [8]. If there are implicit constraints of the form $x \in \Omega$ with an nonempty, closed and convex set $\Omega \in \mathbb{R}^n$ given. This leads to the equation.

$$\exists\ x \text{ s.t. } h(x) < 0, g = 0, \mathrm{ri}(\Omega) \neq \emptyset \tag{A.9}$$

A.2.2. Robinson CQ

A problem satisfies the Robinson CQ (see [7]) at x if $\exists \Delta x$ with

$$\begin{aligned} h(x) + \nabla_x h(x)\Delta x + \widetilde{x} = 0, \widetilde{x} \in \mathbb{R}^n_+, \\ g(x) + \nabla_x g(x)\Delta x = 0 \end{aligned} \tag{A.10}$$

Note, that the Robinson CQ is strongly related to the Mangasarian-Fromovitz CQ [30] and both imply the Slater CQ A.2.1.

A.3. Approximating Hessian of Lagrangian and Jacobian

A.3.1. Approximating the Hessian of the Lagrangian

In (1.46) the Hessian of the Lagrangian $\mathcal{L}$ was defined for a linear f. For problems with a nonlinear goal function, the Lagrangian has the form:

$$\nabla_x^2 \mathcal{L}(z) = \nabla_x^2 f(x) + \sum_{i=1}^{n_{eqC}} y_i \nabla_x^2 g_i(x) \tag{A.11}$$

Calculating this in every time step is very time consuming. Therefore it is approximated by $\widetilde{H} \in \mathcal{S}_+$.

Constant approximation This kind of approximation is the computationally cheapest way. The approximation quality depends on the properties of the used model. Possible choices might be

$$\widetilde{H} = 0 \in \mathbb{R}^{n_{opt} \times n_{opt}},\ \widetilde{H} = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \in \mathbb{R}^{n_{opt} \times n_{opt}} \tag{A.12}$$

Dynamical approximation In (A.11) it can be seen that the Lagrange multiplier y^i is a weight of the Hessian of the equality function i. This leads to approximations of the form:

$$\widetilde{H} = \nabla_x^2 f(x) + \sum_{|y_i| \geq y_{bound}} y_i \nabla_x^2 g_i(x) \tag{A.13}$$

where y_{bound} may be a constant or dependent on y and chosen such that

$$|\,\{|y_i| \geq y_{bound}\}\,| = n_{update} \tag{A.14}$$

holds for all time steps k.

Using previous results The changes in parameters ξ and the optimization variable x in two consecutive time steps k and $k+1$ is assumed to be small. So the change in the Hessian of the Lagrangian might be small as well. Assume that at time step $k=0$ an $\widetilde{H}_0$ is already calculated. For the update length, k_{update} holds

$$\widetilde{H}_k = \begin{cases} \widetilde{\mathcal{H}}(k), & \text{if } k \mod k_{update} = 0 \\ \widetilde{H}_{k-1}, & \text{otherwise} \end{cases} \tag{A.15}$$

where $\widetilde{\mathcal{H}}(k)$ gives a new approximation for time step k. Note that this method has the problem that in the update step the calculation of $\widetilde{H}$ consumes more time than in the other time steps. $k_{update} > k_{max}$ can also hold, this case corresponds to a constant approximation.

A.3.2. Approximating the Jacobian of the equality constraint

As mentioned in section 1.3.5, using the exact solution $g'(x^k)$ as A_k leads to a special case of the APCSCP algorithm, called PCSCP. An other approach is the reuse of previous Jacobians, analogously to the paragraph above.