INAUGURAL-DISSERTATION
zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität Heidelberg

vorgelegt von
Dipl.-Math. Dennis Janka
aus Herborn

Tag der mündlichen Prüfung:
23. Juli 2015

Sequential quadratic programming with indefinite Hessian approximations for nonlinear optimum experimental design for parameter estimation in differential–algebraic equations

Gutachter: Dr. Stefan Körkel
Prof. Dr. Dr. h.c. mult. Hans Georg Bock

Zusammenfassung

In dieser Arbeit entwickeln wir Algorithmen zur numerischen Lösung von Problemen der nichtlinearen optimalen Versuchsplanung zur Parameterschätzung in differential–algebraischen Gleichungen. Diese Probleme können als spezielle, steuerungs- und pfadbeschränkte Probleme der optimalen Steuerung formuliert werden. Als Zielfunktion wird ein Funktional auf der Kovarianzmatrix der Modellparameter minimiert, wobei die Kovarianzmatrix durch Sensitivitäten erster Ordnung der Modellgleichungen gegeben ist. Zusätzlich ist die Zielfunktion nichtlinear in der Zeit gekoppelt, weshalb Probleme der optimalen Versuchsplanung eine schwierige Klasse von Optimalsteuerungsproblemen darstellen. Zur numerischen Lösung schlagen wir eine Parametrisierung mittels der direkten Mehrzielmethode vor, die auf ein strukturiertes Problem der nichtlinearen Programmierung (NLP) führt. Ein erweitertes System von Nominal- und Variationszuständen für die Modellsensitivitäten wird auf Mehrzielintervallen parametrisiert und die Zielfunktion mit Hilfe von zusätzlichen Variablen und Nebenbedingungen entkoppelt. Im resultierenden NLP identifizieren wir verschiedene Strukturen, die eine wesentlich effizientere Auswertung von Ableitungen erlauben im Vergleich zu einem Ansatz für herkömmliche Optimalsteuerungsprobleme.

Zur Lösung der blockstrukturierten NLPs entwickeln wir eine neue Methode der sequentiellen quadratischen Programmierung (SQP). In dieser werden partitionierte quasi-Newton Updates verwendet um die Hessematrix der Lagrange-Funktion zu approximieren, welche Blockdiagonalstruktur besitzt. Wir analysieren ein Modellproblem mit indefiniter blockdiagonaler Hessematrix und beweisen, dass positiv definite Approximationen der einzelnen Blöcke superlineare Konvergenz verhindern. Für ein Modellproblem der optimalen Versuchsplanung zeigen wir, dass, wenn das Mehrzielgitter verfeinert wird, in der Hessematrix immer mehr negative Eigenwerte auftauchen und bestätigen die negativen Auswirkungen positiv definiter Hessematrix Approximationen auf die Konvergenzgeschwindigkeit. Wir schlagen daher vor, indefinite SR1 Updates zu verwenden, die schnelle lokale Konvergenz garantieren. Wir entwickeln eine Filter-Liniensuche Globalisierungsstrategie, welche indefinite Hessematrizen akzeptiert. Dies basiert auf einem Kriterium, das aus dem globalen Konvergenzbeweis der Liniensuche abgeleitet ist. BFGS Updates mit einer Skalierungsstrategie, die zu große Eigenwerte verhindert, werden als Ausweichstrategie eingesetzt, falls die SR1 Updates die Konvergenz nicht begünstigen. Zur Lösung der anfallenden dünnbesetzten und nichtkonvexen quadratischen Teilprobleme wird eine parametrische Aktive-Mengen-Methode mit einer Steuerung des Trägheitsindex innerhalb eines Schur Komplement Ansatzes entwickelt. Die Methode benutzt eine symmetrische, indefinite LBL^T-Faktorisierung der großen, dünnbesetzten KKT Matrix und verwaltet und modifiziert QR-Faktoren eines kleinen, dichtbesetzten Schur Komplements.

Die neuen Methoden werden durch zwei C++ Implementierungen ergänzt: `muse` transformiert ein Problem der optimalen Versuchsplanung oder der optimalen Steuerung in ein strukturiertes NLP mittels der direkten Mehrzielmethode. Ein spezielles Merkmal der neuen Implementierung ist, dass vollständig unabhängige Gitter für die Parametrisierung der

Steuerungen, Zustände, Pfadbeschränkungen und Messzeitpunkte eingesetzt werden. Dies bietet eine höhere Flexibilität um die NLP Formulierung den Charakteristiken des konkreten Problems anzupassen und ermöglicht es, verschiedene Formulierungen im Lichte des Lifted Newton Verfahrens zu untersuchen. Das Softwarepaket `blockSQP` ist eine Implementierung der neuen SQP Methode und benutzt eine neu entwickelte Variante des quadratischen Lösers `qpOASES`. Numerische Ergebnisse für eine Kollektion von Testbeispielen der optimalen Versuchsplanung und optimalen Steuerung werden präsentiert, die zeigen, dass SR1 Approximationen die lokale Konvergenz gegenüber BFGS verbessern. Die neue Methode wird dann auf zwei schwierige Probleme der optimalen Versuchsplanung aus der chemischen Verfahrenstechnik angewandt und erweist sich als schneller als eine erhältliche vorhandene Implementierung.

Abstract

In this thesis we develop algorithms for the numerical solution of problems from nonlinear optimum experimental design (OED) for parameter estimation in differential–algebraic equations. These OED problems can be formulated as special types of path- and control-constrained optimal control (OC) problems. The objective is to minimize a functional on the covariance matrix of the model parameters that is given by first-order sensitivities of the model equations. Additionally, the objective is nonlinearly coupled in time, which make OED problems a challenging class of OC problems. For their numerical solution, we propose a direct multiple shooting parameterization to obtain a structured nonlinear programming problem (NLP). An augmented system of nominal and variational states for the model sensitivities is parameterized on multiple shooting intervals and the objective is decoupled by means of additional variables and constraints. In the resulting NLP, we identify several structures that allow to evaluate derivatives at greatly reduced costs compared to a standard OC formulation.

For the solution of the block-structured NLPs, we develop a new sequential quadratic programming (SQP) method. Therein, partitioned quasi-Newton updates are used to approximate the block-diagonal Hessian of the Lagrangian. We analyze a model problem with indefinite, block-diagonal Hessian and prove that positive definite approximations of the individual blocks prevent superlinear convergence. For an OED model problem, we show that more and more negative eigenvalues appear in the Hessian as the multiple shooting grid is refined and confirm the detrimental impact of positive definite Hessian approximations. Hence, we propose indefinite SR1 updates to guarantee fast local convergence. We develop a filter line search globalization strategy that accepts indefinite Hessians based on a new criterion derived from the proof of global convergence. BFGS updates with a scaling strategy to prevent large eigenvalues are used as fallback if the SR1 update does not promote convergence. For the solution of the arising sparse and nonconvex quadratic subproblems, a parametric active set method with inertia control within a Schur complement approach is developed. It employs a symmetric, indefinite LBL^T-factorization for the large, sparse KKT matrix and maintains and updates QR-factors of a small and dense Schur complement.

The new methods are complemented by two C++ implementations: `muse` transforms an OED or OC problem instance to a structured NLP by means of direct multiple shooting. A special feature is that fully independent grids for controls, states, path constraints, and measurements are maintained. This provides higher flexibility to adapt the NLP formulation to the characteristics of the problem at hand and facilitates comparison of different formulations in the light of the lifted Newton method. The software package `blockSQP` is an implementation of the new SQP method that uses a newly developed variant of the quadratic programming solver `qpOASES`. Numerical results are presented for a benchmark collection of OED and OC problems that show how SR1 approximations improve local convergence over BFGS. The new method is then applied to two challenging OED applications from chemical engineering. Its performance compares favorably to an available existing implementation.

Contents

List of acronyms

AD	Algorithmic differentiation
BVP	Boundary value problem
DAE	Differential–algebraic equation
DMS	Direct multiple shooting
IND	Internal numerical differentiation
IVP	Initial value problem
KKT	Karush-Kuhn-Tucker
NLP	Nonlinear programming problem
OC	Optimal control
ODE	Ordinary differential equation
OED	Optimum experimental design
QP	Quadratic programming problem
VDAE	Variational differential–algebraic equation
SQP	Sequential quadratic programming

List of selected symbols

Optimum experimental design: analytical problem

Symbol	meaning
y	differential states
z	algebraic states
y_v	variational differential states/sensitivities with respect to v
z_v	variational algebraic states/sensitivities with respect to v
u	controls
p	parameters
s^pe	parameterization variables from parameter estimation problem
v	$(p^T, s^{\mathrm{pe}T})^T$
f	ODE right-hand side
g	algebraic equation right-hand side
h	observable
c^d	dynamic constraints
c^b	multi-point boundary constraints
N^b	number of boundary constraints evaluation points
c^pe	parameterized boundary constraints from parameter estimation

Optimum experimental design: finite-dimensional problem

Symbol	meaning
N^s	number of multiple shooting intervals
N^c	number of control intervals
N^d	number of intervals between dynamic constraints evaluation points
N^m	number of intervals between measurement grid points
N^pe	number of evaluation points for parameterized boundary constraints
τ^s	multiple shooting grid points
τ^c	control grid points
τ^d	path constraint grid points
$\tau^\mathrm{d}_{j_i}$	path constraint grid point i on shooting interval j
τ^m	measurement grid points
s^y	multiple shooting variables for nominal differential states
s^z	multiple shooting variables for nominal algebraic states

$s^{y,i}$	multiple shooting variables for variational differential state i
$s^{z,i}$	multiple shooting variables for variational algebraic state i
$\bar{s}^{y}$	multiple shooting variables for nominal and variational differential states
$\bar{s}^{z}$	multiple shooting variables for nominal and variational algebraic states
q	variables for discretized control
$\hat{q}_j$	variables for discretized controls within multiple shooting interval j
w	measurement weights
$\hat{w}_j$	measurement weights within multiple shooting interval j
W	$\mathrm{diag}(w_1, \ldots, w_{N^{\mathrm{m}}})$
H_1	variables for Fisher matrix $J_1^T W J_1$
H_2	variables for Jacobian J_2

Sequential quadratic programming

Iteration indices for all quantities appear as superscripts in squared brackets, for example, $x^{[k]}$ denotes the iterate of primal variables during iteration k.

Symbol	meaning
k	SQP iteration index
ν	QP iteration index
φ	objective function
c	constraints
x	primal variables
λ	Lagrange multipliers
$\mathcal{L}$	Lagrangian
d	QP solution
$\mathcal{A}$	active bounds
$\mathcal{S}$	set of active bounds for which additionally $d = 0$
A	Jacobian of equality constraints
$A_{\mathcal{A}}$	matrix A augmented by unit row-vectors corresponding to indices in $\mathcal{A}$
B	(Approximation of the) Hessian of the Lagrangian
γ	difference of gradients of the Lagrangian
δ	difference of (primal) iterates

Introduction

Modeling, simulation, and optimization have become indispensable tools to understand and improve processes in science and engineering. A common approach is to derive a *model* from physical first principles to describe the process under consideration. In this thesis, we are interested in processes that can be modeled by systems of differential–algebraic equations (DAEs). This applies to numerous processes from chemical engineering, biology, astronomy, and many other fields. The model can then be *simulated*, meaning that the model equations are solved, provided that all initial values and parameters are known. This is typically done by suitable algorithms that approximate the solution and can be done for a variety of scenarios. For example, a chemical process can be simulated for different temperatures. Ideally, we now use the model to *optimize* the process: We want to select those settings which allow to run the process in an optimal way with respect to a given performance indicator, e.g., time or energy consumption.

But how can we be sure that the model correctly predicts the behavior of the process? This leads to the problem of *model validation*, which can be outlined as follows. First, we take observations or measurements under defined conditions and fit the model to the observations by adjusting the values of so-called *model parameters*. In our setting, model parameters are intrinsic quantities of the system. Examples are heat transfer coefficients, or activation energies for chemical reactions. If we knew the value of the parameters, we could make accurate predictions for the process. A reasonable way is to choose those values for the parameters such that the model reproduces the given observations. We call those values *parameter estimates*. Now, one can analyze the statistical uncertainty of these estimates that arises due to uncertainty in the observations. Only if the model can reproduce the observations and the uncertainty of the model parameters is sufficiently low, we say that the model is validated. Only then we can expect it to make accurate predictions about the process.

If the uncertainty of the estimates is too high and hence the model is not validated, more measurements are required. However, experiments to obtain new measurements usually are expensive and tight restrictions often apply under which the process may be operated and measurements may be obtained. This leads to the question of *optimum experimental design (OED)*: Within the given restrictions, find those experimental conditions and take those measurements that allow to identify the model parameters with minimum uncertainty. In this thesis, we develop theory, algorithms, and software for the formulation and the efficient numerical solution of OED problems.

Problem description and challenges

OED for general statistical models has been studied for several decades and it is a well-established field of research, see the textbooks [10, 36, 67, 167]. Nonlinear OED for processes modeled by differential equations can be posed as a special type of optimal control problems, in which the sensitivities of the model states with respect to the model parameters are used to compute an approximation of the covariance matrix or the Fisher information matrix. Consequently, differential equations for the model sensitivities also appear in the optimal control problem formulation. In an optimal control framework, this gives rise to large systems of DAEs of dimension $n_{\mathrm{x}} + n_{\mathrm{x}} \cdot n_{\mathrm{p}}$, where n_{x} is the dimension of the DAE system describing the process and n_{p} is the number of model parameters. In this form, the problem has been discussed by various authors, e.g., [15, 66, 134, 135, 146]. A certain maturity was reached with the advent of the dedicated OED software package VPLAN [134], that allows to apply OED within an industrial context.

Due to their high-dimensionality and complicated structure, OED problems are usually solved using the direct approach: The control functions that represent experimental conditions are approximated in a finite-dimensional space and the states are parameterized by finitely many variables. That way, the OED optimal control problem is replaced by a suitable, finite-dimensional nonlinear programming problem (NLP). This thesis is concerned with the following two aspects of OED: How to best transcribe the infinite-dimensional OED problem to a structured, finite-dimensional NLP, and how to efficiently solve this NLP using sequential quadratic programming (SQP) methods.

Most methods use embedded initial value problem (IVP) solvers to implement a single-shooting approach, wherein states and sensitivities are repeatedly evaluated within an outer optimization loop, see, e.g., the methods proposed in [15, 134, 135], [66], [85], or [177]. The so-called simultaneous optimization approach proposed in [122] applies orthogonal collocation to the full system of nominal and variational states, giving rise to a large-scale NLP. Further references on OED methods and applications can be found in the survey paper [83].

In this work, we concentrate on the direct multiple shooting (DMS) method for OED. Based on Bock's direct multiple shooting method for optimal control problems [34, 142], it was first discussed in [136] and in the diploma thesis [124]. Here, nominal and variational states are discretized on a coarse grid of shooting intervals, and IVP solvers are employed to compute approximate solutions and sensitivities on the subintervals. Continuity constraints ensure equivalence to the solution of the original DAE after convergence. DMS offers several advantages over single shooting: It often reduces nonlinearity of the problem, which results in an enlargement of the local domain of full-step convergence for the NLP solver. Furthermore, the higher-dimensional NLPs possess a decoupled structure, which makes them suitable for parallelization. In this thesis, we further investigate DMS for OED. A main challenge of OED is that the systems are much larger than for standard optimal control problems because the nominal system must be augmented by a variational system for the problem sensitivities. Consequently, the resulting NLP comprises many constraints. An NLP solver needs to repeatedly linearize the problem and for DMS, the evaluation of the

constraint Jacobian is usually the most expensive part of the solution process. A solution approach using a standard optimal control software package that ignores the structures of OED is therefore not suitable for practical use, see [124]. Addressing this situation, we analyze the derivative structures in detail and show how to efficiently evaluate derivatives by taking sparsity into account and by re-using certain directional sensitivities. This ensures that the problem can be linearized very efficiently in a given point.

Another major issue in the numerical treatment of OED problems is that NLP solvers often need excessively large numbers of iterations to converge. To overcome this problem, the second focus of this work is on efficient SQP methods specialized for NLPs that arise in DMS for OED. SQP methods date back to the classical methods of Wilson [203], Han [117], and Powell [165] proposed in the 1960s and 70s. Together with interior point methods, they rank among the most efficient and popular NLP methods today. We favor SQP methods over interior point methods because they usually require less function and derivative evaluations and are especially efficient when second derivatives are not available, see [99]. During the last decades, many SQP methods have been proposed, see the survey papers [35], [109], and [100]. State-of-the-art implementations of some of the methods are available, e.g., SNOPT [93] or filterSQP [76]. However, they were developed for a very broad set of NLPs and make it difficult to address the special structures that arise in DMS. On the other hand, dedicated solvers in multiple shooting packages, such as the SQP method in MUSCOD-II [59, 142], are very closely intertwined with the problem data and do not provide the flexibility required to efficiently treat specially structured subclasses such as OED.

In this thesis, we present a new SQP method that takes into account the structures that arise in DMS and is at the same time separated from the problem data as much as possible. By analyzing an OED model problem, we prove that indefinite Hessians are needed to ensure fast local convergence. This poses challenges for the solution of quadratic subproblems as well as the globalization strategy. For the new method, we enhance the parametric QP solver `qpOASES` [70] to treat sparse, nonconvex quadratic programming problems. This is achieved by a Schur complement approach [27, 94, 97, 104] in connection with a new inertia control mechanism. Global convergence is achieved by a filter line search based on [197, 198]. We extend the line search framework by an inner loop where indefinite Hessians can be accepted. The Hessian of the Lagrangian is approximated blockwise by low-rank update formulae: The *Symmetric Rank One (SR1)* and the *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* update formulae. Our new method is able to switch between indefinite block-SR1 updates and specially scaled, positive definite block-BFGS updates that provide progress if the SR1 update does not promote convergence. A new criterion allows to efficiently detect when the SR1 update is acceptable, such that the SR1-BFGS strategy incurs only little overhead when compared to a pure BFGS strategy.

Finally, the quality and usefulness of numerical methods is not only judged by theoretical considerations but also by software implementations that can solve practically relevant problems. The new methods developed in this thesis are complemented by two efficient C++ implementations: The software package `muse` has been developed to enhance the established software package VPLAN by multiple shooting capabilities for OED and OC problems.

The solver `blockSQP` is a new implementation of the filter line search SQP method with indefinite block updates that is interfaced to `muse`. Their favorable performance is illustrated on a range of demanding applications from OED and OC.

Contributions of the thesis

The thesis contributes nonlinear programming techniques for the numerical treatment of optimum experimental design problems for differential-algebraic equation models. We propose algorithms based on the direct multiple shooting method to transform OED problems into finite-dimensional NLPs, which we thoroughly analyze. Motivated by the problem structures that we identify, we develop a new, suitable sequential quadratic programming method to solve the NLPs. Together with dedicated software implementations, the new methods constitute a powerful toolkit for the efficient solution of nonlinear OED problems. More specifically, the main contributions of this thesis are:

Direct multiple shooting formulation for OED constrained by DAE
We present a decoupled formulation, wherein nominal and variational states are parameterized by multiple shooting. We propose to resolve the nonlinear coupling in the objective by means of a linearly coupled constraint. We introduce four grids for controls, shooting intervals, potential measurements, and path constraints that are explicitly allowed to be chosen independently. We describe how the framework can be applied in four different settings: constrained parameter estimation problems, OED for key performance indicators, the design of multiple experiments, and the presence of continuous measurements.

Analysis of derivative structures of the resulting NLP
We give, for the first time, a detailed analysis of the constraint Jacobian as well as the objective gradient and Hessian of the resulting NLPs. Due to the specific structure induced by the continuity conditions for the variational DAEs, a special sparsity pattern is discovered. Furthermore, many directional derivatives can be re-used, which greatly reduces the computational load for evaluation of the constraint Jacobian. We derive the objective Hessian of the A-criterion in a form that can be efficiently evaluated and exploited in an SQP method with blockwise updates.

`muse` – A parallel multiple shooting method for optimum experimental design
We provide a C++-implementation of the new multiple shooting formulation as an extension to the software package VPLAN. It uses the established VPLAN interfaces to describe the problem input and sets up the multiple shooting NLP exploiting all discovered OED derivative structures. As additional benefit, support for generic optimal control problems is included. For the first time, fully independent grids for controls, shooting intervals, potential measurements, and path constraints are supported in the implementation. Evaluation of sensitivities is parallelized using OpenMP on shooting interval and multi-experiment level. With `muse`, OED and OC problems can now be treated efficiently by the direct multiple shooting method.

Analysis of different Hessian update strategies using an OED model problem
For a model problem with indefinite, block-diagonal Hessian, we prove that every positive definite block-diagonal Hessian approximation yields only a linear rate of convergence. Then, we analyze an OED model problem parameterized by multiple shooting in the light of the lifted Newton method [5] and show that for every lifted term, the Hessian gains a negative eigenvalue. Numerical experiments confirm that increasing the number of multiple shooting intervals leads to an increasing number of SQP iterations if positive definite Hessian approximations are used, while indefinite SR1 approximations provide fast local convergence.

A filter line-search SQP method compatible with indefinite Hessians
We describe a filter line-search SQP framework based on [198]. We extend the method by an inner loop, in which different Hessian approximations are tried that are allowed to be indefinite. We use an assumption from the proof of global convergence to derive a new criterion when to accept a Hessian. We show that this criterion can be efficiently evaluated by modifying a QP solver with inertia control. A disadvantage of most filter methods is the need for a feasibility restoration phase which is computationally expensive. As a remedy, we present an efficient heuristic for problems arising in DMS that allows to circumvent the feasibility restoration phase on most occasions.

Selective sizing strategy for block-diagonal Hessians
We show that the selective sizing strategy proposed for unconstrained optimization in [50] can be applied to block-diagonal Hessians in constrained optimization. We observe that block-BFGS updates often accumulate many large eigenvalues that result in unnecessarily small steps. The selective sizing strategy mitigates this effect by multiplying the approximations in every iteration with a suitable, small scalar based on recently observed curvature information. This strategy greatly improves the performance of the BFGS update in all of our numerical experiments.

An active-set quadratic programming solver for sparse, nonconvex problems
A new version of the parametric active-set solver `qpOASES` [70] is developed that is capable of reliably solving sparse and nonconvex problems. It is based on the Schur complement approach described in [27, 94, 97, 104] and uses the sparse, symmetric, indefinite solver `MA57` [63] to factorize the sparse KKT systems. We build on previous developments by A. Wächter (Northwestern University) and provide a new inertia control mechanism to find critical points of sparse and nonconvex QPs. The inertia control is based on monitoring the sign of the determinant of the Schur complement every time a bound is removed from the working set. Furthermore, we describe details of our implementation of the new method that improve the numerical stability of the method.

`blockSQP`: An SR1-BFGS SQP method for NLPs with block-diagonal Hessian matrix
We provide a new implementation of the proposed SQP method in the C++-package `blockSQP`. The new Schur-complement variant of the QP solver `qpOASES` is used to solve the sparse, nonconvex quadratic subproblems. `blockSQP` approximates the block-diagonal Hessian of the Lagrangian by blockwise, limited- or full-memory SR1 and BFGS update

schemes. Parallel solution of the SR1 and BFGS QPs is supported. When used in connection with `muse`, `blockSQP` can benefit from the restoration heuristic for problems arising in DMS. A minimum 2-norm constraint violation restoration phase is implemented as fallback. `blockSQP` has been designed with OED and OC problems in mind, but can also be used stand-alone to solve generic NLPs.

Numerical results on a benchmark collection of OED and OC problems for different SQP variants
We compare `blockSQP` with different Hessian approximations and find that using SR1 updates when possible provides faster convergence than using only BFGS updates. Furthermore, we show that a dedicated SQP method with block-diagonal Hessian approximations is necessary to solve problems arising in DMS by showing that the general purpose SQP solver SNOPT fails on a number of OED problems.

Numerical results to show the effect of lifting with respect to convergence speed
The flexibility in our implementation makes it possible to compare problem instances with identical grids for controls and path constraints but different multiple shooting grids, which corresponds to a lifting of the constraints in the sense of the lifted Newton method [5]. We identify problems with a tracking objective as a class of problems that clearly benefit from lifting in terms of convergence speed. We also find that for most problems other, non-local effects dominate the speed of convergence.

Two OED case studies with DMS
We illustrate the efficiency of the new methods and implementations with case studies for two challenging example applications, a continuous stirred-tank reactor and the Urethane reaction. For both examples, the new method is significantly better than the existing single shooting implementation in VPLAN in terms of SQP iterations and CPU time.

Thesis overview

The thesis is divided into four parts that comprise a total of twelve chapters. Part I consists of two chapters that provide background material for parts II and III. In the first chapter, we compile basic results from constrained nonlinear optimization. Then we give a broad overview of SQP methods by describing different parts that characterize a method such as, e.g., setup of the quadratic subproblem, choice of the Hessian approximation, globalization strategy, and globalization mechanism. In the second chapter, we give a detailed description of the direct multiple shooting method for optimal control problems. We consider DMS in a general form and introduce notation that allows different grids for controls, states, and path constraints. We briefly comment on practical issues and give an overview of existing DMS methods.

The second part of this thesis deals with optimum experimental design problem formulations. In Chapter 3, we derive the OED problem at the solution of a parameter estimation problem. We explain how to obtain the covariance matrix and discuss the relation between

covariance matrices for constrained and unconstrained parameter estimation problems. Then we set up the OED problem as a special kind of optimal control problem. Finally, we discuss several problem variants such as OED with multiple experiments, continuous measurements or OED for key performance indicators. While Chapter 3 discusses OED only from the point of view of infinite-dimensional optimal control, Chapter 4 is dedicated to formulations suited for the numerical solution of the problem. We first discuss a classical single shooting approach. Then we describe a multiple shooting parameterization that forms the core of the new methods in this thesis. Afterwards, we provide a detailed analysis of the derivative structure of the resulting NLP. Finally, we describe how to formulate several problem variants within our numerical framework.

Part III comprises four smaller chapters and describes the newly developed SQP techniques. We start with the NLPs that result from DMS for OED and discuss desirable features of SQP methods for their solution. The discussion is supported by the analysis of Hessian update strategies using an OED model problem in the light of the lifted Newton method that shows that indefinite Hessian approximations are needed for fast convergence. The following three chapters deal with the different ingredients of the new SQP method. Most results are formulated for general NLPs but always maintain a view towards multiple shooting structured NLPs. We begin with a filter line search SQP framework with an additional inner loop for indefinite Hessian approximations in Chapter 6. The following chapter deals with specific definite and indefinite Hessian approximations based on the SR1 and BFGS updates. Finally, in Chapter 8, we describe how to find critical points of the sparse and indefinite subproblems with a Schur complement variant of the quadratic programming solver `qpOASES`.

Part IV is dedicated to the new software developed in the context of the thesis and numerical tests that illustrate the capabilities of the new methods. In Chapter 9, we describe our multiple shooting implementation `muse` and the SQP solver `blockSQP`. We outline the object-oriented program structure and highlight important classes. We also report default values for all algorithmic parameters. The numerical results achieved with the new software are divided into three chapters: Chapter 10 evaluates different algorithmic variants of `blockSQP` on a benchmark collection of 6 OED and 10 OC case studies. There, we compare different Hessian scaling strategies and approximation sequences. In Chapter 11, we investigate two applications from chemical engineering where optimum experimental designs are computed with the new methods. In Chapter 12, we study the effect of lifting on the convergence behavior. To this end, we create a family of equivalent NLPs with a fixed grid for the control functions but different grids for the multiple shooting nodes.

Part I.

Background

Chapter 1.

Elements of nonlinear programming

In this chapter, we give an introduction to central concepts of nonlinear optimization, often called nonlinear programming. In the first part, we review some definitions and theoretical results on constrained nonlinear optimization. We follow the presentation of [154], but all results can be found in every textbook on the subject, e.g., [74, 87, 148, 191].

In the second part we give an overview of *sequential quadratic programming (SQP) methods* that rank among the most popular methods for the numerical solution of nonlinear optimization problems and play an important role throughout this work. An overview is given in [35], more recent surveys include [109, 143, 100].

In the third part, we briefly present alternative nonlinear programming methods, namely augmented Lagrangian and interior point methods.

1.1. Theoretical foundations

Nonlinear programming is concerned with the solution of the following nonlinear constrained optimization problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \varphi(x) && \text{(1.1a)}\\ \text{s.t.} \quad & c_i(x) = 0, \quad i \in \mathcal{E}, && \text{(1.1b)}\\ & c_i(x) \geq 0, \quad i \in \mathcal{I}, && \text{(1.1c)} \end{aligned}$$

where φ and the functions c_i are all smooth, real-valued functions on a subset of $\mathbb{R}^n$, and $\mathcal{E}$ and $\mathcal{I}$ are two finite, disjoint sets of indices with $\mathcal{E} \cup \mathcal{I} = \{1, \ldots, m\}$. We call ϕ the *objective* function, while c_i, $i \in \mathcal{E}$ are the *equality constraints* and c_i, $i \in \mathcal{I}$ are the *inequality constraints*. A point x is said to be *feasible* for problem (1.1) if it satisfies both the equality and inequality constraints. We denote the set of all feasible points by Ω.

Definition 1.1 (Local and global solution)**.** A vector $x^\star$ is a *local solution* of the problem (1.1) if it is feasible and there is an open neighborhood U of $x^\star$ such that $\varphi(x) \geq \varphi(x^\star)$ for $x \in U \cap \Omega$. A vector $x^\star$ is a *global solution* of the problem (1.1) if it is feasible and $\varphi(x) \geq \varphi(x^\star)$ for $x \in \Omega$.

All algorithms developed in this work are concerned with finding local solutions. For global optimization, entirely different algorithms are needed but the maximum size of

problem instances that can be handled in practical computations is considerably smaller, see [81, 160] for an overview.

We now introduce some definitions that are required to formulate conditions for local optimality.

Definition 1.2 (Active set)**.** The *active set* $\mathcal{A}(x)$ at any feasible point x consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints i for which $c_i(x) = 0$; that is,

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) = 0\}.$$

At a feasible point x, the inequality constraint $i \in \mathcal{I}$ is said to be *active* if $c_i(x) = 0$ and *inactive* if the strict inequality $c_i(x) > 0$ is satisfied.

Definition 1.3 (Lagrangian function)**.** The *Lagrangian function* for problem (1.1) is defined as

$$\mathcal{L}(x, \lambda) = \varphi(x) - \sum_{i=1}^{m} \lambda_i c_i(x).$$

The quantities λ_i, $i \in \mathcal{E} \cup \mathcal{I}$ are called *Lagrange multipliers*.

Definition 1.4 (LICQ)**.** Given a feasible point x and the active set $\mathcal{A}(x)$, we say that the *linear independence constraint qualification (LICQ)* holds if the set of active constraint gradients $\{\nabla c_i(x), i \in \mathcal{A}(x)\}$ is linearly independent.

Note that LICQ is a rather strong condition and in fact does not always hold in practice. An important problem class where LICQ is violated are so-called *mathematical programs with equilibrium constraints (MPEC)* that arise, for example, in bilevel optimization [118] or in mixed integer optimal control [128]. Several other constraint qualification exist to allow the formulation of optimality conditions and facilitate the development of algorithms for problem classes where LICQ does not hold, see, e.g., [1].

We are now able to formulate the *Karush–Kuhn–Tucker (KKT)* conditions which are the foundation for most nonlinear programming algorithms.

Theorem 1.5 (First-order necessary conditions)**.** *Suppose that $x^\star$ is a local solution of* (1.1)*, that the functions φ and c_i in* (1.1) *are continuously differentiable, and that the LICQ holds at $x^\star$. Then there exists a unique Lagrange multiplier vector $\lambda^\star$, with components $\lambda_i^\star$, $i \in \mathcal{E} \cup \mathcal{I}$ such that the following conditions are satisfied at $(x^\star, \lambda^\star)$:*

$$\nabla_x \mathcal{L}(x^\star, \lambda^\star) = 0, \tag{1.2a}$$
$$c_i(x^\star) = 0, \quad \text{for all } i \in \mathcal{E} \tag{1.2b}$$
$$c_i(x^\star) \geq 0, \quad \text{for all } i \in \mathcal{I} \tag{1.2c}$$
$$\lambda_i^\star \geq 0, \quad \text{for all } i \in \mathcal{I} \tag{1.2d}$$
$$\lambda_i^\star c_i(x^\star) = 0, \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}. \tag{1.2e}$$

A point that satisfies the KKT conditions (1.2) is called a *KKT point*. We sometimes refer to condition (1.2a) as *stationarity*, conditions (1.2b) and (1.2c) as *primal feasibility*, condition (1.2d) as *dual feasibility* and to condition (1.2e) as *complementarity* conditions. A special case of complementarity is important and deserves its own definition.

Definition 1.6 (Strict complementarity)**.** Given a local solution $x^\star$ of (1.1) and a vector $\lambda^\star$ satisfying (1.2), we say that the *strict complementarity condition* holds if exactly *one* of $\lambda_i^\star$ and $c_i(x^\star)$ is zero for each index $i \in \mathcal{I}$. In other words, we have that $\lambda_i^\star > 0$ for each $i \in \mathcal{I} \cap \mathcal{A}(x^\star)$.

We denote by $A(x^*) = [\nabla c_i(x^*)]^T_{i\in\mathcal{A}(x^*)}$ the matrix whose rows are the active constraint gradients at x^*. Let further Z be a matrix whose columns are a basis for the null space of $A(x^\star)$, that is,

$$Z \in \mathbb{R}^{n\times(n-|\mathcal{A}(x^\star)|)}, \quad A(x^\star)Z = 0.$$

Then we can formulate second-order necessary and sufficient optimality conditions as follows.

Theorem 1.7 (Second-order necessary conditions)**.** *Suppose $x^\star$ is a local solution of* (1.1) *and that the LICQ as well as strict complementarity are satisfied. Let $\lambda^\star$ be the Lagrange multiplier vector for which the KKT conditions* (1.2) *are satisfied and Z be a matrix whose columns form a basis of the null space of $A(x^\star)$. Then*

$$v^T Z^T \nabla_{xx}\mathcal{L}(x^\star,\lambda^\star)Zv \geq 0, \quad \text{for all } v \in \mathbb{R}^{n-|\mathcal{A}(x^\star)|}, \tag{1.3}$$

that is, the Hessian of the Lagrangian is positive semidefinite on the null space of the active constraints.

Theorem 1.8 (Second-order sufficient conditions)**.** *Given the assumptions of Theorem 1.7 we further assume that*

$$v^T Z^T \nabla_{xx}\mathcal{L}(x^\star,\lambda^\star)Zv > 0, \quad \text{for all } v \in \mathbb{R}^{n-|\mathcal{A}(x^\star)|}\backslash\{0\}, \tag{1.4}$$

that is, the Hessian of the Lagrangian is positive definite on the null space of active constraints. Then $x^\star$ is a strict local solution for (1.1)*.*

We also call $Z^T\nabla_{xx}\mathcal{L}(x^\star,\lambda^\star)Z$ the *reduced* or *projected* Hessian.

1.1.1. Rates of convergence

Algorithms for nonlinear programming typically start with an initial guess for the solution $x^{[0]}$ and then produce a sequence of iterates $\{x^{[k]}\}$ to find a KKT point. For these algorithms, we distinguish between two notions of convergence: *Global convergence* of an algorithm means that the limit point of the sequence of iterates produced by the algorithm is in fact a KKT point, independent of the choice of the initial point. On the other hand, the term *local convergence* describes how fast the iterates approach a KKT point from within a neighborhood around that point. Rates of (local) convergence are typically defined in the following terms:

Definition 1.9 (Q-rates of convergence)**.** Let $\{x^{[k]}\}$ be a sequence in $\mathbb{R}^n$ that converges to $x^\star$. We say that the convergence is

- *Q-linear* if there is a constant $r \in (0,1)$ such that

$$\frac{\left\|x^{[k+1]} - x^\star\right\|}{\left\|x^{[k]} - x^\star\right\|} \leq r, \quad \text{for all } k \text{ sufficiently large,}$$

- *Q-quadratic* if there is a constant $M \in \mathbb{R}$ such that

$$\frac{\left\|x^{[k+1]} - x^\star\right\|}{\left\|x^{[k]} - x^\star\right\|^2} \leq M, \quad \text{for all } k \text{ sufficiently large,}$$

- *Q-superlinear* if

$$\lim_{k\to\infty} \frac{\left\|x^{[k+1]} - x^\star\right\|}{x^{[k]} - x^\star} = 0.$$

A weaker form of convergence is given by

Definition 1.10 (R-rates of convergence)**.** Let $\{x^{[k]}\}$ be a sequence in $\mathbb{R}^n$ that converges to $x^\star$. We say that the convergence is *R-{linear | superlinear | quadratic}* if

$$\left\|x^{[k]} - x^\star\right\| \leq \beta^{[k]}, \quad \text{for all } k$$

and $\{\beta^{[k]}\}$ converges Q-{linearly | superlinearly | quadratically} to zero.

Note that in R-convergent sequences the error is allowed to increase, which cannot occur for Q-convergent sequences, as they insist on a decrease at every step k, for k sufficiently large.

1.2. Sequential quadratic programming methods

The term sequential quadratic programming (SQP) method rather describes a conceptual approach than a specific algorithm. The first SQP method was formulated by Wilson in his thesis in 1963 [203] but SQP has gained popularity mainly due to the methods formulated by Han [117] and Powell [165]. Since then, a great variety of methods have been proposed and SQP remains an active field of research.

1.2.1. General solution approach

To solve the general NLP (1.1), SQP methods try to find a KKT point by producing a sequence of *iterates* $(x^{[k]}, \lambda^{[k]})$, $k = 0, 1, 2, \ldots$, starting with an initial guess for the solution $(x^{[0]}, \lambda^{[0]})$. To obtain the new iterate, a *step* or *displacement* $d^{[k]}$ is computed by minimizing a quadratic model subject to a linearization of the constraints about the current iterate:

$$\min_{d \in \mathbb{R}^n} \quad \tfrac{1}{2} d^T B^{[k]} d + \nabla \varphi^{[k]T} d \tag{1.5a}$$

$$\text{s.t.} \quad \nabla c_i^{[k]T} d + c_i^{[k]} = 0, \quad i \in \mathcal{E}, \tag{1.5b}$$

$$\nabla c_i^{[k]T} d + c_i^{[k]} \geq 0, \quad i \in \mathcal{I}, \tag{1.5c}$$

where we use the notation $c_i^{[k]} := c_i(x^{[k]})$, $\nabla c_i^{[k]} := \nabla c_i(x^{[k]})$, $\nabla \varphi^{[k]} := \nabla \varphi(x^{[k]})$, and $B^{[k]} := \nabla^2_{xx} \mathcal{L}(x^{[k]}, \lambda^{[k]})$. The new iterate is $x^{[k+1]} = x^{[k]} + d^{[k]}$ and $\lambda^{[k+1]}$ are the multipliers of the linearized constraints (1.5b) and (1.5c). The procedure is repeated until an optimum is found. We have summarized this basic SQP iteration scheme in Algorithm 1. This approach is

Algorithm 1: Local SQP iteration.

Given initial estimate $x^{[0]}$, $\lambda^{[0]}$, set $k = 0$.
while $x^{[k]}$ *is not optimal* **do**
  Evaluate $\varphi(x^{[k]})$, $\nabla \varphi(x^{[k]})$, $c(x^{[k]})$, $\nabla c(x^{[k]})$, $\nabla^2_{xx} \mathcal{L}(x^{[k]}, \lambda^{[k]})$.
  Solve the quadratic program (1.5) to obtain $d^{[k]}$ and multipliers $\lambda_{QP}^{[k]}$.
  Set the new iterates $x^{[k+1]} = x^{[k]} + d^{[k]}$, $\lambda^{[k+1]} = \lambda_{QP}^{[k]}$.
  Set $k = k + 1$.
end

closely related to Newton's method for the solution of nonlinear equations. It is easy to show that the SQP method described above applied to the equality constrained problem,

$$\min_{x \in \mathbb{R}^n} \quad \varphi(x), \quad \text{s.t.} \quad c(x) = 0, \tag{1.6}$$

is equivalent to Newton's method applied to the KKT conditions (1.2) of this problem (cf. [154, ch. 18]).

Note that the basic iteration in Algorithm 1 does not constitute an SQP method that is suitable for practical use. The following issues must be addressed to obtain a practical method:

1. Convergence tests and termination criteria to decide when an iterate is sufficiently close to a minimum
2. Algorithms for quadratic programming to compute the step
3. Choice of the Hessian of the quadratic subproblem

4. Globalization strategies that monitor progress of the iterates and thus ensure convergence from remote starting points

5. Globalization mechanisms to reduce a step computed by the quadratic subproblem

For all these aspects several options have been proposed, giving rise to a broad family of algorithms that all fall in the category of SQP methods.

1.2.2. Convergence test

Since SQP methods aim to find a KKT point, it is reasonable to base the convergence test on the KKT conditions, i.e. declare optimality if

$$\left\|c_+^{[k]}\right\| \leq \varepsilon, \qquad \left\|\nabla_x \mathcal{L}(x^{[k]}, \lambda^{[k]})\right\| \leq \varepsilon, \tag{1.7}$$

$$\lambda_i^{[k]} \geq -\varepsilon, \quad i \in \mathcal{I}, \qquad c_i^{[k]} \lambda_i^{[k]} \leq \varepsilon, \quad i \in \mathcal{E} \cup \mathcal{I} \tag{1.8}$$

where ε is a given tolerance and the vector $c_+^{[k]} = c_+(x^{[k]})$ denotes the constraint violation and consists of the components

$$[c_+^{[k]}]_i := \begin{cases} c_i(x^{[k]}) & \text{if } i \in \mathcal{E}, \\ \min\{0, c_i(x^{[k]})\} & \text{if } i \in \mathcal{I}. \end{cases} \tag{1.9}$$

We refer to [62] for a survey of convergence tests based on this observation.

An alternative is the so-called KKT tolerance that also takes into account the current step, see [141]:

$$\left|\nabla \varphi^{[k]T} d^{[k]}\right| + \sum_{i=1}^{m} \left|\lambda_i^{[k]} c_i^{[k]}\right| \leq \varepsilon$$

However, a disadvantage of this criterion is that small stepsizes can lead to small values of the KKT tolerance and thus premature termination at non-optimal points.

Further measures need to be taken to detect (local) infeasibility of the problem (1.1) in which case the convergence tests given above fail, see, for example, the method described in [42].

1.2.3. Solution of the quadratic subproblem

A key requirement for any SQP method is the efficient and reliable solution of the quadratic subproblem (1.5) in every step. Quadratic programming is a field of research in its own right and here we only give a very brief overview of the main concepts of a quadratic programming method. A comprehensive bibliography on the subject is compiled in [108].

Algorithms for quadratic programming can be broadly classified as *interior point* or *active set* methods. *Interior point methods* generate a sequence of points that lie in the interior of the feasible region. To this end, a barrier function is constructed such that it becomes

(infinitely) expensive to violate the constraints, e.g., by choosing a logarithmic barrier. The resulting equality constrained nonlinear program is then solved by Newton's method, and the procedure is repeated with a decreased barrier weight, see also Section 1.3.2. Typically, interior point methods require fewer, but more expensive iterations than active set methods to solve a single QP. However, they lack the good warm-start capabilities of active set methods when solving a series of related QPs and are thus less common in an SQP context.

Active set methods maintain a *working set* of active constraints and solve the resulting equality constrained QP problem. The working set is updated repeatedly until the optimality conditions of the QP are satisfied. Active set methods are closely related to the simplex method of linear programming [204]. In *primal* active set methods, a sequence of primal feasible iterates is generated until dual feasibility and hence optimality is reached [154]. Note that an initial primal feasible point is required. It can be obtained by a so-called *Phase I* which usually is an iterative—possibly computationally expensive—procedure itself. *Dual* active set methods generate a sequence of dual feasible points until primal feasibility and hence optimality is reached, see [102]. A third class of active set methods are *parametric* QP methods. Introduced in [21], they trace the solution of a linear homotopy between a QP problem with known solution and the QP problem to be solved, while maintaining primal as well as dual feasibility for all intermediate, perturbed problems. In Chapter 8, we describe some extensions to the parametric active set method `qpOASES` [70]. A major advantage of all active set methods over interior point methods in an SQP context is that they can be *warm-started*. Practical experience shows that active set obtained in one SQP iteration is a very good guess for the quadratic subproblem in the following SQP iteration, that means only few active set iterations are needed. In the following, we mention further aspects of active set methods that also affect the corresponding SQP method.

Convex and nonconvex problems

Many quadratic programming methods require the QP to be *convex*, which means the Hessian $B^{[k]}$ needs to be positive definite. In this case, SQP methods usually need to guarantee convexity of the QP by employing positive definite approximations to the exact Hessian or make use of suitable convexification schemes if the exact Hessian or an indefinite approximation should be used.

For general nonconvex QPs, so-called *inertia-controlling* algorithms such as [73] or [95] monitor the eigenvalues of the reduced Hessian over the course of the active set iterations. Note, however, that the solution of nonconvex QPs and even the verification of the optimality conditions for a given point is known to be an NP-hard problem [49, 161, 162]. In practice, methods for nonconvex QPs usually only provide solutions that satisfy the first- and second-order necessary conditions 1.5 and 1.7.

Numerical linear algebra techniques

In theory, many algorithms for quadratic programming are equivalent in the sense that they produce the same iterates if exact arithmetic and the same pivot rules are used [20]. In practice, however, their performance largely depends on the numerical procedures used. Numerical linear algebra techniques form a vital part of any practial quadratic programming

method and must guarantee that the method is numerically stable as well as efficient. Most active set methods maintain matrix factorizations that are efficiently updated every time the working set changes thus keeping the cost of a working set change comparably low. Existing methods maintain, e.g., a LDL^T factorization of the (positive definite) reduced Hessian and a QR or TQ factorization of the constraint matrix [92, 70]. For sparse problems, the symmetric indefinite KKT matrix for the initial working set can be factored efficiently by means of a direct sparse factorization and in subsequent iterations a dense factorization of the Schur complement is constructed and updated according to the changes in the working set [27, 94]. A variant of this *Schur complement method* for general nonconvex QPs is presented in Chapter 8.

Finally, for large and sparse problem, iterative linear algebra techniques can be applied if the iteration matrices cannot be explicitly formed or factored to produce inexact solutions of the QP. This gives rise to the class of *inexact* SQP methods, see, e.g., [51].

1.2.4. Choice of the Hessian

The local rate of convergence is largely determined by the choice of the Hessian $B^{[k]}$ in the QP (1.5). The exact Hessian of the Lagrangian yields a method that is locally equivalent to Newton's method for solving the KKT conditions and inherits the fast local quadratic convergence rate of Newton's method. However, using the exact Hessian suffers from several drawbacks: First of all, in many applications, second derivatives are not available or very expensive to evaluate. Secondly, the Hessian is only required to be positive definite on the null space of active constraints at a solution and it is usually not a positive definite matrix. The resulting nonconvex QPs may be unbounded and usually a trust region must be employed to guarantee the existence of a solution.

The most popular alternatives are so-called *quasi-Newton* approximations of the Hessian that are used in many SQP methods. They usually start with an initial approximation, e.g., $B^{[0]} = I$ and then update it in a simple manner to incorporate curvature measured in the most recent step. This is done by imposing the secant condition on the new approximation $B^{[k+1]}$:

$$B^{[k+1]}\delta^{[k]} = \gamma^{[k]},$$

where $\delta^{[k]}$ is the displacement and $\gamma^{[k]}$ denotes the difference of the gradients of the Lagrangian at $x^{[k]}$ and $x^{[k+1]}$. This secant condition gives rise to a family of quasi-Newton update formulae that add a simple rank 1 or rank 2 correction to the current Hessian approximation. A generalization of the quasi-Newton approach are *total quasi-Newton* methods, that additionally rely on an adjoint secant condition to approximate not only the Hessian of the Lagrangian but also the constraint Jacobian by low rank update formulae [114].

The first quasi-Newton update formula was proposed by W.C. Davidon in 1959 [54] and popularized by Fletcher and Powell in 1963 [80]. The most widely used update formula today is the BFGS update (named after Broyden [37], Fletcher [72], Goldfarb [101], and Shanno [184]). In particular, the BFGS update can be modified to retain positive definiteness, always leading to convex subproblems, see [166]. In contrast, the Symmetric Rank One

(SR1) may become indefinite or even undefined in some iterations yet it is often found to yield better approximations of the Hessian, see [40, 46, 129].

A successful variant of quasi-Newton methods are *limited memory* methods [153, 41], where only curvature from the M most recent SQP iterations is incorporated to form the current approximation. Chapter 7 deals with Hessian approximations based on the BFGS and SR1 updates that are modified by special scaling techniques.

Note that the choice of the Hessian is largely responsible for the local rate of convergence of the SQP method. An SQP method that uses the exact Hessian of the Lagrangian converges locally Q-quadratic under certain assumptions [154]. A quasi-Newton BFGS method for *un*constrained problems can be shown to converge Q-superlinearly [55]. A similar result carries over to the constrained case if the Hessian of the Lagrangian is positive definite at the solution and the unmodified BFGS update is used. However, this assumption is not satisfied for many applications and the BFGS update needs to be damped as described in [166], in which case the rate of convergence is only R-superlinear. For the limited memory version of BFGS, only R-linear convergence can be shown, however, it is often found to be very efficient in practice [145].

1.2.5. Globalization strategies: merit functions and filters

Like Newton's method for nonlinear equations, the general solution approach outlined above guarantees convergence only in a small neighbourhood of a solution. Globalization strategies are concerned with ensuring convergence from remote starting points by monitoring progress of the iterates generated by the quadratic subproblems. If progress is deemed insufficient the step must be modified appropriately. How the modification is done is defined by globalization *mechanisms* that are described in the next section. Note that in the literature one often only finds the term globalization strategies to describe both concepts. However, we feel that a distinction between these two is appropriate as both are complementary to each other. Thus we adopt the terms globalization strategy and globalization mechanism as used, e.g., in [143].

Merit functions

The idea of merit or penalty functions is to combine objective and a measure of the constraint violation into a single function, whose local minimizers correspond to the ones of the NLP (1.1). Convergence from remote starting points can then be achieved by forcing descent in the merit functions using one of the mechanisms presented in the next section.

A popular merit function is the ℓ_1 exact penalty function:

$$\Phi_\rho(x) = \varphi(x) + \rho \left\| c_+(x) \right\|_1 ,$$

where $\rho > 0$ is the penalty parameter and $c_+(x)$ is the constraint violation as defined in (1.9). It can be shown that a local minimizer $x^\star$ of $\Phi_\rho(x)$ is a local minimizer of problem (1.1) if $\rho > \|\lambda^\star\|_\infty$, where $\lambda^\star$ are the corresponding Lagrange multipliers. Other possibilities for penalty functions that have been used to promote global convergence include augmented Lagrangian functions of the form $\varphi(x) - \lambda^T c(x) + \frac{\rho}{2} \left\| c_+(x) \right\|_2^2$.

For all penalty functions, the choice of the parameter ρ in every step is a critical issue because one would have to know $\lambda^\star$ in advance to make sure that a minimum of the penalty function corresponds in fact to a minimum of problem (1.1). Furthermore, an inappropriate choice of ρ can cause an unnecessary modification of the full step and thus interference with the progress of the iterations. Strategies for choosing the penalty parameter can be found, e.g., in [154, ch. 18.3]. A recent approach suggests to steer the penalty parameter by comparing predicted and actual decrease of the constraint violation, see [45].

Filters

The difficulty of choosing an appropriate penalty parameter eventually lead to the development of filter methods. First introduced by Fletcher and Leyffer [77], other filter methods have been proposed subsequently, e.g., [106, 192] along with analysis of global and local convergence properties [75, 79, 193, 197, 198]. A brief history of filter methods is given in [78].

Filter methods require an iterate to improve either the objective or the constraint violation compared to all previous iterates instead of a linear combination of both. More precisely, filter methods maintain a set of pairs—called *filter*—$\mathcal{F}^{[k]}$ of objective function values $\varphi^{[l]} := \varphi(x^{[l]})$ and constraint violation $\eta^{[l]} := \left\|c_+(x^{[l]})\right\|$ from previous iterations l that define a "prohibited" region in the (φ, η) plane. A new point $\hat{x}$ is acceptable if it sufficiently decreases objective function or constraint violation, that is, if

$$\varphi(\hat{x}) \leq \varphi^{[l]} - \beta_\varphi^{\mathcal{F}} \eta^{[l]} \quad \text{or} \quad \|c_+(\hat{x})\| \leq (1 - \beta_\eta^{\mathcal{F}})\eta^{[l]}, \quad \text{for all } (\varphi^{[l]}, \eta^{[l]}) \in \mathcal{F}^{[k]},$$

where $\beta_\eta^{\mathcal{F}}, \beta_\varphi^{\mathcal{F}} \in (0,1)$ are constants that define a so-called slanting envelope to ensure that iterates cannot accumulate at infeasible limit points. Figure 1.1 illustrates the concept of a filter that defines a prohibited region of dominated points. If a step is rejected by the filter it can be modified by means of a line search or a trust region mechanism. A related concept are so-called *funnel* methods proposed by Gould and Toint [107] that can be viewed as filter methods with just a single filter entry. This entry corresponds to an upper bound of the constraint violation and the upper bound is reduced during iterations that mostly reduce the constraint violation.

1.2.6. Globalization mechanisms: line search and trust region

Whenever a step computed by the quadratic subproblem does not yield sufficient progress as defined by the globalization strategy, it must be modified appropriately. Two broad concepts exist that can be combined with merit functions as well as filters: line search and trust region methods.

Line search methods

Line search methods search along the direction $d^{[k]}$ produced by the quadratic subproblem (1.5) until sufficient reduction in either a merit function or a filter is found. Different criteria for sufficient reduction can be defined, e.g., the Armijo condition [154]. Algorithm 2 presents a generic backtracking line search method with a merit function. It is important that

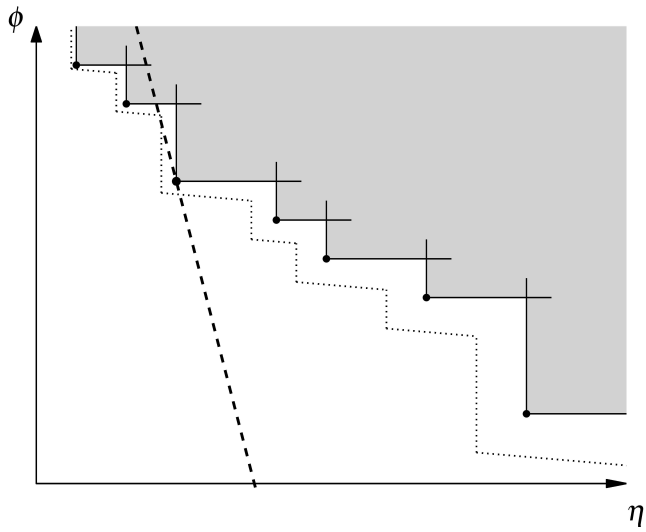


Figure 1.1.: A filter defined by a set of iterates (dots) in the (φ, η) plane with slanting envelope defined by $\beta_\eta^{\mathcal{F}} = \beta_\varphi^{\mathcal{F}} = 0.1$ (dotted line). The gray area corresponds to the set of dominated points. For comparison, the level set of a penalty function evaluated at one iterate for a fixed penalty parameter ρ is given (dashed line). The choice of ρ determines the slope of the line. Iterates on the left-hand side of the dashed line provide descent in the merit function.

the direction $d^{[k]}$ is indeed a descent direction. This is the case, e.g., for positive definite Hessian approximations. In Chapter 6, we present a filter line search method that also accepts indefinite Hessians, provided they are positive definite on a certain subspace.

Trust region methods

Trust region methods explicitly restrict the step *before* solving the QP by adding a *trust region* constraint of the form $\|d\| \leq \Delta^{[k]}$ to the QP. The trust region radius $\Delta^{[k]}$ is adjusted in every iteration depending on how well the QP approximates the NLP inside the trust region. A disadvantage of trust region methods is that the QPs can be become inconsistent if $\Delta^{[k]} \to 0$, which can be dealt with, e.g., by invoking a feasibility restoration phase, see [77]. Algorithm 3 describes a generic trust region SQP method, for a comprehensive treatment of trust region methods we refer to [48].

1.2.7. Examples of SQP methods

SNOPT

SNOPT [93] implements a line search SQP method. A full space or limited memory BFGS update is used as Hessian approximation. Its quadratic programming solver SQOPT [98] is a reduced Hessian active set method and solves a convex QP in every iteration. Sparsity in the constraints is exploited. Convergence from remote starting points is ensured via a line search

Algorithm 2: Generic line search SQP method with merit function.

Given initial estimate $x^{[0]}$, $\lambda^{[0]}$, set $k = 0$.
while $x^{[k]}$ *is not optimal* **do**
  Evaluate $\varphi(x^{[k]})$, $\nabla\varphi(x^{[k]})$, $c(x^{[k]})$, $\nabla c(x^{[k]})$, $\nabla^2_{xx}\mathcal{L}(x^{[k]},\lambda^{[k]})$.
  Solve the quadratic program (1.5) to obtain $d^{[k]}$ and multipliers $\lambda^{[k]}_{QP}$.
  Set $\alpha^{[k,0]} = 1$, $l = 0$.
  repeat
    Set $\alpha^{[k,l+1]} = \alpha^{[k,l]}/2$.
    Evaluate $\Phi(x^{[k]} + \alpha^{[k,l+1]}d^{[k]})$.
    Set $l = l + 1$.
  until $\Phi(x^{[k]} + \alpha^{[k,l]}d^{[k]})$ *is sufficiently smaller than* $\Phi(x^{[k]})$
  Set the new iterates $x^{[k+1]} = x^{[k]} + \alpha^{[k,l]}d^{[k]}$, $\lambda^{[k+1]} = \alpha^{[k,l]}\lambda^{[k]}_{QP} + (1-\alpha^{[k,l]})\lambda^{[k-1]}$.
  Set $k = k + 1$.
end

Algorithm 3: Generic trust region SQP method.

Given initial estimate $x^{[0]}$, $\lambda^{[0]}$, initial trust region radius $\Delta^{[k]}$, set $k = 0$.
while $x^{[k]}$ *is not optimal* **do**
  Evaluate $\varphi(x^{[k]})$, $\nabla\varphi(x^{[k]})$, $c(x^{[k]})$, $\nabla c(x^{[k]})$, $\nabla^2_{xx}\mathcal{L}(x^{[k]},\lambda^{[k]})$.
  Set $\Delta^{[k,0]} = \Delta^{[k]}$, $l = 0$.
  while $x^{[k]} + d^{[k,l]}$ *is not sufficiently better than* $x^{[k]}$ **do**
    Solve the quadratic program

$$\begin{aligned} \min_d \quad & \tfrac{1}{2}d^T B^{[k]} d + \nabla\varphi^{[k]T} d \\ \text{s.t.} \quad & \nabla c_i^{[k]T} d + c_i^{[k]} = 0, \quad i \in \mathcal{E}, \\ & \nabla c_i^{[k]T} d + c_i^{[k]} \geq 0, \quad i \in \mathcal{I}, \\ & \|d\| \leq \Delta^{[k,l]}. \end{aligned}$$

    to obtain $d^{[k,l]}$ and $\lambda^{[k+1,l]}_{QP}$.
    if $x^{[k]} + d^{[k,l]}$ *is sufficiently better than* $x^{[k]}$ **then**
      Choose trust region radius for next iteration $\Delta^{[k+1]} \geq \Delta^{[k,l]}$.
    else
      Decrease the trust region radius, e.g., $\Delta^{[k,l+1]} = \Delta^{[k,l]}/2$, set $l = l + 1$.
    end
  end
  Set the new iterates $x^{[k+1]} = x^{[k]} + d^{[k,l]}$ and $\lambda^{[k+1]}$ depending on if the trust region constraint is active. Set $k = k + 1$.
end

on an augmented Lagrangian merit function.

filterSQP
filterSQP [76] is a trust region SQP method and global convergence is enforced by a filter. It accepts exact Hessians of the Lagrangian. It employs BQPD as QP solver, which solves the resulting nonconvex quadratic subproblems and is able to exploit sparsity to some extent. If the QPs become inconsistent because the trust region radius becomes too small, a feasibility restoration phase is invoked.

KNITRO
The software package KNITRO [44] comprises three algorithms one of which, KNITRO active, is a *sequential linear quadratic programming (SLQP)* algorithm [43]. It uses two kinds of trust regions: First, a linear programming problem with a trust region is solved to obtain a guess for the active set. Afterwards, an equality constrained QP with a trust region is solved whose objective is a quadratic approximation of the Lagrangian. The solution of the LP and the QP are used to produce a search direction and a penalty function enforces global convergence.

1.3. Other nonlinear programming methods

For this section, we consider a special form of problem 1.1 where we only allow simple bounds on some of the variables instead of general inequalities:

$$\min_{x\in\mathbb{R}^n} \quad \varphi(x) \tag{1.11a}$$
$$\text{s.t.} \quad c_i(x) = 0, \quad i = 1,2,\ldots,m, \tag{1.11b}$$
$$x_i \geq 0, \quad i \in \mathcal{I}. \tag{1.11c}$$

Note that the general NLP 1.1 can be transformed to this form by introducing slack variables.

1.3.1. Augmented Lagrangian methods

Penalty or augmented Lagrangian methods are named after the *augmented Lagrangian* function for problem 1.11:

$$\mathcal{L}_A(x,\lambda;\mu) = \varphi(x) - \sum_{i=1}^{m} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i=1}^{m} c_i^2(x), \tag{1.12}$$

where μ is called the *penalty parameter*. They try to find a minimum of problem 1.11 by successively minimizing the following bound constrained problem for an increasing sequence of penalty parameters μ:

$$\min_{x\in\mathbb{R}^n} \quad \mathcal{L}_A(x,\lambda;\mu), \quad \text{s.t.} \quad x_i \geq 0,\ i \in \mathcal{I}. \tag{1.13a}$$

A general framework is described by Algorithm 4. For details on how to solve the bound constrained minimization problem in every step or how to update the multipliers we refer

to the literature, e.g., [154, ch. 17]. Examples of augmented Lagrangian methods are LANCELOT [47] and MINOS [151].

Algorithm 4: Bound-constrained augmented Lagrangian method.

Given starting point $x_s^{[0]}$, $\lambda^{[0]}$, penalty parameter $\mu^{[0]}$, tolerance $\tau^{[0]}$, set $k = 0$.
while $x^{[k]}$ *is not optimal* **do**
  Find an approximate minimizer $x^{[k]}$ of $\mathcal{L}_A(\cdot, \lambda^{[k]}; \mu^{[k]})$ satisfying $x_i \geq 0,\ i \in \mathcal{I}$, starting at $x_s^{[k]}$, with convergence tolerance $\tau^{[k]}$.
  Update Lagrange multipliers to obtain $\lambda^{[k+1]}$.
  Choose new penalty parameter $\mu^{[k+1]} \geq \mu^{[k]}$.
  Set starting point for the next iteration $x_s^{[k+1]} = x^{[k]}$.
  Select tolerance τ^{k+1}.
  Set $k = k + 1$.
end

1.3.2. Interior point methods

Interior point or barrier methods solve problem 1.11 by solving a sequence of *barrier problems* with a decreasing barrier parameter $\mu > 0$ given by:

$$\min_{x \in \mathbb{R}^n} \quad \varphi(x) - \mu \sum_{i \in \mathcal{I}} \log x_i \tag{1.14a}$$

$$\text{s.t.} \quad c_i(x) = 0, \quad i = 1, 2, \ldots, m. \tag{1.14b}$$

Note that the objective of this problem becomes arbitrarily large as x_i, $i \in \mathcal{I}$ approach zero. Hence, a solution to Problem 1.14 lies in the interior of the feasible region of problem 1.11, i.e. the strict inequalities $x_i > 0$ hold for $i \in \mathcal{I}$.

Interior point methods often come in the form of *primal-dual* methods. The following dual variables are introduced:

$$v_i := \frac{\mu}{x_i}, \quad i \in \mathcal{I}.$$

With this definition, the KKT conditions for problem 1.14 are:

$$\nabla \varphi(x) - \sum_{i=1}^{m} \lambda_i \nabla c_i(x) - v = 0, \tag{1.15a}$$

$$c_i(x) = 0, \quad i = 1, 2, \ldots, m \tag{1.15b}$$

$$x_i v_i - \mu = 0, \quad i \in \mathcal{I}. \tag{1.15c}$$

Note, that for $\mu = 0$ these conditions together with the inequalities

$$x_i \geq 0 \quad \text{and} \quad v_i \geq 0, \quad i \in \mathcal{I} \tag{1.16}$$

are the KKT conditions for the original problem 1.11. The dual variables v then correspond to the multipliers for the bound constraints (1.11c). Primal-dual interior point methods solve system (1.15) by a Newton-type approach, maintaining iterates for the primal variables x and the dual variables v and possibly λ. Different strategies exist to choose a sequence of barrier parameters with $\mu \to 0$. Either μ is changed after the perturbed KKT conditions (1.15) are satisfied to some accuracy or it is changed adaptively in every iteration.

Popular interior point methods are, e.g., IPOPT [199], KNITRO [44], and LOQO [195].

1.3.3. The generalized Gauss–Newton method

The generalized Gauss–Newton method is a method for solving special types of NLPs, namely *constrained nonlinear least squares problems* of the form

$$\min_{x\in\mathbb{R}^n} \quad \tfrac{1}{2}\,\|F_1(x)\|_2^2 \tag{1.17a}$$

$$\text{s.t.} \quad F_2(x) = 0. \tag{1.17b}$$

with twice continuous differentiable functions $F_1 : \mathbb{R}^n \to \mathbb{R}^{m_1}$ and $F_2 : \mathbb{R}^n \to \mathbb{R}^{m_2}$. Least squares problems appear, e.g., in the context of maximum likelihood parameter estimation, see Sec. 3.1.

Search directions for the generalized Gauss–Newton method are computed by solving *constrained linear least squares problems* of the form:

$$\min_{\Delta x\in\mathbb{R}^n} \quad \tfrac{1}{2}\|F_1(x) + J_1(x)\Delta x\|_2^2 \tag{1.18a}$$

$$\text{s.t.} \quad F_2(x) + J_2(x)\Delta x = 0, \tag{1.18b}$$

where $J_i(x)$ are the Jacobians of F_i:

$$J_1(x) := \nabla F_1(x)^T \in \mathbb{R}^{m_1\times n},$$
$$J_2(x) := \nabla F_2(x)^T \in \mathbb{R}^{m_2\times n}.$$

Note that problem (1.18) is a special case of the quadratic program (1.5). In exact arithmetic, a Gauss–Newton method produces (locally) the same iterates as an SQP method that uses $J_1(x)^T J_1(x)$ as approximation of the Hessian. However, for solving the linear least squares problems (1.18), tailored linear algebra based on suitable factorizations of J is used instead of a general QP algorithm. The generalized Gauss–Newton method was first described in [29] and is analyzed in detail in [31]. Algorithm 5 describes the basic steps of the method.

Algorithm 5: Generalized Gauss–Newton method.

Given initial estimate $x^{[0]}$, set $k = 0$.

while *a convergence test for* $x^{[k]}$ *is not satisfied* **do**

Solve the following linear least squares problem to obtain $\Delta x^{[k]}$:

$$\min_{\Delta x \in \mathbb{R}^n} \quad \tfrac{1}{2}\|F_1^{[k]} + J_1^{[k]}\Delta x\|_2^2 \tag{1.19a}$$

$$\text{s.t.} \qquad F_2^{[k]} + J_2^{[k]}\Delta x = 0, \tag{1.19b}$$

where $F_i^{[k]} := F_i(x^{[k]})$ and $J_i^{[k]} := J_i(x^{[k]})$, $i \in \{1,2\}$.

Set the new iterate

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]}\Delta x^{[k]},$$

with $\alpha^{[k]} \in (0,1]$ obtained by a suitable globalization strategy.

Set $k = k+1$.

end

Chapter 2.

Optimization of dynamic processes

In this chapter, we describe optimization problems that are constrained by differential algebraic equations. First, we introduce the problem class of optimal control problems along with some notation that is used throughout this work. Then we present the direct multiple shooting method as one of the most prominent methods for the numerical solution of optimal control problems. It transforms the problems into specially structured nonlinear programs that can be efficiently solved by tailored NLP methods. In contrast to the original formulations of the direct multiple shooting method [34, 163], a more generic variant is described here which allows the parameterization grids for the states, controls, and path constraints to be chosen independently. We conclude with a brief discussion of the structures of the resulting NLPs that are characteristic for the direct multiple shooting method.

For a more comprehensive overview on optimization of dynamic processes we refer to the textbooks [22, 24, 38, 89].

2.1. Problem formulation

We start our discussion of dynamic optimization with two classes of problems, that appear in various contexts throughout this work. First, we give a brief description of differential algebraic equation models. Many optimization problems involving differential algebraic equations can be cast in the form of optimal control problems, which are described afterwards.

2.1.1. Differential algebraic equation models

Processes in science and engineering can often be modeled by differential equations derived from physical principles. Throughout this work, we consider dynamic systems described by *semi-explicit differential algebraic equations (DAE)*. Given a finite time horizon $t_0 \leq t \leq t_\mathrm{f}$, they have the form

$$\dot{y}(t) = f(t, y(t), z(t), p, u(t)), \tag{2.1a}$$

$$0 = g(t, y(t), z(t), p, u(t)). \tag{2.1b}$$

We call $y(t) \in \mathbb{R}^{n_y}$ *differential states* and $z(t) \in \mathbb{R}^{n_z}$ *algebraic states*. We assume the system to be of *index 1*, which means that $\partial_z g(t, y, z, p, u)$ is regular. The right-hand side functions f and g are assumed to be piecewise sufficiently smooth.

The system also depends on *parameters* $p \in \mathbb{R}^{n_\mathrm{p}}$ and (time-dependent) *controls* $u(t) \in \mathbb{R}^{n_\mathrm{u}}$. We think of a parameter as some intrinsic property of the process that has a true and constant value, for example a material constant or a frequency factor for a chemical reaction. Because their true values are usually not known, parameters must be *estimated* from measurement data.

Controls, on the other hand, are extrinsic quantities that act upon the process such as temperature or pressure. They can usually be selected within certain ranges. In general, controls are time-dependent but sometimes we want to explicitly consider time-independent controls $u_0 \in \mathbb{R}^{n_{\mathrm{u}_0}}$, e.g., initial concentrations of substances in a reactor. For notational convenience, we just identify them with continuous, constant functions and think of them as the first n_{u_0} components of u. This means that whenever we refer to the vector u_0, we mean in fact $\big(u_1(t), \dots, u_{n_{\mathrm{u}_0}}(t)\big)^T$ for some $t \in [t_0, t_\mathrm{f}]$. Typically, controls and states must stay within certain bounds. We denote these dynamic or path constraints by

$$0 \le c^\mathrm{d}(t, y(t), z(t), p, u(t)) \in \mathbb{R}^{n_\mathrm{d}}, \quad t \in [t_0, t_\mathrm{f}]. \tag{2.2}$$

Initial conditions are required for the differential states only. For the algebraic states, initial values are given implicitly by the algebraic equation (2.1b) which must be satisfied in particular at t_0. Note that it is a nontrivial task to find initial values $z(t_0)$ that are consistent with Equation (2.1b). Within an optimization context, this is often done with the help of a relaxed formulation, see the discussion in Sec. 2.2.1 below. Depending on the process, initial conditions for the differential states (2.1a) can be fixed, or modeled as parameters or controls. We use the following general setting of linearly coupled multi-point boundary constraints that include the initial conditions as special case:

$$0 = \sum_{i=1}^{N^\mathrm{b}} c_i^\mathrm{b}(t_i, y(t_i), z(t_i), p, u_0) \in \mathbb{R}^{n_\mathrm{b}}, \quad t_i \in [t_0, t_\mathrm{f}]. \tag{2.3}$$

Remark 2.1. Note that we do not allow pointwise evaluation of control functions except for u_0, the ones that are explicitly marked as continuous constant functions. The reason is that a control could be altered at a single point, i.e., a set of measure zero, without affecting the state trajectories.

2.1.2. The optimal control problem

When we have a model describing a process, the model can be used to optimize the process with respect to some performance index, e.g., time, energy, or yield. Many optimization problems for dynamic systems can be formulated as *optimal control (OC) problems* of the

following form:

$$\min_{u,y,z} \quad \Phi(y(t_\mathrm{f}), z(t_\mathrm{f}), u_0) \tag{2.4a}$$

$$\text{s.t.} \quad \dot{y}(t) = f(t, y(t), z(t), p, u(t)), \quad t \in [t_0, t_\mathrm{f}] \tag{2.4b}$$

$$0 = g(t, y(t), z(t), p, u(t)), \quad t \in [t_0, t_\mathrm{f}] \tag{2.4c}$$

$$0 \leq c^\mathrm{d}(t, y(t), z(t), p, u(t)), \quad t \in [t_0, t_\mathrm{f}] \tag{2.4d}$$

$$0 = \sum_{i=1}^{N^\mathrm{b}} c_i^\mathrm{b}(t_i, y(t_i), z(t_i), p, u_0). \tag{2.4e}$$

The task is to choose control functions u for the states y and z such that the performance index Φ is minimzed. The objective (2.4a) is of Mayer-type, which means it depends on the differential states evaluated at the final time t_f only. The states y and z are required to satisfy the DAE system (2.4b)–(2.4c) with multi-point boundary constraints (2.4e) as well as path constraints (2.4d).

Remark 2.2. The model parameters p do not appear as degrees of freedom in the optimal control problem (2.4). This is because they have to be estimated from measurement data obtained under known experimental conditions u. A model is only valid and suitable for making predictions if we have *reliable* estimates for the model parameters p. In Part II we investigate in detail how to obtain a validated model using optimum experimental design for parameter estimation. In the remainder of this chapter, we omit dependencies on the parameters p to improve readability.

2.2. Direct multiple shooting for optimal control problems

Methods for the solution of the optimal control problem (2.4) can be roughly classified as direct or indirect. In indirect methods, necessary optimality conditions in function space are derived that result in a two-point boundary value problem for state and so-called co-state equations. For the solution of the boundary value problem, multiple shooting can be employed [39, 28]. However, the practical applicability of indirect methods is limited, especially for large systems with path constraints. In direct methods the infinite-dimensional OCP (2.4) is replaced by a finite dimensional NLP which is then solved by suitable numerical methods.

Various direct methods for the solution of problem (2.4) have been proposed, ranging from single shooting approaches [173, 196, 178], where only the control vector is parameterized and the states are repeatedly solved by suitable integration schemes, to direct collocation methods [23, 52, 181, 187, 188], where a full discretization of the states is part of the resulting NLP.

A somewhat hybrid approach is the direct multiple shooting method (DMS) for OC problems. It was first introduced by Bock and Plitt [163, 34] and has been applied and extended subsequently in different contexts, e.g., for multi-stage DAE problems [16, 88, 142] or nonlinear model predictive control [58]. It usually comprises a multiple shooting

parameterization of problem (2.4) and a tailored, structure exploiting sequential quadratic programming method for the solution of the resulting highly structured NLP. Suitable numerical integration schemes are employed to solve the DAE and evaluate derivatives of the states on subintervals, the multiple shooting intervals. In this section, we first give a detailed description of multiple shooting parameterization. Practical issues concerning derivative evaluation and solution of the resulting structured NLP are discussed in Section 2.3.

2.2.1. Problem parameterization

In any direct method, the following infinite-dimensional objects in problem (2.4) must be treated appropriately when replacing it by a finite-dimensional NLP:

- control functions u
- differential and algebraic states y and z
- path constraints c^{d}

In Bock's original method as well as in most subsequent works, states, controls, and path constraints are discretized on a single common time grid. In this work, we explicitly want to consider independent grids as they lead to NLPs that—while providing identical solutions for the controls—may exhibit very different convergence behaviours for different shooting grids. This requires some more complicated notation, but the core ideas of the "classical" DMS method still apply to this more general method.

Control functions
We consider a time grid

$$t_0 = \tau_0^{\mathrm{c}} < \tau_1^{\mathrm{c}} < \cdots < \tau_{N^{\mathrm{c}}}^{\mathrm{c}} = t_{\mathrm{f}} \tag{2.5}$$

on which the control function u is parameterized by means of local basis functions:

$$u(t) = \pi_j(t, q_j), \quad t \in [\tau_j^{\mathrm{c}}, \tau_{j+1}^{\mathrm{c}}), \quad j = 0, \ldots, N^{\mathrm{c}} - 1,$$

where q_j are vectors of finitely many real optimization variables and the functions π_j are typically polynomials of low degree, e.g. linear or constant functions. At the single point t_{f} we also introduce a variable $q_{N^{\mathrm{c}}}$ but require continuity of the control via a linear constraint:

$$u(t_{\mathrm{f}}) = q_{N^{\mathrm{c}}}, \qquad \pi_{N^{\mathrm{c}}-1}(t_{\mathrm{f}}, q_{N^{\mathrm{c}}-1}) - q_{N^{\mathrm{c}}} = 0. \tag{2.6}$$

Depending on parameterization further linear constraints might arise, e.g. to implement a piecewise linear control. We define $q := \left[q_0^T, q_1^T, \ldots, q_{N^{\mathrm{c}}}^T\right]^T \in \mathbb{R}^{n_{\mathrm{q}}}$.

Keep in mind that the choice of the discretization grid (2.5) is often guided by practical considerations rather than an analytical solution of the problem in function space. For example, switping of the control function may be possible only at certain points in time for a given system.

States

In shooting methods, initial value problem solvers are employed to obtain representations of the states y and z for given discretized controls q. In the case of direct single shooting and pure ODEs, the states are regarded as dependent variables, and only q is kept as variable in the optimization problem. Thus the tasks of simulation and optimization are kept separate.

The DMS method for DAEs is a simultaneous strategy to resolve simulation and optimization in parallel. Again, we consider a discretization of the time horizon

$$t_0 = \tau_0^{\mathrm{s}} < \tau_1^{\mathrm{s}} < \cdots < \tau_{N^{\mathrm{s}}}^{\mathrm{s}} = t_{\mathrm{f}}. \tag{2.7}$$

Without loss of generality, we choose the grid points of the shooting grid (2.7) as a subset of the grid points of the control grid (2.5) (in particular $N^{\mathrm{c}} \geq N^{\mathrm{s}}$).

Remark 2.3. If a finer shooting discretization is desirable, e.g. because an error-controlled state integration is only possible on small intervals, we can achieve an equivalent formulation as follows: Assume there is a grid point $\tau_j^{\mathrm{c}} < \tau_{k_j}^{\mathrm{s}} < \tau_{j+1}^{\mathrm{c}}$. Then we introduce an additional grid point $\tau_{k_j}^{\mathrm{c}} = \tau_{k_j}^{\mathrm{s}}$. We choose $\pi_{k_j} = \pi_j$ and introduce variables q_{k_j} for which we explicitly require $q_{k_j} = q_j$ by means of a linear constraint that we add to the NLP.

Let us denote by $\hat{q}_j$ all variables that parameterize the control functions on interval j of the shooting grid and by $\hat{\pi}_j(t,\hat{q}_j)$ the corresponding local basis functions. The number of control discretization intervals that are contained in shooting interval j is denoted by N_j^{c}. Formally, we have:

$$\begin{aligned} q &= \left[\hat{q}_0^T, \hat{q}_1^T, \ldots, \hat{q}_{N^{\mathrm{s}}}^T\right]^T, \\ \hat{q}_j &:= \left[q_{j_0}^T, q_{j_1}^T, \ldots, q_{j_{N_j^{\mathrm{c}}}-1}^T\right]^T, \quad \tau_j^{\mathrm{s}} = \tau_{j_0}^{\mathrm{c}} < \ldots < \tau_{j_{N_j^{\mathrm{c}}}}^{\mathrm{c}} = \tau_{j+1}^{\mathrm{s}}, \quad j = 0, \ldots, N^{\mathrm{s}} - 1, \\ \hat{q}_{N^{\mathrm{s}}} &:= q_{N^{\mathrm{c}}}. \end{aligned} \tag{2.8}$$

On the shooting grid (2.7) we consider the following set of initial value problems with initial values $s^y = \left[s_0^{yT}, s_1^{yT}, \ldots, s_{N^{\mathrm{s}}}^{yT}\right]^T$ for the differential states and $s^z = \left[s_0^{zT}, s_1^{zT}, \ldots, s_{N^{\mathrm{s}}}^{zT}\right]^T$ for the algebraic states that become variables in the optimization problem:

$$\dot{y}(t) = f(t, y(t), z(t), \hat{\pi}_j(t,\hat{q}_j)), \qquad y(\tau_j^{\mathrm{s}}) = s_j^y \tag{2.9a}$$

$$0 = g(t, y(t), z(t), \hat{\pi}_j(t,\hat{q}_j)) - \theta_j(t) g(\tau_j^{\mathrm{s}}, s_j^y, s_j^z, \hat{\pi}_j(\tau_j^{\mathrm{s}}, \hat{q}_j)), \qquad z(\tau_j^{\mathrm{s}}) = s_j^z, \tag{2.9b}$$

where θ_j is a fast decreasing damping function with $\theta_j(\tau_j^{\mathrm{s}}) = 1$, e.g.

$$\theta_j(t) = \exp\left(-\alpha_j \frac{t - \tau_j^{\mathrm{s}}}{\tau_{j+1}^{\mathrm{s}} - \tau_j^{\mathrm{s}}}\right), \quad \alpha_j > 0.$$

This relaxed formulation was proposed in [32] and implies that the algebraic condition (2.9b) is automatically consistent for any initial values s_j^z. That means a DAE solver does not need to solve the (nonlinear) algebraic condition in every iteration of the optimization algorithm to find feasible initial values. Instead, nonlinear algebraic consistency conditions

$$0 = g(\tau_j^{\mathrm{s}}, s_j^y, s_j^z, \hat{\pi}_j(\tau_j^{\mathrm{s}}, \hat{q}_j)), \quad j = 0, \ldots, N^{\mathrm{s}}$$

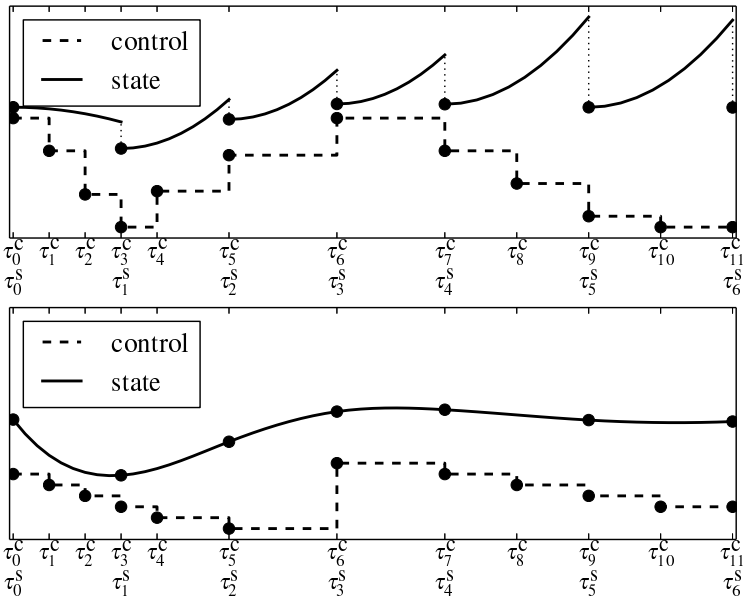


Figure 2.1.: Concept of DMS for one state and one piecewise constant control. In the upper plot the continuity conditions are violated (vertical dotted lines), in the lower plot they are satisfied. Note that the control grid τ^{c} is finer than the shooting grid τ^{s} so the control is sometimes allowed to switch within a shooting interval.

are added to the optimization problem to ensure that the original DAE is solved at the solution of the optimization problem.

Note that the DAEs (2.9) are solved independently on the smaller time intervals $[\tau_j^{\mathrm{s}}, \tau_{j+1}^{\mathrm{s}}]$ as the initial values (s_j^y, s_j^z) are variables of the optimization problem. To ensure equivalence to the original system (2.1), continuity conditions are added to the optimization problem for every shooting interval. Let us denote by $y(\cdot; s_j^y, s_j^z, \hat{q}_j)$ and $z(\cdot; s_j^y, s_j^z, \hat{q}_j)$ a representation of the solution to problem (2.9) on the intervals $[\tau_j^{\mathrm{s}}, \tau_{j+1}^{\mathrm{s}}]$ for given $(s_j^y, s_j^z, \hat{q}_j)$. Then the continuity conditions read as:

$$y(\tau_{j+1}^{\mathrm{s}}; s_j^y, s_j^z, \hat{q}_j) = s_{j+1}^y, \quad j = 0, \ldots, N^{\mathrm{s}} - 1.$$

Figure 2.1 illustrates the concept of DMS. Note how we use different grids for controls and states. A special case is of course to choose the same grid for both. However, in our experience, the decoupling of grids provides greater flexibility: sometimes a smaller number of shooting intervals can greatly accelerate convergence of an SQP method for problems where a relatively fine discretization of the controls is desirable.

In this and the following chapters which deal with experimental design problems we will use the letter x to denote the combined differential and algebraic states, i.e.

$$x := \begin{bmatrix} y \\ z \end{bmatrix},$$

as this should not create any ambiguity with the usage of x as optimization variables in the context of nonlinear programming and simplifies the notation. Also, we use s^x for the corresponding shooting variables:

$$s^x = \begin{bmatrix} s_0^x \\ \vdots \\ s_{N^\mathrm{s}}^x \end{bmatrix} := \begin{bmatrix} s_0^y \\ s_0^z \\ \vdots \\ s_{N^\mathrm{s}}^y \\ s_{N^\mathrm{s}}^z \end{bmatrix}. \tag{2.10}$$

Path constraints

Path constraints of the form (2.4d) are generally required to hold everywhere on $[t_0, t_\mathrm{f}]$ which makes the constraints infinite-dimensional. In DMS, we evaluate the path constraints on a grid of finitely many points:

$$t_0 = \tau_0^\mathrm{d} < \tau_1^\mathrm{d} < \cdots < \tau_{N^\mathrm{d}}^\mathrm{d} = t_\mathrm{f}. \tag{2.11}$$

Depending on the choice of the time grid, the constraints might be violated in between grid points. There are strategies how to adaptively add checkpoints, see, e.g., [164]. For the scope of this text, however, we will assume that grid (2.11) is always chosen sufficiently dense.

Again, we assume that all points of the shooting grid (2.7) are also part of grid (2.11). We assign the points (2.11) to the multiple shooting grid by introducing the following notation:

$$\begin{aligned} &\tau_j^\mathrm{s} = \tau_{j_0}^\mathrm{d} < \tau_{j_1}^\mathrm{d} < \cdots < \tau_{j_{N_j^\mathrm{d}}}^\mathrm{d} = \tau_{j+1}^\mathrm{s}, \quad j = 0, \ldots, N^\mathrm{s} - 1 \\ &\tau_j^\mathrm{s} = \tau_{j_0}^\mathrm{d}, \quad j = N^\mathrm{s}, \end{aligned}$$

where N_j^d denotes the number of constraint checkpoints on shooting interval j. The discretized path constraints read as

$$\begin{aligned} 0 \leq c^\mathrm{d}(\tau_{j_i}^\mathrm{d}, y(\tau_{j_i}^\mathrm{d}; s_j^x, \hat{q}_j), z(\tau_{j_i}^\mathrm{d}; s_j^x, \hat{q}_j), \hat{\pi}_j(\tau_{j_i}^\mathrm{d}, \hat{q}_j)), \quad & j = 0, \ldots, N^\mathrm{s}, \\ & i = 0, \ldots, N_j^\mathrm{d} - 1. \end{aligned} \tag{2.12}$$

Note that (2.12) has the following simple form if the same grids for multiple shooting and the discretization of path constraints are used:

$$0 \leq c^\mathrm{d}(\tau_j^\mathrm{s}, s_j^x, \hat{\pi}_j(\tau_j^\mathrm{s}, \hat{q}_j)), \quad j = 0, \ldots, N^\mathrm{s}.$$

Multi-point boundary constraints

We split up the sum in the linearly coupled multi-point boundary constraints (2.4e) according to the multiple shooting grid and plug in the solution representation depending on discretized controls q and shooting variables s^y and s^z. Equation (2.4e) is then replaced by:

$$0 = \sum_{j=0}^{N^\mathrm{s}} \sum_{\substack{i: \\ \tau_j^\mathrm{s} \leq t_i < \tau_{j+1}^\mathrm{s}}} c_i^\mathrm{b}(t_i, y(t_i; s_j^x, \hat{q}_j), z(t_i; s_j^x, \hat{q}_j), \hat{\pi}_j(t_i, \hat{q}_j)).$$

We formally set $\tau^{\mathrm{s}}_{N^{\mathrm{s}}+1} = \infty$ to avoid additional notation.

Structured NLP

We have now addressed all constraints of the OC problem and can formulate the resulting structured NLP as follows:

$$\min_{s^x, q} \quad \Phi(s^x_{N^{\mathrm{s}}}, \hat{q}_{N^{\mathrm{s}}}) \tag{2.13a}$$

$$\text{s.t.} \quad 0 = y(\tau^{\mathrm{s}}_0; \hat{q}_0) - s^y_0 \tag{2.13b}$$

$$0 = y(\tau^{\mathrm{s}}_{j+1}; s^x_j, \hat{q}_j) - s^y_{j+1}, \qquad j = 0, \ldots, N^{\mathrm{s}} - 1 \tag{2.13c}$$

$$0 = g(\tau^{\mathrm{s}}_j, s^x_j, \hat{\pi}_j(\tau^{\mathrm{s}}_j, \hat{q}_j)), \qquad j = 0, \ldots, N^{\mathrm{s}} \tag{2.13d}$$

$$0 \leq c^{\mathrm{d}}(\tau^{\mathrm{d}}_{j_i}, x(\tau^{\mathrm{d}}_{j_i}; s^x_j, \hat{q}_j), \hat{\pi}_j(\tau^{\mathrm{d}}_{j_i}, \hat{q}_j)), \qquad j = 0, \ldots, N^{\mathrm{s}} \quad i = 0, \ldots, N^{\mathrm{d}}_j - 1 \tag{2.13e}$$

$$0 = \sum_{j=0}^{N^{\mathrm{s}}} \sum_{\substack{i: \\ \tau^{\mathrm{s}}_j \leq t_i < \tau^{\mathrm{s}}_{j+1}}} c^{\mathrm{b}}_i(t_i, x(t_i; s^x_j, \hat{q}_j), \hat{\pi}_j(t_i, \hat{q}_j)) \tag{2.13f}$$

$$0 \leq L(s^x_0, \hat{q}_0, \ldots, s^x_{N^{\mathrm{s}}}, \hat{q}_{N^{\mathrm{s}}}), \tag{2.13g}$$

where (2.13b) are the ODE initial conditions and (2.13c) and (2.13d) are the continuity and consistency conditions that guarantee the solution of the original DAE (2.1) at the solution of the optimization problem. Furthermore, we have the discretized path constraints (2.13e) and multi-point boundary constraints (2.13f). The number of variables and constraints depend on the selection of the grids τ^{s}, τ^{c}, and τ^{d}. A finer grid τ^{c} increases the number of variables, a finer grid τ^{d} increases the number of nonlinear constraints. Refinement of the multiple shooting grid τ^{s} simultaneously increases the number of variables and constraints. An important difference, however, is that different grids τ^{c} and τ^{d} usually lead to different solutions of problem (2.13), while the choice of the shooting grid τ^{s} does not affect the solution but only changes the dimension and structure of the NLP and possibly the convergence behavior of the NLP method.

For the following discussion, we group the variables according to their shooting nodes and define

$$\xi := \begin{bmatrix} \xi_0 \\ \vdots \\ \xi_{N^{\mathrm{s}}} \end{bmatrix} \in \mathbb{R}^{n_\xi}, \qquad \xi_j := \begin{bmatrix} s^y_j \\ s^z_j \\ \hat{q}_j \end{bmatrix}, \quad j = 0, \ldots, N^{\mathrm{s}}.$$

All linear constraints that may arise during parameterization, such as (2.6) or additional continuity constraints to implement a piecewise linear control, are subsumed in (2.13g) with a *linear* function L that we identify with a matrix $L \in \mathbb{R}^{n_{\mathrm{L}} \times n_\xi}$.

2.3. Direct multiple shooting: practical issues

In this section, we discuss three important issues in the practical application of DMS. First, how to efficiently evaluate states and derivatives using the principle of internal numerical differentiation; second, the structure of derivatives that are of importance for the optimization algorithm; and third, how SQP methods can exploit problem structures that arise in DMS.

2.3.1. Function and derivative evaluation

When we solve problem (2.13) with a Newton type method, the objective (2.13a) and constraints (2.13b)–(2.13g) as well as their derivatives with respect to s^y, s^z, and q must be evaluated at different points in the variable space. The functions Φ, g, c^{d}, and c_i^{b} are explicitly formulated as analytic functions and their derivatives can be evaluated using *algorithmic differentiation (AD)* [112].

The states y and z are given implicitly as solutions of the DAE (2.9). Solution representations $y(\cdot; s_j^y, s_j^z, \hat{q}_j)$ and $z(\cdot; s_j^y, s_j^z, \hat{q}_j)$ that are needed to evaluate the continuity constraints (2.13c) as well as path constraints (2.13e) and boundary constraints (2.13f) are usually obtained by suitable numerical schemes. In the case of chemical reaction systems, for example, the underlying differential equations are usually stiff and *backwards differentiation formula (BDF)* methods are the methods of choice [14]. For non-stiff systems one may use, e.g., Runge–Kutta methods. Advanced methods usually employ error estimators to choose stepsizes and orders adaptively. Hence, they are able to generate solutions that approximate the analytical solution within a given error bound while still maintaining a high degree of efficiency by avoiding unnecessarily small stepsizes. For a detailed discussion of numerical methods for ODE and DAE we refer to the literature [8, 9, 115, 116]. Derivatives of the DAE solutions play an important role in the numerical treatment of OC and OED and are discussed next.

The principle of internal numerical differentiation

When we evaluate derivatives of the constraints, we need derivatives of the solution representations $y(\cdot; s_j^y, s_j^z, \hat{q}_j)$ and $z(\cdot; s_j^y, s_j^z, \hat{q}_j)$ in the direction of the initial values s_j^y and s_j^z as well as the discretized controls $\hat{q}_j$. We sometimes call these derivatives *sensitivities* of the states to distinguish them from derivatives of analytic functions whose evaluation can be easily handled by applying AD. For the efficient computation of these sensitivities, we follow the principle of *internal numerical differentiation (IND)* introduced by Bock [29, 30]. The idea is to obtain state derivatives by differentiating the numerical scheme for the nominal DAE system. Because this scheme is usually generated adaptively, we freeze all adaptive components such as stepsizes, orders, or iteration matrices. Then this numerical scheme can be interpreted as a sequence of differentiable mappings, each leading from the state solution at one point of the discretization grid to the next. This approach has the advantage, among other things, that by construction, it generates the exact numerical derivative of the approximate solution of the nominal DAE.

An important property of IND is that differentiating the integration scheme following the

principle of IND is equivalent to numerically solving the nominal DAE system augmented by the corresponding variational differential equations. They are defined as follows:

Definition 2.4 (Variational DAE)**.** For a (nominal) DAE system of the form

$$\begin{aligned} \dot{y}(t) &= f(t,y,z,q), & y(\tau_0) &= s^y, \\ 0 &= g(t,y,z,q) - \theta(t) g(\tau_0, s^y, s^z, q), & z(\tau_0) &= s^z, \end{aligned}$$

the (forward) variational differential algebraic equations (VDAE) for $y_\mathrm{s} := \frac{\mathrm{d}y}{\mathrm{d}s^x}$ and $z_\mathrm{s} := \frac{\mathrm{d}z}{\mathrm{d}s^x}$, are given by:

$$\dot{y_\mathrm{s}}(t) = \frac{\partial f}{\partial y} \cdot y_\mathrm{s}(t) + \frac{\partial f}{\partial z} \cdot z_\mathrm{s}(t), \qquad y_\mathrm{s}(\tau_0) = \begin{bmatrix} I_{n_\mathrm{y}} & 0 \end{bmatrix}, \tag{2.14a}$$

$$0 = \frac{\partial g}{\partial y} \cdot y_\mathrm{s}(t) + \frac{\partial g}{\partial z} \cdot z_\mathrm{s}(t) - \theta(t) \left[\frac{\partial g_0}{\partial y}, \frac{\partial g_0}{\partial z} \right], \qquad z_\mathrm{s}(\tau_0) = \begin{bmatrix} 0 & I_{n_\mathrm{z}} \end{bmatrix}, \tag{2.14b}$$

where we omit the arguments of $\frac{\partial f}{\partial(\cdot)}$ and $\frac{\partial g}{\partial(\cdot)}$ and $\frac{\partial g_0}{\partial(\cdot)}$ denotes the evaluation of $\frac{\partial g}{\partial(\cdot)}$ at τ_0. The VDAE for y_q and z_q are given by:

$$\dot{y_\mathrm{q}}(t) = \frac{\partial f}{\partial y} \cdot y_\mathrm{q}(t) + \frac{\partial f}{\partial z} \cdot z_\mathrm{q}(t) + \frac{\partial f}{\partial q}, \qquad y_\mathrm{q}(\tau_0) = 0, \tag{2.15a}$$

$$0 = \frac{\partial g}{\partial y} \cdot y_\mathrm{q}(t) + \frac{\partial g}{\partial z} \cdot z_\mathrm{q}(t) + \frac{\partial g}{\partial q} - \theta(t) \frac{\partial g_0}{\partial q}, \qquad z_\mathrm{q}(\tau_0) = 0. \tag{2.15b}$$

For derivatives of higher order the corresponding VDAEs can be obtained in a similar way by linearizing the first order VDAEs (2.14) and (2.15), respectively.

While solving the augmented system and differentiating the integration scheme yield the same solution for the sensitivities, the principle of IND allows to re-use certain *internal* information of the integration scheme for the nominal DAE to obtain efficient schemes for the combined nominal and variational DAE system.

For efficient IND schemes, the computational effort for evaluating sensitivities is mainly governed by evaluation of the derivatives of the DAE right-hand side [4]. Depending on the optimization algorithm, *forward* or *adjoint* sensitivities of the states may be computed. In this work, we restrict ourselves to forward sensitivities. In this case, derivatives of the DAE right-hand side should be evaluated using the forward mode of AD. The theory of AD gives a worst-case upper bound for the evaluation of n_dir directional derivatives of $1 + 1.5 n_\mathrm{dir}$ times the effort of evaluating the function. From this, we immediately obtain a theoretical upper bound for the evaluation of DAE sensitivities: Computing n_dir sensitivities x_s costs no more than $1 + 1.5 n_\mathrm{dir}$ times the nominal integration, and n_dir sensitivities x_q cost no more than $2 + 3 n_\mathrm{dir}$ times the nominal integration. A detailed discussion of IND exceeds the scope of this thesis and we refer to the comprehensive treatment in [4].

2.3.2. Structured derivatives

The DMS discretization imposes specific structures on the problem derivatives. These structures need to be taken into account by an SQP method to solve problem (2.13) efficiently.

Banded constraint Jacobian

For a given shooting interval j, each of the constraints (2.13c), (2.13d), and (2.13e) depends in a nonlinear way on the corresponding variables ξ_j only. Furthermore, there is a linear dependence on s^y_{j+1} in the continuity constraints for interval j. This leads to a banded structure in the upper part of the constraint Jacobian. The linearly coupled multi-point boundary constraints (2.13f) and the linear constraints (2.13g) yield a dense block with n_d and n_L rows, respectively. The full constraint Jacobian has the following structure:

$$\left[\begin{array}{ccccccccccc}
-I_{n_\mathrm{y}} & 0 & \frac{\mathrm{d}y_0}{\mathrm{d}\hat{q}_0} & & & & & & & & \\
\frac{\mathrm{d}g_0}{\mathrm{d}s^y_0} & \frac{\mathrm{d}g_0}{\mathrm{d}s^z_0} & \frac{\mathrm{d}g_0}{\mathrm{d}\hat{q}_0} & & & & & & & & \\
\frac{\mathrm{d}c^\mathrm{d}_{0_i}}{\mathrm{d}s^y_0} & \frac{\mathrm{d}c^\mathrm{d}_{0_i}}{\mathrm{d}s^z_0} & \frac{\mathrm{d}c^\mathrm{d}_{0_i}}{\mathrm{d}\hat{q}_0} & & & & & & & & \\
\frac{\mathrm{d}y_1}{\mathrm{d}s^y_0} & \frac{\mathrm{d}y_1}{\mathrm{d}s^z_0} & \frac{\mathrm{d}y_1}{\mathrm{d}\hat{q}_0} & -I_{n_\mathrm{y}} & & & & & & & \\
 & & & \ddots & & & & & & & \\
 & & & & \ddots & & & & & & \\
 & & & & & \frac{\mathrm{d}g_{N^\mathrm{s}-1}}{\mathrm{d}s^y_{N^\mathrm{s}-1}} & \frac{\mathrm{d}g_{N^\mathrm{s}-1}}{\mathrm{d}s^z_{N^\mathrm{s}-1}} & \frac{\mathrm{d}g_{N^\mathrm{s}-1}}{\mathrm{d}\hat{q}_{N^\mathrm{s}-1}} & & & & \\
 & & & & & \frac{\mathrm{d}c^\mathrm{d}_{N^\mathrm{s}-1_i}}{\mathrm{d}s^y_{N^\mathrm{s}-1}} & \frac{\mathrm{d}c^\mathrm{d}_{N^\mathrm{s}-1_i}}{\mathrm{d}s^z_{N^\mathrm{s}-1}} & \frac{\mathrm{d}c^\mathrm{d}_{N^\mathrm{s}-1_i}}{\mathrm{d}\hat{q}_{N^\mathrm{s}-1}} & & & & \\
 & & & & & \frac{\mathrm{d}y_{N^\mathrm{s}}}{\mathrm{d}s^y_{N^\mathrm{s}-1}} & \frac{\mathrm{d}y_{N^\mathrm{s}}}{\mathrm{d}s^z_{N^\mathrm{s}-1}} & \frac{\mathrm{d}y_{N^\mathrm{s}}}{\mathrm{d}\hat{q}_{N^\mathrm{s}-1}} & -I_{n_\mathrm{y}} & & & \\
 & & & & & & & & \frac{\mathrm{d}g}{\mathrm{d}s^y_{N^\mathrm{s}}} & \frac{\mathrm{d}g}{\mathrm{d}s^z_{N^\mathrm{s}}} & \frac{\mathrm{d}g}{\mathrm{d}\hat{q}_{N^\mathrm{s}}} \\
 & & & & & & & & \frac{\mathrm{d}c^\mathrm{d}_{N^\mathrm{s}}}{\mathrm{d}s^y_{N^\mathrm{s}}} & \frac{\mathrm{d}c^\mathrm{d}_{N^\mathrm{s}}}{\mathrm{d}s^z_{N^\mathrm{s}}} & \frac{\mathrm{d}c^\mathrm{d}_{N^\mathrm{s}}}{\mathrm{d}\hat{q}_{N^\mathrm{s}}} \\
\frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}s^y_0} & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}s^z_0} & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}\hat{q}_0} & \cdots & \cdots & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}s^y_{N^\mathrm{s}-1}} & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}s^z_{N^\mathrm{s}-1}} & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}\hat{q}_{N^\mathrm{s}-1}} & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}s^y_{N^\mathrm{s}}} & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}s^z_{N^\mathrm{s}}} & \frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}\hat{q}_{N^\mathrm{s}}} \\
\multicolumn{3}{c}{\underline{\qquad\qquad\qquad}} & & & \multicolumn{3}{c}{L} & \multicolumn{3}{c}{\underline{\qquad\qquad\qquad}} & \\
\end{array}\right], \tag{2.16}$$

where we use the following abbreviations:

$$\begin{aligned}
\frac{\mathrm{d}g_j}{\mathrm{d}(\cdot)_j} &:= \frac{\mathrm{d}g}{\mathrm{d}(\cdot)_j}(\tau^\mathrm{s}_j, s^x_j, \hat{\pi}_j(\tau^\mathrm{s}_j, \hat{q}_j)) \\
\frac{\mathrm{d}c^\mathrm{d}_{j_i}}{\mathrm{d}(\cdot)_j} &:= \frac{\mathrm{d}c^\mathrm{d}}{\mathrm{d}(\cdot)_j}(\tau^\mathrm{d}_{j_i}, x(\tau^\mathrm{d}_{j_i}), \hat{\pi}_j(\tau^\mathrm{d}_{j_i}, \hat{q}_j)), \quad i = 0, \ldots, N^\mathrm{d}_j - 1 \\
\frac{\mathrm{d}y_{j+1}}{\mathrm{d}(\cdot)_j} &:= \frac{\mathrm{d}y}{\mathrm{d}(\cdot)_j}(\tau^\mathrm{s}_{j+1}; s^x_j, \hat{q}_j) \\
\frac{\mathrm{d}\sum c^\mathrm{b}_i}{\mathrm{d}(\cdot)_j} &:= \sum_{\substack{i: \\ \tau^\mathrm{s}_j \le t_i < \tau^\mathrm{s}_{j+1}}} \frac{\mathrm{d}c^\mathrm{b}_i}{\mathrm{d}(\cdot)_j}(t_i, x(t_i), \hat{\pi}_j(t_i, \hat{q}_j)).
\end{aligned}$$

Note that we use the total differential $\frac{\mathrm{d}}{\mathrm{d}(\cdot)}$ here to indicate explicit and implicit dependencies on the variables, e.g.,

$$\frac{\mathrm{d}c^\mathrm{d}}{\mathrm{d}\hat{q}_j}(\tau^\mathrm{d}_{j_i}, x(\tau^\mathrm{d}_{j_i}), \hat{\pi}_j(\tau^\mathrm{d}_{j_i}, \hat{q}_j)) = \frac{\partial c^\mathrm{d}}{\partial x}\frac{\partial x}{\partial \hat{q}_j}(\tau^\mathrm{d}_{j_i}; s^x_j, \hat{q}_j) + \frac{\partial c^\mathrm{d}}{\partial \hat{\pi}_j}\frac{\partial \hat{\pi}_j}{\partial \hat{q}_j}(\tau^\mathrm{d}_{j_i}, \hat{q}_j).$$

In practice, expressions such as $\frac{\partial c^{\mathrm{d}}}{\partial x}$ can be easily evaluated using algorithmic differentiation, while $\frac{\partial x}{\partial \hat{q}_j}$ usually has to be supplied by the integrator, see the discussion about IND in the precious section.

The size of each block corresponding to the continuity-, consistency-, and path constraints on one shooting interval can vary for different shooting intervals depending on the control parameterization and the evaluation of path constraints: The grid τ^{c} determines the number of control variables $\hat{q}_j$ and the grid τ^{d} determines the number of constraints in addition to the continuity and consistency constraints for the current interval.

Block-diagonal Hessian of the Lagrangian

The Lagrangian of problem (2.13) is a special case of a partially separable function (see [113] for an exact definition of partial separability). That means it can be written as a sum of *element functions* that depend only on a subset of the variables:

$$\mathcal{L}(\xi,\lambda) = \Phi(v_{N^{\mathrm{s}}}) - \sum_{j=0}^{N^{\mathrm{s}}} \left(\lambda_j^T c_j(\xi_j,\xi_{j+1}) + \lambda_{c^{\mathrm{b}}}^T \sum_i c_i^{\mathrm{b}}(\xi_j) \right) - \lambda_L^T L\xi, \tag{2.17}$$

where c_j combines continuity, consistency, and path constraints for node j and λ_j is the corresponding subvector of the Lagrange multipliers. The dependency on ξ_{j+1} for the constraint set c_j is only due to the *linear* coupling in the continuity constraints and thus equation (2.17) can be rewritten as

$$\mathcal{L}(\xi,\lambda) = \sum_{j=0}^{N^{\mathrm{s}}} \mathcal{L}_j(\xi_j,\lambda).$$

Therefore, it holds that

$$\nabla^2_{\xi_i\xi_j}\mathcal{L}(\xi,\lambda) = 0, \quad i \neq j,$$

which means that the Hessian of the Lagrangian has diagonal block structure according to the variable partition ξ_j:

$$\nabla^2_{\xi\xi}\mathcal{L} = \begin{bmatrix} \nabla^2_{\xi_0\xi_0}\mathcal{L} & & & \\ & \nabla^2_{\xi_1\xi_1}\mathcal{L} & & \\ & & \ddots & \\ & & & \nabla^2_{\xi_{N^{\mathrm{s}}}\xi_{N^{\mathrm{s}}}}\mathcal{L} \end{bmatrix}. \tag{2.18}$$

2.3.3. Solution of the NLP

In principle, problem (2.13) could be solved by any general purpose SQP method. However, a crucial factor for the efficiency of DMS is the use of tailored NLP solvers that exploit the problem structure outlined above. Two features that are of special importance are partitioned

Quasi-Newton approximations of the Hessian and an efficient solution of the sparse quadratic subproblem that is formulated in every iteration of an SQP method.

Partitioned Quasi-Newton updates

In many SQP methods, Quasi-Newton update formulae such as the BFGS formula are employed instead of the exact Hessian to obtain approximations that are cheap to compute and that circumvent the difficulty of nonconvex QPs by using appropriate modifications that guarantee positive definiteness of the approximation. NLPs resulting from shooting methods particularly benefit from this approach because their Hessian includes second order derivatives of the states that are usually very time consuming to evaluate.

Starting with an initial approximation $B^{[0]}$, e.g. a positive definite diagonal matrix, a new approximation $B^{[k+1]}$ is obtained by a low-rank modification to the current approximation $B^{[k]}$ using the vectors

$$\begin{aligned}\gamma^{[k]} &= \mathcal{L}(\xi^{[k+1]}, \lambda^{[k+1]}) - \mathcal{L}(\xi^{[k]}, \lambda^{[k+1]}) \\ \delta^{[k]} &= \xi^{[k+1]} - \xi^{[k]}.\end{aligned}$$

The two most popular updates, the BFGS and the SR1 update, yield a rank-2 and rank-1 correction, respectively. However, standard use of the updates does not take into account the block diagonal structure of the Hessian (2.18) that results from the partial separability of the Lagrangian as laid out in the previous section.

On the one hand, this leads to dense QPs that can be relatively large, depending on the multiple shooting discretization. On the other hand, the low-rank corrections can result in a high number of SQP iterations as only little new curvature information can be incorporated into the quadratic model during each iteration.

Bock and Plitt [163, 34] suggest to use *partitioned* Quasi-Newton updates that retain the diagonal block structure and lead to high-rank corrections of the current approximation $B^{[k]}$. The idea is to maintain $N^{\mathrm{s}}+1$ independent approximations B_j, $j = 0, \ldots, N^{\mathrm{s}}$ for the diagonal blocks in (2.18). To this end, the update formulae are simply applied to the appropriate subvectors of $\gamma^{[k]}$ and $\delta^{[k]}$:

$$\begin{aligned}\gamma_j^{[k]} &= \mathcal{L}(\xi_j^{[k+1]}, \lambda^{[k+1]}) - \mathcal{L}(\xi_j^{[k]}, \lambda^{[k+1]}) \\ \delta_j^{[k]} &= \xi_j^{[k+1]} - \xi_j^{[k]}.\end{aligned}$$

In Chapter 7 we will describe in detail different Quasi-Newton updates and their application to diagonal block structured Hessians.

Solution of sparse subproblems

A crucial step of every SQP algorithm for problem (2.13) is the solution of the large, block-structured QP subproblems that arise due to the multiple shooting discretization. Figure 2.2 shows the KKT matrix with a Hessian of the form (2.18) and a constraint Jacobian of the form (2.16) that illustrates the structure of the problem.

A common approach in DMS is to use a *condensing* algorithm as preprocessing step that makes use of the structure induced by the continuity constraints (2.13c) to eliminate the

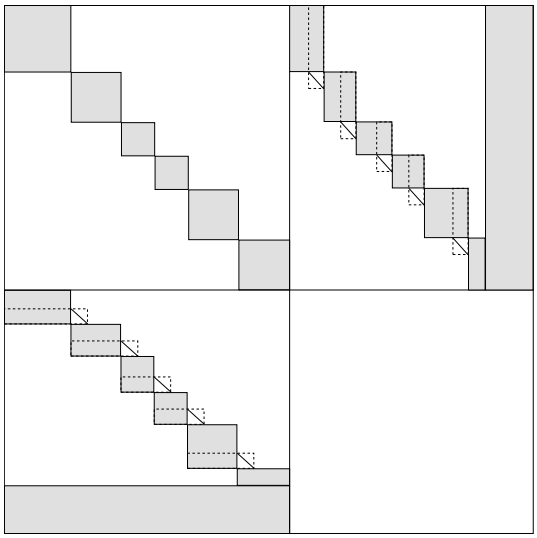


Figure 2.2.: Structured KKT matrix with block diagonal Hessian and banded Jacobian.

shooting variables s_j^y, $j = 1, \ldots, N^s$ and the corresponding constraints before the solution of the QP. The resulting smaller and densely populated QP is then solved by a suitable QP solver and a solution for s_j^y as well as the multipliers for the eliminated continuity constraints are recovered by a simple recursion. This approach has been described in [31, 34, 163, 159] and has been extended to the case where many algebraic states are present in [140, 142].

A related method that is especially well-suited for problems involving many control variables but comparably few states is described by Kirches et al. [131, 133, 132], based on ideas by Steinbach [186]. This method is not implemented as a preprocessing step but exploits the block structure of the KKT matrix directly by a tailored factorization that can be incorporated in a specialized QP solver. The factorization can be updated efficiently as the QP working set iterations proceed.

An alternative to condensing-based block-QP solvers are *reduced* SQP methods [174, 181, 189] where the step is partitioned into a null space direction and a range space direction that are computed by projecting on the constraints. However, this does not allow the efficient approximation of the Hessian by blockwise quasi-Newton updates. In [91] a modified QP subproblem is solved where the structure of the continuity constraints is exploited but a dense quasi-Newton update is maintained for the sparse, block-diagonal Hessian.

Both approaches, condensing and reduced methods, require close integration of problem evaluation and SQP method, that means the SQP method needs to be aware of the specific constraint structure. In Chapter 8 we present an alternative approach based on a sparse symmetric indefinite LDL^T factorization of the KKT matrix. It has the advantage that few assumptions need to be made concerning the problem structure and additional sparsity, that may arise in more specific problem classes such as optimum experimental design problems parameterized by multiple shooting, is exploited.

Part II.

Optimum experimental design for parameter estimation

Chapter 3.

Formulation of optimum experimental design problems

In Chapter 2, we introduced dynamic models described by DAEs of the form

$$\dot{y}(t) = f(t, y(t), z(t), p, u(t)), \qquad t \in [t_0, t_f] \tag{3.1a}$$

$$0 = g(t, y(t), z(t), p, u(t)), \qquad t \in [t_0, t_f] \tag{3.1b}$$

$$0 = \sum_{i=1}^{N^b} c_i^b(t_i, y(t_i), z(t_i), p, u_0). \tag{3.1c}$$

Here, we assume that the boundary constraints (3.1c) are formulated such that for a given p there exist unique solutions y and z that satisfy (3.1c). This is to ensure that all parameters p are random variables that depend only on the data.

Before we can use a model to optimize a process we need to make sure that the model describes the process quantitatively. Two types of errors can occur: Systematic or structural errors mean that the model equations do not describe all aspects of the process. For example, a major reaction taking place during a chemical process is not represented by the appropriate terms in the DAE. The second type of error is the statistical error that is due to *uncertainty* in the parameters p and thus leads to false predictions of the model.

In this work, we exclude structural errors and always make the assumption that the models are structurally correct, i.e., the model describes the process correctly if the true parameter values were known. We focus on the task of significantly identifying the model parameters p to validate the model. To this end, we perform experiments and take measurements. As measurements are subject to statistical errors, they result in uncertain estimates $\hat{p}$ for the parameters. The relation between measurement error and parameter uncertainty is typically nonlinear and poorly designed experiments can lead to highly uncertain parameter estimates.

Optimum experimental design (OED) for parameter estimation[1] is used to select an experimental setup such that the corresponding measurements allow to estimate the parameters with minimum uncertainty. In this way, a model can be validated using as little experimental effort as possible. In this chapter, we formally describe the problem of optimum experimental design for nonlinear, dynamic models of the form (3.1) as part of the parameter identification process.

[1] A related problem not covered in this work is experimental design for model discrimination where the "best" model among several rival models is to be chosen, see [83, §3] and the references therein for a survey.

OED for statistical models has been studied for several decades and there exist a number of textbooks on the subject, for example, [10, 36, 67, 167]. Nonlinear OED for parameter estimation subject to DAE models has received considerable attention over the last years, see, for example, [11, 15, 66, 134, 83, 146, 177]. The specific problem formulations in the literature vary, e.g., with regard to the dynamic system, objective function, or measurement design. We follow the approach presented in [15, 134] to derive the OED problem for parameter estimation in DAE models in a general form. We also present a novel view on the relation between covariance matrices of constrained and unconstrained parameter estimation problems. We conclude the chapter with some problem extensions, namely the design of multiple experiments, OED for key performance indicators and continuous measurements.

3.1. Parameter estimation

In this section, we describe a maximum likelihood parameter estimation problem constrained by DAE that forms the underlying problem class for the OED problems in this work. Parameter estimation problems and numerical methods for their solution are treated in [31, 179].

3.1.1. Problem formulation

Estimation of the model parameters p, sometimes referred to as parameter identification or model calibration, is the process of determining an estimate $\hat{p}$ for the model parameters based on measurement data. In this work, we are interested in the *maximum likelihood estimator* $\hat{p}$. Loosely speaking, $\hat{p}$ are the parameters with the highest probability of giving rise to a given set of observations.

Let us assume that observations $\eta_1, \eta_2, \ldots, \eta_M \in \mathbb{R}$ are available at sampling times $t_1, t_2, \ldots, t_M$. We assume that the corresponding measurement errors $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_M$ are random variables that are independently normally distributed with zero mean and standard deviations $\sigma_1, \sigma_2, \ldots, \sigma_M$. We denote by

$$h_i(t_i, y(t_i), z(t_i), p, u_0) \in \mathbb{R}, \quad i = 1, \ldots, M$$

the corresponding *model response*. If we assume that the model is structurally correct and errors arise only due to inaccuracies in the measurement process, the following relation holds:

$$h_i(t_i, y(t_i), z(t_i), p^\star, u_0) = \eta_i + \varepsilon_i, \quad i = 1, \ldots, M,$$

where $p^\star$ are the true—but inaccessible—parameter values and y and z are the corresponding states. Following the assumptions above, the maximum likelihood parameter estimation

problem can be written in the form of the following nonlinear least squares problem [13]:

$$\min_{p,y,z} \quad \frac{1}{2}\sum_{i=1}^{M}\left(\frac{h_i(t_i,y(t_i),z(t_i),p,u_0)-\eta_i}{\sigma_i}\right)^2 \tag{3.2a}$$

$$\text{s.t.} \quad \dot{y}(t) = f(t,y(t),z(t),p,u(t)), \qquad t\in[t_0,t_\mathrm{f}] \tag{3.2b}$$

$$0 = g(t,y(t),z(t),p,u(t)), \qquad t\in[t_0,t_\mathrm{f}] \tag{3.2c}$$

$$0 = \sum_{i=1}^{N^\mathrm{b}} c_i^\mathrm{b}(t_i,y(t_i),z(t_i),p,u_0). \tag{3.2d}$$

Note that for this problem the controls $u(\cdot)$ are fixed, representing the experimental conditions under which the measurement data η_i were obtained.

3.1.2. Transformation to finite-dimensional problem

The DAE constrained nonlinear least squares problem (3.2) can be solved efficiently by DMS in combination with a generalized Gauss–Newton method, see [31, 179]. Similar to the case of optimal control problems as introduced in Chapter 2, with DMS for parameter estimation problems, the states are parameterized to obtain a finite-dimensional, equality-constrained nonlinear least-squares problem, by introducing additional variables $s\in\mathbb{R}^{n_\mathrm{s}}$ and constraints. Together with the boundary constraints (3.2d), that may also be transformed by the parameterization, we combine all constraints into a linearly coupled constraint with element functions c_i^pe evaluated at points t_i, $i=1,\dots,N^\mathrm{pe}$. The resulting constraint is also of dimension n_s, given a suitable parameterization. This represents the assumption that the n_p model parameters p are random variables that can be estimated from the data alone. Introducing the notation

$$F_1:\mathbb{R}^{n_\mathrm{p}+n_\mathrm{s}}\longrightarrow\mathbb{R}^M, \qquad F_1(p,s) := \left(\frac{h_i(t_i,y(t_i;p,s),z(t_i;p,s),p,u_0)-\eta_i}{\sigma_i}\right)_{i=1,\dots,M},$$

$$F_2:\mathbb{R}^{n_\mathrm{p}+n_\mathrm{s}}\longrightarrow\mathbb{R}^{n_\mathrm{s}}, \qquad F_2(p,s) := \sum_{i=1}^{N^\mathrm{pe}} c_i^\mathrm{pe}(t_i,y(t_i;p,s),z(t_i;p,s),p,u_0,s),$$

the resulting finite dimensional problem can be written as

$$\min_{p,s} \quad \tfrac{1}{2}\left\|F_1(p,s)\right\|_2^2 \tag{3.3a}$$

$$\text{s.t.} \quad F_2(p,s)=0. \tag{3.3b}$$

We denote the Jacobians of F_1 and F_2 with respect to p and s by

$$J := \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}, \tag{3.4}$$

$$J_1 := \begin{bmatrix} J_{1p}, & J_{1s} \end{bmatrix} := \begin{bmatrix} \nabla_p F_1^T, & \nabla_s F_1^T \end{bmatrix} \in \mathbb{R}^{M\times(n_\mathrm{p}+n_\mathrm{s})}, \tag{3.5}$$

$$J_2 := \begin{bmatrix} J_{2p}, & J_{2s} \end{bmatrix} := \begin{bmatrix} \nabla_p F_2^T, & \nabla_s F_2^T \end{bmatrix} \in \mathbb{R}^{n_\mathrm{s}\times(n_\mathrm{p}+n_\mathrm{s})}. \tag{3.6}$$

The parameterization must be chosen such that the following regularity assumptions hold where F_1, F_2, J_1, and J_2 are evaluated:

$$\text{rank } J_2(p,s) = n_\mathrm{s}, \tag{CQ}$$
$$\text{rank } J(p,s) = n_\mathrm{p} + n_\mathrm{s}. \tag{PD}$$

Then problem 3.3 can be solved efficiently by the generalized Gauss–Newton method [29], outlined in Algorithm 5 in Sec. 1.3.3.

Remark 3.1. Assumptions (CQ) and (PD) are special cases of LICQ (cf. Def. 1.4) and the second order sufficient condition (cf. Thm. 1.8) applied to the linear least squares problem (1.19) in Algorithm 5. To see this, reformulate (1.19) as a standard QP and use that condition (PD) implies that the Hessian of this QP, $J_1^T J_1$, is positive definite on the null space of J_2.

3.2. Sensitivity analysis of the estimates

OED aims at improving statistical uncertainty of the parameter estimates. The uncertainty is described by the covariance matrix. We show how an approximation of the covariance matrix is obtained by sensitivity analysis of the solution of the parameter estimation problem and how it can be used to compute confidence intervals. Furthermore, we show that the covariance matrix is independent of the parameterization of the infinite-dimensional parameter estimation problem.

3.2.1. Approximation of the covariance matrix

The solution of the finite dimensional parameter estimation problem (3.3) is a maximum likelihood estimator for p and s, which we denote by $\hat{v} := (\hat{p}^T, \hat{s}^T)^T \in \mathbb{R}^{n_\mathrm{v}}$. It is a random variable because the measurements η_i are random. To analyze its statistical properties, we consider the linearized least squares problem at the solution:

$$\min_{\Delta v} \quad \tfrac{1}{2} \left\| F_1(\hat{v}) + J_1(\hat{v})\Delta v \right\|_2^2 \tag{3.7a}$$
$$\text{s.t.} \quad F_2(\hat{v}) + J_2(\hat{v})\Delta v = 0. \tag{3.7b}$$

The solution operator of (3.7) is the generalized inverse J^+ and is defined as follows [31].

Definition 3.2 (Generalized Inverse)**.** The matrix

$$J^+(v) := \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} J_1(v)^T J_1(v) & J_2(v)^T \\ J_2(v) & 0 \end{bmatrix}^{-1} \begin{bmatrix} J_1(v)^T & 0 \\ 0 & \mathbb{I} \end{bmatrix}$$

is called the *generalized inverse* of J.

The existence of J^+ is ensured by conditions (CQ) and (PD). The solution $\Delta\hat{v}$ of (3.7) can now be formally written as

$$\Delta\hat{v} = -J^+(\hat{v}) \begin{bmatrix} F_1(\hat{v}) \\ F_2(\hat{v}) \end{bmatrix}.$$

Using this representation of the solution, we can compute an approximation of the variance-covariance matrix, which we will later call, for the sake of brevity, a *covariance matrix*, by

$$\begin{aligned} C &:= \mathbb{E}(\Delta\hat{v}\Delta\hat{v}^T) \\ &= \mathbb{E}(J^+ \begin{bmatrix} F_1F_1^T & F_1F_2^T \\ F_2F_1^T & F_2F_2^T \end{bmatrix} J^{+T}) \\ &= J^+ \begin{bmatrix} \mathbb{E}(F_1F_1^T) & \mathbb{E}(F_1F_2^T) \\ \mathbb{E}(F_2F_1^T) & \mathbb{E}(F_2F_2^T) \end{bmatrix} \\ &= J^+ \begin{bmatrix} \mathbb{I} & 0 \\ 0 & 0 \end{bmatrix} J^{+T} \\ &= \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} J_1^TJ_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} J_1^TJ_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} J_1^TJ_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-T} \begin{bmatrix} \mathbb{I} \\ 0 \end{bmatrix}, \end{aligned} \tag{3.8}$$

where we have used that

$$\mathbb{E}(F_1F_1^T) = \mathbb{E}(\Sigma^{-1}\varepsilon\varepsilon^T\Sigma^{-1}) = \Sigma^{-1}\Sigma^2\Sigma^{-1} = \mathbb{I}$$

with $\Sigma := \mathrm{diag}(\sigma_i)$, $i = 1,\dots,M$ and $\varepsilon = (\varepsilon_i, i = 1,\dots,M)$. We also used that $\mathbb{E}(F_1F_2^T) = \mathbb{E}(F_2F_1^T) = \mathbb{E}(F_2F_2^T) = 0$ because F_2 is not random and zero.

A more compact representation of C can be derived as follows. Define

$$\begin{bmatrix} X & Y^T \\ Y & T \end{bmatrix} := \begin{bmatrix} J_1^TJ_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-1},$$

with $X \in \mathbb{R}^{n_v \times n_v}$, $Y \in \mathbb{R}^{n_s \times n_v}$, $T \in \mathbb{R}^{n_s \times n_s}$.

Lemma 3.3. *[33] The covariance matrix C* (3.8) *is equal to the matrix X.*

Proof. According to (3.8), we have

$$\begin{aligned} C &= \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} X & Y^T \\ Y & T \end{bmatrix} \begin{bmatrix} J_1^TJ_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X & Y \\ Y^T & T \end{bmatrix} \begin{bmatrix} \mathbb{I} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} X & Y^T \end{bmatrix} \begin{bmatrix} J_1^TJ_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = XJ_1^TJ_1X. \end{aligned} \tag{3.9}$$

By definition, the blocks X and Y satisfy the linear system

$$\begin{aligned} J_1^TJ_1X + J_2^TY &= \mathbb{I}, \\ J_2X &= 0. \end{aligned}$$

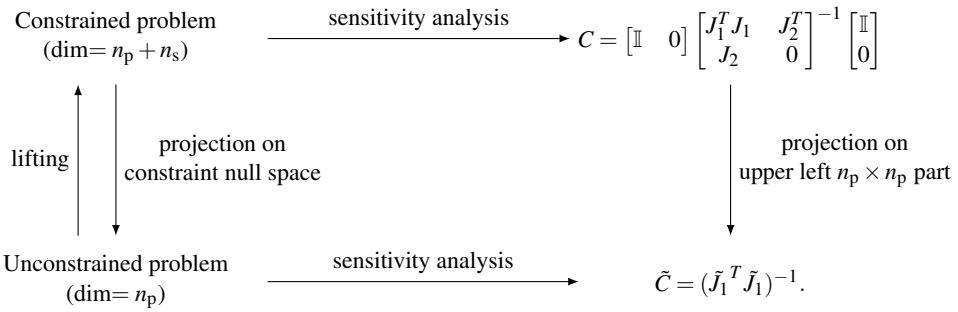
Together with relation (3.9) we obtain

$$C = X(\mathbb{I} - J_2^T Y) = X.$$

□

3.2.2. The relation between constrained and unconstrained problems

Let us now analyze a covariance matrix C for a given parameterization of the parameter estimation problem (3.2) by variables p and s. As the parameterization, i.e., the choice of variables s, is arbitrary and usually motivated by practical considerations, we want that the part of C corresponding to p is not affected by the choice of s. And indeed, the relation between covariance matrices for constrained and unconstrained parameter estimation problems is illustrated by the following diagram that is proven in Theorem 3.4:



Theorem 3.4. *Consider a parameterization of Problem* (3.2) *by variables p and s that yields the constrained least squares problem* (3.3)*. The covariance matrix for p and s is given by*

$$\begin{aligned} C &= \begin{bmatrix}\mathbb{I} & 0\end{bmatrix} \begin{bmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-1} \begin{bmatrix}\mathbb{I} \\ 0\end{bmatrix} \\ &= \begin{bmatrix}\mathbb{I} & 0 & 0 \\ 0 & \mathbb{I} & 0\end{bmatrix} \begin{bmatrix} J_{1p}^T J_{1p} & J_{1p}^T J_{1s} & J_{2p}^T \\ J_{1s}^T J_{1p} & J_{1s}^T J_{1s} & J_{2s}^T \\ J_{2p} & J_{2s} & 0 \end{bmatrix}^{-1} \begin{bmatrix}\mathbb{I} & 0 \\ 0 & \mathbb{I} \\ 0 & 0\end{bmatrix}. \end{aligned} \tag{3.10}$$

Furthermore, consider a minimum parameterization of Problem (3.2) *by variables* $p \in \mathbb{R}^{n_p}$ *that yields the unconstrained least squares problem*

$$\min_p \quad \tfrac{1}{2} \left\| \tilde{F}_1(p) \right\|_2^2, \tag{3.11}$$

and covariance matrix

$$\tilde{C} = (\tilde{J}_1{}^T \tilde{J}_1)^{-1} \in \mathbb{R}^{n_p \times n_p}.$$

Then

$$\Pi_{n_p}(C) = \tilde{C},$$

where $\Pi_{n_p}(C)$ *is the projection onto the* $n_p \times n_p$ *leading principal submatrix of* C*.*

Proof. We compare the two linear least squares problems from which C and $\tilde{C}$ are derived. First, the covariance matrix $\tilde{C}$ is derived by linearizing Problem (3.11) at a point solution $\hat{p}$:

$$\min_{\Delta p} \quad \tfrac{1}{2} \left\| \tilde{F}_1(\hat{p}) + \tilde{J}_1(\hat{p})\Delta p \right\|_2^2. \tag{3.12}$$

On the other hand, consider the linear least squares problem obtained by linearizing Problem (3.3) at the solution $\hat{p}$, $\hat{s}$:

$$\min_{\Delta p, \Delta s} \quad \tfrac{1}{2} \left\| F_1(\hat{p},\hat{s}) + J_{1p}(\hat{p},\hat{s})\Delta p + J_{1s}(\hat{p},\hat{s})\Delta s \right\|_2^2 \tag{3.13a}$$

$$\text{s.t.} \qquad F_2(\hat{p},\hat{s}) + J_{2p}(\hat{p},\hat{s})\Delta p + J_{2s}(\hat{p},\hat{s})\Delta s = 0. \tag{3.13b}$$

Because we assume that $\operatorname{rank} J_2 = n_\mathrm{s}$ we may assume that the partition of the variable vector in p and s is such that J_{2s} is nonsingular. Then for every feasible point, Δs in Prob. (3.13) is given by

$$\Delta s = -J_{2s}^{-1} J_{2p} \Delta p$$

because $F_2(\hat{p},\hat{s}) = 0$. Hence, an equivalent unconstrained problem for the given parameterization is

$$\min_{\Delta p} \quad \tfrac{1}{2} \left\| F_1 + (J_{1p} - J_{1s}J_{2s}^{-1}J_{2p})\Delta p \right\|_2^2. \tag{3.14}$$

The parameterization does not change the solution of the original problem. This means that $F_1(\hat{p},\hat{s}) = \tilde{F}_1(\hat{p})$ and Problems (3.13) and (3.12) must yield the same solution Δp, i.e.,

$$(J_{1p} - J_{1s}J_{2s}^{-1}J_{2p})^{+} \tilde{F}_1(\hat{p}) = \tilde{J}_1{}^{+} \tilde{F}_1(\hat{p}). \tag{3.15}$$

Here, the generalized inverse $(\cdot)^+$ reduces to the Moore–Penrose pseudoinverse. (3.15) holds for every realization of the measurement data. A perturbation of the measurements yields a different F_1 but leaves F_2 and the matrices $J_{(\cdot)}$ unchanged as they are independent of the measurements. In particular, (3.15) holds for M different realizations of the measurement data such that the resulting $\tilde{F}_1(\hat{p}) \in \mathbb{R}^M$ span the whole space. Together with uniqueness of the Moore–Penrose pseudoinverse, (3.15) implies that $\tilde{J}_1$ can be expressed in terms of the given parameterization as

$$\tilde{J}_1 = J_{1p} - J_{1s}J_{2s}^{-1}J_{2p}.$$

We now consider the covariance matrix of the constrained problem (3.13). To simplify expression (3.10), we choose the following basis Z for the null space of the constraint Jacobian J_2:

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \in \mathbb{R}^{(n_p+n_s)\times n_p}, \quad Z_1 = \mathbb{I}, \quad Z_2 = -J_{2s}^{-1}J_{2p}. \tag{3.16}$$

Then C can be computed as (see, e.g., [19]):

$$\begin{aligned} C &= Z\left(Z^T \begin{bmatrix} J_{1p}^T J_{1p} & J_{1p}^T J_{1s} \\ J_{1s}^T J_{1p} & J_{1s}^T J_{1s} \end{bmatrix} Z\right)^{-1} Z^T \\ &= Z\left(\left(\begin{bmatrix} \mathbb{I} & Z_2^T \end{bmatrix} \begin{bmatrix} J_{1p}^T \\ J_{1s}^T \end{bmatrix}\right)\left(\begin{bmatrix} J_{1p} & J_{1s} \end{bmatrix} \begin{bmatrix} \mathbb{I} \\ Z_2 \end{bmatrix}\right)\right)^{-1} Z^T \\ &= Z\left(\left[J_{1p} + J_{1s}Z_2\right]^T \left[J_{1p} + J_{1s}Z_2\right]\right)^{-1} Z^T. \end{aligned} \tag{3.17}$$

We note that $J_{1p} + J_{1s}Z_2 = \tilde{J}_1$. Substituting in Equation (3.17), we obtain

$$\begin{aligned} C &= \begin{bmatrix} \mathbb{I} \\ Z_2 \end{bmatrix} \left(\tilde{J}_1{}^T \tilde{J}_1\right)^{-1} \begin{bmatrix} \mathbb{I} & Z_2^T \end{bmatrix} \\ &= \begin{bmatrix} \left(\tilde{J}_1{}^T \tilde{J}_1\right)^{-1} & \left(\tilde{J}_1{}^T \tilde{J}_1\right)^{-1} Z_2^T \\ Z_2\left(\tilde{J}_1{}^T \tilde{J}_1\right)^{-1} & Z_2\left(\tilde{J}_1{}^T \tilde{J}_1\right)^{-1} Z_2^T \end{bmatrix}, \end{aligned} \tag{3.18}$$

and see that the upper $n_\mathrm{p} \times n_\mathrm{p}$ part of C is equal to $\tilde{C}$, the covariance matrix of the unconstrained problem (3.12). □

Note that Eq. (3.18) also reveals that $\operatorname{rank} C = n_\mathrm{p}$, which is consistent with our intuition that n_p parameters are given by measurement data, while the remaining n_s variables are completely defined by the constraints and the parameters p.

3.2.3. Confidence intervals

The diagonal elements of the covariance matrix play an important role in the statistical assessment and can reveal a badly-identified model. In [31] it is shown that an approximation to the linearized confidence region is given by

$$G_L(\alpha) = \left\{ \hat{v} + \Delta v \,\middle|\, \Delta v = -J^+(\hat{v}) \begin{bmatrix} \varepsilon \\ 0 \end{bmatrix}, \quad \|\varepsilon\|_2^2 \leq \gamma^2(\alpha) \right\},$$

where $\hat{v}$ is a solution of problem (3.3) and $\gamma(\alpha)$ is the quantile of the χ^2 distribution for value α with n_p degrees of freedom. It can be shown that the linearized confidence region $G_L(\alpha)$ is contained exactly in a minimal box, which is the cross product of so-called confidence intervals.

Lemma 3.5. *Let $\hat{v}$ be a solution of problem* (3.3)*. Then*

$$G_L(\alpha) \subset [\hat{v}_1 - \theta_1, \hat{v}_1 + \theta_1] \times \cdots \times [\hat{v}_{n_\mathrm{v}} - \theta_{n_\mathrm{v}}, \hat{v}_{n_\mathrm{v}} + \theta_{n_\mathrm{v}}],$$

where $\theta_i = \sqrt{C_{ii}}\gamma(\alpha)$ and C_{ii} are the diagonal elements of C.

Proof. See [31]. □

Thus the diagonal elements of C provide estimates for the confidence intervals of the parameters, that is, the intervals where the true parameter values lie within probability α. For a more detailed statistical analysis of the parameters, techniques such as likelihood profiles [168] or computation of nonlinear confidence regions [152] can be employed. Another technique to capture nonlinear effects in parameter estimation is the so-called *unscented transformation* or σ*-point method* [127] that has also been applied to OED [176].

3.3. The optimum experimental design problem

The idea of nonlinear optimum experimental design is to use the covariance matrix obtained at the solution of a parameter estimation problem to predict a covariance matrix for different experimental conditions $u(\cdot)$ and in this way try to find the experimental setup that yields a covariance matrix that is "best" in some statistical sense.

Recall the definition of the covariance matrix as given in Eq. (3.10)

$$C = \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{I} \\ 0 \end{bmatrix},$$

with the Jacobians J_1 and J_2 defined as:

$$J_1 = \begin{bmatrix} J_{1p}, & J_{1s} \end{bmatrix}, \qquad J_2 = \begin{bmatrix} J_{2p}, & J_{2s} \end{bmatrix}, \tag{3.19}$$

$$J_{1p,i} = \frac{1}{\sigma_i} \left(\frac{\partial h_i}{\partial x}(t_i, x(t_i; p, u_0, s), p, u_0) \cdot x_{\mathrm{p}}(t_i) + \frac{\partial h_i}{\partial p}(t_i, x(t_i; p, u_0, s), p, u_0) \right), \tag{3.20}$$

$$J_{1s,i} = \frac{1}{\sigma_i} \left(\frac{\partial h_i}{\partial x}(t_i, x(t_i; p, u_0, s), p, u_0) \cdot x_{\mathrm{s}}(t_i) \right), \tag{3.21}$$

$$J_{2p} = \sum_{i=1}^{N^{\mathrm{pe}}} \frac{\partial c_i^{\mathrm{pe}}}{\partial x}(t_i, x(t_i; p, u_0, s), p, u_0, s) \cdot x_{\mathrm{p}}(t_i) + \frac{\partial c_i^{\mathrm{pe}}}{\partial p}(t_i, x(t_i; p, u_0, s), p, u_0, s), \tag{3.22}$$

$$J_{2s} = \sum_{i=1}^{N^{\mathrm{pe}}} \frac{\partial c_i^{\mathrm{pe}}}{\partial x}(t_i, x(t_i; p, u_0, s), p, u_0, s) \cdot x_{\mathrm{s}}(t_i) + \frac{\partial c_i^{\mathrm{pe}}}{\partial s}(t_i, x(t_i; p, u_0, s), p, u_0, s), \tag{3.23}$$

where $J_{1,i} = \begin{bmatrix} J_{1p,i}, & J_{1s,i} \end{bmatrix}$, denotes the i-th row of J_1. The sensitivities of the states x with respect to p and s are subject to the following variational differential-algebraic equations (VDAE) (cf. Eqs. (2.14) and (2.15)):

$$\dot{y_{\mathrm{v}}}(t) = \frac{\partial f}{\partial x}(t, y(t), z(t), p, u(t)) \cdot x_{\mathrm{v}}(t) + \frac{\partial f}{\partial v}(t, y(t), z(t), p, u(t)) \tag{3.24}$$

$$0 = \frac{\partial g}{\partial x}(t, y(t), z(t), p, u(t)) \cdot x_{\mathrm{v}}(t) + \frac{\partial g}{\partial v}(t, y(t), z(t), p, u(t)), \tag{3.25}$$

where we use the notation

$$x_{\mathrm{v}}(t) = \begin{bmatrix} y_{\mathrm{v}}(t) \\ z_{\mathrm{v}}(t) \end{bmatrix} = \begin{bmatrix} y_{\mathrm{p}}(t) & y_{\mathrm{s}}(t) \\ z_{\mathrm{p}}(t) & z_{\mathrm{s}}(t) \end{bmatrix} = \begin{bmatrix} \frac{\mathrm{d}y}{\mathrm{d}p}(t) & \frac{\mathrm{d}y}{\mathrm{d}s}(t) \\ \frac{\mathrm{d}z}{\mathrm{d}p}(t) & \frac{\mathrm{d}z}{\mathrm{d}s}(t) \end{bmatrix} \qquad \text{and} \qquad v = \begin{bmatrix} p \\ s \end{bmatrix}.$$

Initial values for the VDAE are given by

$$y_{\mathrm{v}}(t_0) = \frac{\mathrm{d}y(t_0;v)}{\mathrm{d}v}$$

for the variational differential states and by (3.25) for the variational algebraic states. The terms (3.20)–(3.23) reveal the following important properties of C:

1. C does not depend explicitly on the data η
2. C depends on the experimental conditions $u(\cdot)$ and measurement times t_i
3. C depends on p and s

Because of observations 1 and 2, we may view C as a function $C(u,t_1,\ldots,t_M)$ for a fixed estimate $\hat{p}$ and *predict* a covariance matrix for different experimental setups. We use this to formulate the OED problem which is explained in the following. Item 3 will be addressed below and in Section 3.4.1.

3.3.1. Objective function

The statistical quality of the matrix C is measured by a performance criterion Φ from statistical experimental design [10, 167]. Popular criteria are:

- A-criterion: $\Phi_A = \frac{1}{n_{\mathrm{v}}} \operatorname{tr} C$,
- D-criterion: $\Phi_D = (\det(P^T C P))^{1/n_{\mathrm{v}}}$,
- E-criterion: $\Phi_E = \max\{\lambda_i \mid \lambda_i \text{ eigenvalue of } C\} = \|C\|_2$,
- M-criterion: $\Phi_M = \max\{C_{ii},\ i \in \{1,2,\ldots,n_{\mathrm{v}}\}\}$,

where $P \in \mathbb{R}^{n_{\mathrm{v}} \times n_{\mathrm{p}}}$ is a projection matrix onto an n_{p}-dimensional subspace, as the n_{v}-dimensional matrix C only has rank n_{p}, see Section 3.2. For unconstrained parameter estimation problems, these criteria have geometrical interpretations for the confidence ellipsoid defined by the matrix C as can be seen in Figure 3.1.

Criteria based on Fisher Information
Other criteria operate on the Fisher information matrix instead of the covariance matrix, which is defined here for the unconstrained case, see, e.g., [167].

Definition 3.6 (Fisher information matrix)**.** The matrix $H \in \mathbb{R}^{n_{\mathrm{p}} \times n_{\mathrm{p}}}$ defined by

$$H = J_1^T J_1 \in \mathbb{R}^{n_{\mathrm{p}} \times n_{\mathrm{p}}}$$

is called the *(discrete) Fisher information matrix*.

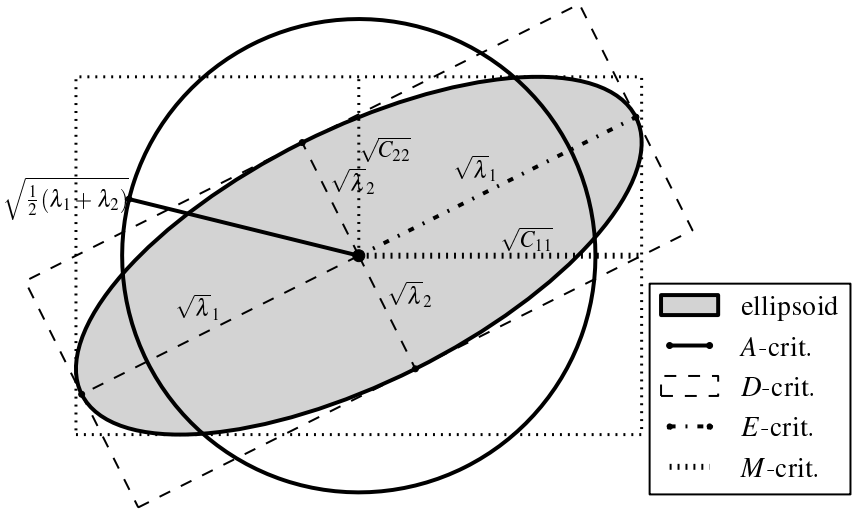


Figure 3.1.: 2-dimensional confidence ellipsoid and geometrical interpretations of the A-, D-, E-, and M-criteria (taken from [200] in modified form).

This is the inverse of the covariance matrix and thus the objective needs to be maximized in the OED problem. For example, the T-criterion is defined as

$$\Phi_T = \frac{1}{n_v} \operatorname{tr} H.$$

While it is appealing to use it in numerical methods as it does not involve a matrix inversion, its practical use is limited. A T-optimal design does not guarantee invertibility of H which means some parameters may not be identifiable by the experiment at all. Instead the T-criterion tends to favor the best identifiable parameter. An interesting alternative which mitigates this effect while still avoiding the matrix inverse has been recently proposed in [120]. Instead of maximizing the sum of the diagonal elements of H, the following function is minimized:

$$\Phi_{\text{exp}} = \frac{1}{n_v} \sum_{i=1}^{n_v} \exp(-H_{ii}). \tag{3.26}$$

Scaling

The covariance matrix and the Fisher matrix are *not* invariant with respect to the absolute values of the parameters. Thus it is usually advisable to scale p and s before optimizing the covariance matrix. A reasonable choice is to scale everything to 1, corresponding to an equal weighting of all uncertainties. The covariance matrix with respect to rescaled variables can then be derived as follows.

Lemma 3.7. *We set*

$$v = S\bar{v},$$

with, e.g., $\bar{v}_i = 1$, $i = 1,\dots,n_\mathrm{v}$ *and a scaling matrix* $S := \mathrm{diag}(v_i)$. *Then the covariance matrix* $\bar{C}$ *for the scaled quantities* $\bar{v}$ *is*

$$\bar{C} = \begin{bmatrix} S^{-1} & 0 \end{bmatrix} \begin{bmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} S^{-1} \\ 0 \end{bmatrix}, \tag{3.27}$$

that means, the i-th row and column of C are multiplied by $1/v_i$.

Proof. We apply the chain rule to obtain:

$$\begin{aligned} \bar{C} &= \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} S J_1^T J_1 S & S J_2^T \\ J_2 S & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{I} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \left(\begin{bmatrix} S & 0 \\ 0 & \mathbb{I} \end{bmatrix} \begin{bmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & \mathbb{I} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbb{I} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} S^{-1} & 0 \end{bmatrix} \begin{bmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} S^{-1} \\ 0 \end{bmatrix}. \end{aligned}$$

□

Recall that $v = (p^T, s^T)^T$ and s is allowed to vary during optimization. This implies that the part of $S = \mathrm{diag}(v_i)$ corresponding to s also changes during optimization. This must be taken into account when differentiating the scaled covariance matrix $\bar{C}$ with respect to s. Moreover, it must be guaranteed that $s_i \neq 0$ by introducing suitable bounds.

3.3.2. Measurements and constraints

Standard deviation of measurement errors

When we compute J_1, the standard deviations of the measurement errors σ_i are derived from the measurement data. However, we want to account for the fact that changing the experimental conditions results in different measurement accuracies. In many cases the measurement accuracy can be assumed as a function of the model response that is given by the measurement procedure at hand. We will denote this predicted or estimated measurement accuracy by functions $\sigma_i(t_i) = \sigma_i(t_i, x(t_i), u_0)$, $i = 1,\dots,M$.

Measurement design

Part of an experimental design are decisions *when* to measure and *what* to measure. To find the optimum measurement times for given M measurements, we can simply add $\tau_i^{\mathrm{meas}} := t_i$ as optimization variables to the problem. Constraints such as prescribed distances between measurements, may be formulated by a function $\sum_{i=1}^{M} \psi(\tau_i^{\mathrm{meas}}) \leq 0$.

The optimal selection of measurement procedures, however, makes the problem mixed-integer. Suppose there exist n_h potential measurement procedures or observables to choose from. Then we could introduce $M \cdot n_\mathrm{h}$ binary variables $\omega_{ij} \in \{0,1\}$, $i = 1,\dots,M$, $j = 1,\dots,n_\mathrm{h}$ to select a measurement procedure for time t_i and impose the additional constraint

$\sum_{j=1}^{n_\mathrm{h}} \omega_{ij} = 1$, $i = 1,\dots,M$. For now, let us assume that we only have one observable. In the next chapter, the selection of different measurement procedures can be easily included in the numerical treatment of optimal measurement time selection.

Constraints
We denote all constraints on the states and controls in the form of path constraints, cf. Eq. (2.2):

$$0 \leq c^\mathrm{d}(t, x(t), \hat{p}, u(t)) \in \mathbb{R}^{n_\mathrm{d}}, \quad t \in [t_0, t_\mathrm{f}].$$

We also need to include the boundary constraints of the model equations (3.1c). We include them in a parameterized form because the parameterization defines the covariance matrix to be optimized:

$$\sum_{i=1}^{N^\mathrm{pe}} c_i^\mathrm{pe}(t_i, y(t_i), z(t_i), \hat{p}, u_0, s^\mathrm{pe}) = 0.$$

In the following, we denote possible parameterization variables by s^pe to stress that they are optimization variables in both parameter estimation and OED problems.

3.3.3. Problem statement

The complete optimum experimental design problem is

$$\min_{\substack{x,x_\mathrm{v},J_1,J_2,\\ u,s^\mathrm{pe},\tau_i^\mathrm{meas}}} \quad \Phi\left(\begin{bmatrix}\mathbb{I} & 0\end{bmatrix}\begin{bmatrix}J_1^T J_1 & J_2^T\\ J_2 & 0\end{bmatrix}^{-1}\begin{bmatrix}\mathbb{I}\\ 0\end{bmatrix}\right) \tag{3.28a}$$

$$\text{s.t.} \quad J_{1,i} = \frac{1}{\sigma_i(\tau_i^\mathrm{meas})}\left(\frac{\partial h_i}{\partial x}x_\mathrm{v}(\tau_i^\mathrm{meas}) + \frac{\partial h_i}{\partial v}\right), \quad i = 1,\dots,M \tag{3.28b}$$

$$J_2 = \sum_{i=1}^{N^\mathrm{pe}} \frac{\partial c_i^\mathrm{pe}}{\partial x}x_\mathrm{v}(t_i) + \frac{\partial c_i^\mathrm{pe}}{\partial v}, \tag{3.28c}$$

$$\dot{y}(t) = f(t,y(t),z(t),\hat{p},u(t)), \qquad t \in [t_0,t_\mathrm{f}] \tag{3.28d}$$

$$0 = g(t,y(t),z(t),\hat{p},u(t)), \qquad t \in [t_0,t_\mathrm{f}] \tag{3.28e}$$

$$0 = \sum_{i=1}^{N^\mathrm{pe}} c_i^\mathrm{pe}(t_i,y(t_i),z(t_i),\hat{p},u_0,s^\mathrm{pe}) \tag{3.28f}$$

$$\dot{y_\mathrm{v}}(t) = \frac{\partial f}{\partial x}x_\mathrm{v}(t) + \frac{\partial f}{\partial v}, \qquad t \in [t_0,t_\mathrm{f}] \tag{3.28g}$$

$$0 = \frac{\partial g}{\partial x}x_\mathrm{v}(t) + \frac{\partial g}{\partial v}, \qquad t \in [t_0,t_\mathrm{f}] \tag{3.28h}$$

$$y_\mathrm{v}(t_0) = \frac{\partial y_0}{\partial v} \tag{3.28i}$$

$$0 \le c^\mathrm{d}(t,x(t),\hat{p},u(t)), \qquad t \in [t_0,t_\mathrm{f}] \tag{3.28j}$$

$$0 \le \sum_{i=1}^{M} \psi(\tau_i^\mathrm{meas}) \tag{3.28k}$$

where we have omitted argument lists for the partial derivatives of model and measurement functions. A functional Φ is optimized on the covariance matrix (3.28a) that is defined in terms of the Jacobians of the parameter estimation residuals (3.28b) and (3.28c). Here we have used the unscaled covariance matrix but it is also possible to use the scaled version (3.27). The Jacobians depend on the nominal DAE system (3.28d)–(3.28e) and the variational DAE system (3.28g)–(3.28h). Boundary conditions for the dynamic states are given by (3.28f) and for the variational dynamic states by (3.28i). Further constraints are process constraints (3.28j) and measurement constraints (3.28k). Apart from the states and Jacobians, the optimization variables are the controls $u(\cdot)$, the measurement times τ_i and the parameterization variables from the parameter estimation problem s^pe, whereas the current parameter estimate, $\hat{p}$, is not optimized here.

3.4. Discussion and problem variants

We conclude the chapter with a discussion of the fact that the covariance matrix depends on the parameters we want to estimate. Furthermore, we present some extensions to Prob-

lem (3.28), namely the design of multiple experiments, OED for key performance indicators and continuous measurements. We also give some prototypical examples how to choose a parameterization to set up the covariance matrix C.

3.4.1. Parameter dependency of the covariance matrix

As we have noted above, the covariance matrix C not only depends on the experimental conditions $u(\cdot)$ and the measurement times τ_i, but also on the parameter estimate $\hat{p}$, that is, the linearization point where the sensitivity analysis is done. We now present two possibilities to reduce the detrimental effect of bad parameter guesses.

A sequential approach to model validation
In practice, one should alternate the tasks of parameter estimation and optimum experimental design several times to refine the estimate for p and to arrive at a validated model. The OED approach described above can be seen as one step of an outer iteration to identify the correct parameters as depicted in Figure 3.2.

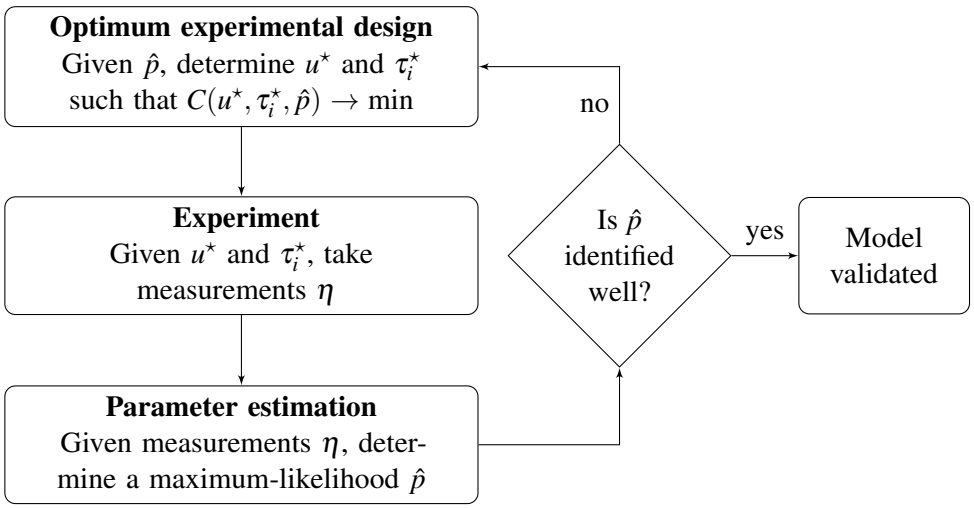


Figure 3.2.: Model validation flowchart. The first step can be either experimental design, experiment, or parameter estimation, depending on the data available. Also, adjustments to the model formulation can occur at all stages.

Robust OED
Another possibility to mitigate the effect that bad parameter values can lead to suboptimally designed experiments is to consider *robust* OED. In [135] a worst-case problem is formulated where the maximum value of $\Phi(C)$ over the confidence region of the parameters is minimized. This leads to a min-max optimization problem:

$$\min_{u,\tau_i^{\text{meas}}} \max_{\|p-\hat{p}\|_{2,\Sigma^{-1}} \leq \gamma} \Phi\Big(C(p,u,\tau_i^{\text{meas}})\Big).$$

A first-order Taylor expansion of the min-max objective function with respect to p yields:

$$\min_{u,\tau_i^{\text{meas}}} \max_{\|p-\hat{p}\|_{2,\Sigma^{-1}}\leq\gamma} \left(\Phi\Big(C(\hat{p},u,\tau_i^{\text{meas}})\Big) + \frac{\mathrm{d}}{\mathrm{d}p}\Phi\Big(C(\hat{p},u,\tau_i^{\text{meas}})\Big)(p-\hat{p}) \right)$$

The linear maximization problem can be solved explicitly:

$$\min_{u,\tau_i^{\text{meas}}} \Phi\Big(C(\hat{p},u,\tau_i^{\text{meas}})\Big) + \gamma \left\| \frac{\mathrm{d}}{\mathrm{d}p}\Phi\Big(C(\hat{p},u,\tau_i^{\text{meas}})\Big) \right\|_{2,\Sigma}.$$

Note that second-order derivatives of p appear in the objective which makes the numerical treatment of robust OED problems challenging.

3.4.2. Design of multiple experiments

Problem (3.28) can be easily extended to allow for simultaneous design of multiple experiments. An experiment is characterized by the nominal and variational dynamic system, the design variables u and τ_i, the additional parameterization variables s^{pe}, as well as process, boundary, and measurement constraints. Hence, in a multi-experiment setting with N^{exp} experiments, we have N^{exp} instances of constraints (3.28d)–(3.28k) each characterizing one experiment. The Jacobians J_1 and J_2 have the following structure:

$$J_i = \begin{bmatrix} J_{ip}^1 & J_{is^1}^1 & & & \\ J_{ip}^2 & & J_{is^2}^2 & & \\ \vdots & & & \ddots & \\ J_{ip}^{N^{\text{exp}}} & & \cdots & & J_{is^{N^{\text{exp}}}}^{N^{\text{exp}}} \end{bmatrix}, \quad i \in \{1,2\},$$

with dimensions $\left(\sum_{k=1}^{N^{\text{exp}}} M^j\right) \times \left(n_{\text{p}} + \sum_{k=1}^{N^{\text{exp}}} n_{\text{s}}^k\right)$ and $\left(\sum_{k=1}^{N^{\text{exp}}} n_{\text{s}}^k\right) \times \left(n_{\text{p}} + \sum_{k=1}^{N^{\text{exp}}} n_{\text{s}}^k\right)$, respectively. Similarly, a priori information about the parameters p, e.g. by previously performed experiments, can be incorporated by augmenting J_1 with additional, fixed rows representing these information, see [134]. The design of multiple experiments can be easily incorporated within our numerical framework, see Sec. 4.4.1.

3.4.3. Optimum experimental design for key performance indicators

In practice, one is sometimes not interested in minimizing the uncertainty of the model parameters, but of a specific output of the model, a so-called *key performance indicator (KPI)*. This could be, e.g., the yield of a product in a chemical reaction after a given time. In the context of OED the concept has first been introduced in [137].

The KPI depends not only on the parameters but also on the states and controls. Thus we define the n_{s}-dimensional KPI s^{kpi} under fixed process conditions x^{proc} and u^{proc} as

$$s^{\text{kpi}} = r(t_k, x^{\text{proc}}(t_k; p, u^{\text{proc}}), p, u_0^{\text{proc}}) \in \mathbb{R}^{n_{\text{s}}}. \tag{3.29}$$

In our framework, we regard the process characterized by x^{proc} and u^{proc} as a fixed experiment without measurements and (3.29) as boundary constraint. Now additional experiments are designed to minimize the uncertainty of the KPI. For simplicity, we assume that there is one experiment to be designed without any additional variables s^{pe}.

Lemma 3.8. *The covariance matrix of the KPI is given by*

$$C^{\text{kpi}} := r_p \left(J_{1p}^T J_{1p}\right)^{-1} r_p^T \in \mathbb{R}^{n_s}.$$

Proof. We compute the multi-experiment covariance matrix for the variables s^{kpi} and p. J_1 is vacuous for the fixed experiment and J_2 is vacuous for the second experiment. The Jacobians J_1 and J_2 for the combined experiments then have the form (cf. Sec. 3.4.2):

$$J_1 = \begin{bmatrix} J_{1p} & 0 \end{bmatrix}, \qquad J_2 = \begin{bmatrix} \frac{\mathrm{d}r(t_k)}{\mathrm{d}p} & -\mathbb{I} \end{bmatrix}$$

Note that the term $r_p := \frac{\mathrm{d}r(t_k)}{\mathrm{d}p}$ is fixed here because we consider fixed process conditions. As in Eq. (3.16), a nullspace Z of J_2 is given by

$$Z = \begin{bmatrix} \mathbb{I}_{n_p}, & -r_p^T \end{bmatrix}^T \in \mathbb{R}^{(n_s+n_p)\times n_p}.$$

Using Eq. (3.18), the covariance matrix for p and s^{kpi} simplifies to

$$\begin{aligned} C &= \begin{bmatrix} \mathbb{I} & 0 & 0 \\ 0 & \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} J_{1p}^T J_{1p} & 0 & -r_p^T \\ 0 & 0 & \mathbb{I} \\ -r_p & \mathbb{I} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{I} & 0 \\ 0 & \mathbb{I} \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \left(J_{1p}^T J_{1p}\right)^{-1} & -\left(J_{1p}^T J_{1p}\right)^{-1} r_p^T \\ -r_p \left(J_{1p}^T J_{1p}\right)^{-1} & r_p \left(J_{1p}^T J_{1p}\right)^{-1} r_p^T \end{bmatrix}, \end{aligned}$$

and the lower right principal submatrix is the covariance of the KPI. □

The objective for the KPI OED problem is

$$\Phi\left(C^{\text{kpi}}\right),$$

with Φ being one of the functionals introduced in Sec. 3.3.1.

3.4.4. Continuous measurements

Current measurement technology sometimes allows measurements to be taken at very high frequencies up to a point where measurements may be assumed to be obtained in a continuous way modeled by a continuous model response $h(t)$. For a continuous measurement design we define a binary function $w(t) \in \{0,1\}$ that defines the intervals where measurements are taken. To formulate the OED problem we need to define a continuous version of the Fisher information matrix.

Definition 3.9 (Continuous Fisher information matrix)**.** The matrix $H(t) \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}$ defined by

$$H(t_\mathrm{f}) = \int_{t_0}^{t_\mathrm{f}} w(t) J_1(t)^T J_1(t) \mathrm{d}t$$

is called the *continuous Fisher information matrix*.

We can then define an OED problem for continuous measurements as follows:

$$\begin{aligned}
\min_{\substack{x, x_\mathrm{p}, J_1, H, \\ u, w}} \quad & \Phi\left(H(t_\mathrm{f})^{-1}\right) && && \text{(3.30a)} \\
\text{s.t.} \quad & J_1(t) = \frac{1}{\sigma(t)} \left(\frac{\partial h}{\partial x} x_\mathrm{p}(t) + \frac{\partial h}{\partial p}(t) \right), && t \in [t_0, t_\mathrm{f}] && \text{(3.30b)} \\
& \dot{H}(t) = w(t) \cdot J_1(t)^T J_1(t), && t \in [t_0, t_\mathrm{f}] && \text{(3.30c)} \\
& \dot{m}(t) = w(t), && t \in [t_0, t_\mathrm{f}] && \text{(3.30d)} \\
& \dot{y}(t) = f(t, y(t), z(t), \hat{p}, u(t)), && t \in [t_0, t_\mathrm{f}] && \text{(3.30e)} \\
& 0 = g(t, y(t), z(t), \hat{p}, u(t)), && t \in [t_0, t_\mathrm{f}] && \text{(3.30f)} \\
& \dot{y_\mathrm{p}}(t) = \frac{\partial f}{\partial x} x_\mathrm{p}(t) + \frac{\partial f}{\partial p}, && t \in [t_0, t_\mathrm{f}] && \text{(3.30g)} \\
& 0 = \frac{\partial g}{\partial x} x_\mathrm{p}(t) + \frac{\partial g}{\partial p}, && t \in [t_0, t_\mathrm{f}] && \text{(3.30h)} \\
& H(t_0) = 0 && && \text{(3.30i)} \\
& m(t_0) = 0 && && \text{(3.30j)} \\
& y(t_0) = y_0(p) && && \text{(3.30k)} \\
& y_\mathrm{p}(t_0) = \frac{\partial y_0}{\partial p} && && \text{(3.30l)} \\
& 0 \leq c^\mathrm{d}(t, x(t), \hat{p}, u(t)), && t \in [t_0, t_\mathrm{f}] && \text{(3.30m)} \\
& 0 \leq M - m(t_\mathrm{f}) && && \text{(3.30n)} \\
& w(t) \in \{0, 1\} && && \text{(3.30o)}
\end{aligned}$$

Note that we introduced additional ODE states for the Fisher matrix and a continuous "measurement counter" $m(t)$ given by (3.30d) and the constraint (3.30d) to model restrictions in the measurement design. We revisit problem (3.30) in Sec. 4.4.2 and show how to treat it numerically using the methods for the discrete case.

3.4.5. Example parameterizations: Initial and boundary value problems

We conclude the discussion with some practical advice on how to choose a parameterization that defines the covariance matrix C. The discussion in Section 3.2.2 suggests different approaches to obtain a covariance matrix from a constrained or an unconstrained parameter

estimation problem. Note that the parameterizations given below do not constitute the best way to numerically *solve* the parameter estimation problem but only suggest how to set up the OED problem afterwards. In a common situation, the underlying system is a system of ODEs that are given as initial or boundary value problems and the statistical quantities of interest are only the model parameters p.

Initial value problem
If the underlying system is an initial value problem with, say, fixed initial values $y(t_0) = y_0$ and no further boundary conditions, we may eliminate these boundary conditions from the problem by incorporating the initial values in an ODE solution operator. Consequently, no variables s^{pe} are needed and J_2 is vacuous. Hence the covariance matrix is $C = (J_1^T J_1)^{-1}$. The same applies if y_0 is a model parameter or a control.

Boundary value problem
If an ODE system is given in the form of a boundary value problem (BVP) with, e.g., terminal constraints of the form $r(t_\mathrm{f}, y(t_\mathrm{f}), p) = 0$, we would introduce a variable s^{pe} and set

$$\begin{aligned} y(t_0) &= s^{\mathrm{pe}}, \\ J_{2p} &= \frac{\partial r}{\partial y} y_p(t_\mathrm{f}) + \frac{\partial r}{\partial p}, \\ J_{2s} &= \frac{\partial r}{\partial y} y_s(t_\mathrm{f}), \end{aligned}$$

to facilitate numerical solution of the BVP. In particular, we are usually not interested in the statistical properties of s^{pe} and consider only the projection of C to the upper left $n_\mathrm{p} \times n_\mathrm{p}$-part in the objective of the OED problem.

Chapter 4.

Direct shooting parameterizations for optimum experimental design problems

In this chapter we show how to transform the OED problem (3.28) to a finite-dimensional problem that can be solved by nonlinear programming algorithms such as SQP methods. OED problems are special kinds of optimal control problems, but they have several characteristics that make their numerical treatment challenging. In particular, the problem itself is defined on first-order sensitivities of the dynamic system. This means that derivative-based optimization methods require at least second order derivative information. Furthermore, the objective function operates on the covariance matrix, whose formation via a matrix inversion constitutes a nonlinear coupling in time. This is contrary to the usual assumption of a Mayer-type objective as in the standard OC problem (2.4).

Some numerical methods for OED simply augment the nominal system by the variational system and treat this as a large, unstructured system [66, 177]. The problem of the coupled objective is sometimes avoided by using the trace of the Fisher information matrix as objective criterion [84], which only exhibits linear coupling in time. These approaches are often unsatisfactory in practice because on the one hand, ignoring the structure allows only the treatment of small systems due to the resulting computational load: A dynamic system of size n_{x} must be augmented by a system of size $n_{\mathrm{x}} \cdot n_{\mathrm{p}}$. On the other hand, using the trace of the information matrix instead of the covariance matrix can lead to non-identifiability of parameters.

In [15, 134] a single shooting approach is developed that use specialized DAE solvers for the efficient generation of first- and second-order derivatives of the states by means of IND. A variant of this approach that uses adjoint information to evaluate the required sensitivity is presented in [201]. A DMS formulation has been first applied in [136] and is further investigated in [124]. Finally, a collocation approach with a full discretization of the nominal and variational system is presented in [122]. In this chapter we first review the existing single shooting approach from [134] and then describe a DMS formulation. The DMS parameterization is accompanied by a detailed discussion of the derivative structure, that must be taken into account in an efficient method for the solution of OED problems. In modified form, parts of this chapter have been published in the paper [126].

4.1. General approach

We first describe the main features of a direct method for OED problems. The following can be applied for both direct single shooting and direct multiple shooting.

Controls, path constraints and boundary constraints
As described in Sec. 2.2.1 for standard OC problems, we introduce a grid τ^{c} to parameterize the controls $u(\cdot)$ by a finite-dimensional vector $q \in \mathbb{R}^{n_{\mathrm{q}}}$ and local basis functions π_j. In this chapter, we omit dependencies on the functions π_j to simplify notation. A second grid τ^{d} is introduced for the evaluation of path constraints. Furthermore, the parameterized boundary constraints (3.28f) and variables s^{pe} from the underlying parameter estimation problem enter the resulting NLP.

Measurement design
In principle, one could directly optimize the M measurement times τ_i^{meas}. However, this introduces additional nonconvexity to the resulting NLP. Also, constraints on the measurement design and modeling the optimum selection of a measurement function may be complicated. A well-established alternative is to introduce a grid of potential measurement times:

$$t_0 = \tau_0^{\mathrm{m}} \leq \tau_1^{\mathrm{m}} \leq \cdots \leq \tau_{N^{\mathrm{m}}}^{\mathrm{m}} = t_{\mathrm{f}}. \tag{4.1}$$

The choice of the grid may be guided by the process and the measurement procedures available. Note that $\tau_j = \tau_i$, $i \neq j$ is allowed to represent the selection between multiple measurement functions that are available at the same time. Then we introduce a binary vector of *measurement weights* $w = (w_0, \ldots, w_{N^{\mathrm{m}}})^T \in \{0,1\}^{N^{\mathrm{m}}}$, where w_i represents the decision if a measurement at time τ_i^{m} should be carried out or not. The corresponding Jacobian J_1 has N^{m} rows of which M are to be selected. Recall the definition of a row $J_{1,i}$:

$$J_{1,i} = \frac{1}{\sigma_i(\tau_i^{\mathrm{m}})} \left(\frac{\partial h_i}{\partial x} x_{\mathrm{v}}(\tau_i^{\mathrm{m}}; q, s^{\mathrm{pe}}) + \frac{\partial h_i}{\partial v} \right), \quad i = 1, \ldots, N^{\mathrm{m}}.$$

With $W := \mathrm{diag}(w_i)$, the complete information matrix can be written as $J_1^T W J_1$. The constraint

$$\sum_{i=0}^{N^{\mathrm{m}}} w_i \leq M \tag{4.2}$$

ensures that only M measurements are performed. Further constraints on the measurement design may also be formulated easily in terms of linear functions of w_i. We combine all constraints on the measurement design together with (4.2) into a linear function L_M with

$$L_M(w) \geq 0. \tag{4.3}$$

Instead of requiring integrality of the measurement weights we employ a continuous relaxation:

$$0 \leq w_i \leq 1, \quad i = 0, \ldots, N^{\mathrm{m}}.$$

In practice, this often yields satisfactory results, as a bang-bang structure is often observed for the measurements weights and so integrality is satisfied automatically. Analysis of sampling decisions and a regularization technique that promotes integrality is proposed in [171] and reviewed in Sec. 4.5.

4.1.1. Direct single shooting parameterization

The simplest direct method is a direct single shooting approach. Therein, the dynamic states are eliminated from the problem and numerical integrators are employed to provide a solution for given values of controls and parameters.

Dynamic System
We assume that for every q, and s^{pe} we can compute representations of the DAE *and* VDAE solutions which can be evaluated at given times t. Note that the states do not depend on the measurement weights w. We denote this representations by $x(\cdot;q,s^{\mathrm{pe}})$ and $x_{\mathrm{v}}(\cdot;q,s^{\mathrm{pe}})$, respectively. In practice, these representations can be obtained by a numerical integrator equipped with IND. In our implementation, we use the BDF code DAESOL [17].

Objective
The Jacobians J_1 and J_2 can be computed by evaluating the solution representation of the nominal and variational states at every τ_i^{m}. We denote this by

$$J_{1,i}(q,s^{\mathrm{pe}}) = \frac{1}{\sigma_i(\tau_i^{\mathrm{m}})}\left(\frac{\partial h_i}{\partial x}x_{\mathrm{v}}(\tau_i^{\mathrm{m}};q,s^{\mathrm{pe}}) + \frac{\partial h_i}{\partial v}\right), \quad i=1,\dots,N^{\mathrm{m}},$$
$$J_2(q,s^{\mathrm{pe}}) = \sum_{i=1}^{N^{\mathrm{pe}}} \frac{\partial c_i^{\mathrm{pe}}}{\partial x}x_{\mathrm{v}}(t_i;q,s^{\mathrm{pe}}) + \frac{\partial c_i^{\mathrm{pe}}}{\partial v}.$$

Note that for solution of the resulting NLP, derivative information of $J_1(\cdot)$ and $J_2(\cdot)$ and hence of the VDAE solution representation must be available as well. For example, the gradient of the objective can be evaluated efficiently by using adjoint information as described in [201].

NLP resulting from single shooting parameterization
In summary, the NLP resulting from a direct single shooting parameterization reads as

$$\min_{q,s^{\mathrm{pe}},w} \quad \Phi\left(\begin{bmatrix}\mathbb{I} & 0\end{bmatrix}\begin{bmatrix} J_1(q,s^{\mathrm{pe}})^T W J_1(q,s^{\mathrm{pe}}) & J_2(q,s^{\mathrm{pe}})^T \\ J_2(q,s^{\mathrm{pe}}) & 0\end{bmatrix}^{-1}\begin{bmatrix}\mathbb{I}\\0\end{bmatrix}\right) \tag{4.4a}$$
$$\text{s.t.} \quad 0 = \sum_{i=1}^{N^{\mathrm{pe}}} c_i^{\mathrm{pe}}(t_i, x(t_i;q,s^{\mathrm{pe}}),q,s^{\mathrm{pe}}), \tag{4.4b}$$
$$0 \le c^{\mathrm{d}}(\tau_i^{\mathrm{d}}, x(\tau_i^{\mathrm{d}};q,s^{\mathrm{pe}}),q), \qquad i=0,\dots,N^{\mathrm{d}} \tag{4.4c}$$
$$0 \le w_i \le 1, \qquad i=0,\dots,N^{\mathrm{m}} \tag{4.4d}$$
$$0 \le L_M(w). \tag{4.4e}$$

The number of variables is $n_{\mathrm{q}} + N^{\mathrm{m}} + 1$, depending on the selection of grids τ^{c} and τ^{m}. The number of nonlinear constraints is $n_{\mathrm{d}} \cdot (N^{\mathrm{d}}+1)$, depending on the grid τ^{d}. The computational

bulk when solving this problem with an SQP algorithm is usually the evaluation of derivatives of the objective (4.4a) as this involves evaluation of derivatives of J_1 and J_2 which themselves are defined in terms of first-order derivatives of the model equations.

4.2. Direct multiple shooting parameterization

We now show how to apply a DMS discretization as described in Chapter 2 to the optimum experimental design problem (3.28). In particular we need to modify the problem formulation (2.13) to cope with the coupled objective that is characteristic for OED. We now describe how to parameterize the dynamic system and decouple the objective by means of additional variables and constraints.

Dynamic System
We combine nominal and variational states into one system, writing

$$\bar{y}(t) = \begin{cases} y(t) \\ \mathrm{vec}(y_{\mathrm{v}}(t)) \end{cases} \in \mathbb{R}^{n_y + n_y \cdot n_p}$$

for the differential states and

$$\bar{z}(t) = \begin{cases} z(t) \\ \mathrm{vec}(z_{\mathrm{v}}(t)) \end{cases} \in \mathbb{R}^{n_z + n_z \cdot n_p}$$

for the algebraic states, where $\mathrm{vec}(\cdot)$ denotes the linear transformation that maps an $m \times n$ matrix to an $m \cdot n$ column vector by stacking the columns on top of each other. The corresponding DAE system reads as

$$\dot{\bar{y}}(t) = \bar{f}(t, \bar{y}, \bar{z}, p, q) = \begin{bmatrix} f(t, y(t), z(t), p, q) \\ \mathrm{vec}\left(\frac{\partial f}{\partial x}(t, y(t), z(t), p, q) x_{\mathrm{v}}(t) + \frac{\partial f}{\partial v}(t, y(t), z(t), p, q) \right) \end{bmatrix} \tag{4.5}$$

$$0 = \bar{g}(t, \bar{y}, \bar{z}, p, q) = \begin{bmatrix} g(t, y(t), z(t), p, q) \\ \mathrm{vec}\left(\frac{\partial g}{\partial x}(t, y(t), z(t), p, q) x_{\mathrm{v}}(t) + \frac{\partial g}{\partial v}(t, y(t), z(t), p, q) \right) \end{bmatrix}. \tag{4.6}$$

This large, structured system is discretized on a multiple shooting grid, introducing additional variables and continuity and consistency constraints as described in Sec. 2.2.1. We denote these variables by $\bar{s}_j^y$ for the differential and $\bar{s}_j^z$ for the algebraic states consisting of subvectors that correspond to the different parts of the VDAE as follows:

$$\bar{s}_j^y = \begin{bmatrix} s_j^y \\ s_j^{y,1} \\ \vdots \\ s_j^{y,n_{\mathrm{p}}} \\ s_j^{y,n_{\mathrm{p}}+1} \\ \vdots \\ s_j^{y,n_{\mathrm{p}}+n_{\mathrm{s}}} \end{bmatrix} \in \mathbb{R}^{n_{\mathrm{y}} + n_{\mathrm{y}} \cdot n_{\mathrm{p}} + n_{\mathrm{y}} \cdot n_{\mathrm{s}}}, \quad \bar{s}_j^z = \begin{bmatrix} s_j^z \\ s_j^{z,1} \\ \vdots \\ s_j^{z,n_{\mathrm{p}}} \\ s_j^{z,n_{\mathrm{p}}+1} \\ \vdots \\ s_j^{z,n_{\mathrm{p}}+n_{\mathrm{s}}} \end{bmatrix} \in \mathbb{R}^{n_{\mathrm{z}} + n_{\mathrm{z}} \cdot n_{\mathrm{p}} + n_{\mathrm{z}} \cdot n_{\mathrm{s}}}, \quad j = 0, \ldots, N^{\mathrm{s}}.$$

Additionally, we define $\vec{s}_j^x$ as in (2.10):

$$\vec{s}^x = \begin{bmatrix} \vec{s}_0^x \\ \vdots \\ \vec{s}_{N^s}^x \end{bmatrix} := \begin{bmatrix} \vec{s}_0^y \\ \vec{s}_0^z \\ \vdots \\ \vec{s}_{N^s}^y \\ \vec{s}_{N^s}^z \end{bmatrix}.$$

This system has a special structure that can and should be exploited in an efficient implementation. We will give details on this in Section 4.3.

Additional constraints
To maintain partial separability of the problem we also need to introduce a variable s_j^{pe} for every shooting interval with $s_0^{\mathrm{pe}} = s^{\mathrm{pe}}$ and trivial linear continuity constraints

$$s_j^{\mathrm{pe}} - s_0^{\mathrm{pe}} = 0, \quad j = 1, \ldots, N^s.$$

Together with additional linear constraints that may be necessary for the control parameterization (cf. Remark 2.3) and the linear constraints on the measurement weights L_M we subsume all linear constraints on the design variables q, w, and s^{pe} in a linear function

$$L(q, s^{\mathrm{pe}}, w) \geq 0.$$

Objective Function
An important difference between the problem of optimum experimental design (3.28) and the general OC problem (2.4) is the nonlinear coupling in time in the objective that is due to the inversion when computing the covariance matrix, as it has been noted in [136]. In particular this violates the property of partial separation of the Lagrange function that is responsible for its sparse, block-diagonal Hessian. Recall the definition of the covariance matrix:

$$C = \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} J_1^T W J_1 & J_2^T \\ J_2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{I} \\ 0 \end{bmatrix}$$

While the expression for C is nonlinearly coupled, we can rewrite $J_1^T W J_1$ and J_2 as

$$J_1^T W J_1 = \sum_{i=0}^{N^{\mathrm{m}}} w_i J_{1,i}^T J_{1,i} \tag{4.7}$$

$$J_2 = \sum_{i=0}^{N^{\mathrm{pe}}} J_{2,i}, \tag{4.8}$$

which means the matrices $J_1^T W J_1$ and J_2 only exhibit a linear coupling in time. The term $J_{1,i}^T J_{1,i}$ can be seen as the amount of information about p and s^{pe} that is contributed by the measurement at τ_i^{m}.

In a multiple shooting context, we assign the measurement points τ_i^{m} to their respective shooting intervals and plug in the representation of the solution $x(\tau_i^{\mathrm{m}}; s_j^x, \hat{q}_j, s_j^{\mathrm{pe}})$ and $x_{\mathrm{v}}(\tau_i^{\mathrm{m}}; \bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}})$ for a suitable j. Again, $\hat{q}_j$ denotes the part of the control vector q corresponding to shooting interval j, see Eq. (2.8). We write this as

$$J_1^T W J_1 = \sum_{j=0}^{N^{\mathrm{s}}} \sum_{\substack{i: \\ \tau_j^{\mathrm{s}} \leq \tau_i^{\mathrm{m}} < \tau_{j+1}^{\mathrm{s}}}} w_i J_{1,i}(\bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}})^T J_{1,i}(\bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}),$$

$$J_2 = \sum_{j=0}^{N^{\mathrm{s}}} \sum_{\substack{i: \\ \tau_j^{\mathrm{s}} \leq t_i < \tau_{j+1}^{\mathrm{s}}}} J_{2,i}(t_i, \bar{s}_j^x, \hat{q}_j),$$

where we again set by convention $\tau_{N^{\mathrm{m}}+1}^{\mathrm{s}} = \infty$ to avoid additional notation. The terms $J_{1,i}$ and $J_{2,i}$ are defined similar as in the single shooting case:

$$J_{1,i}(\bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}) = \frac{1}{\sigma_i(\tau_i^{\mathrm{m}})} \left(\frac{\partial h_i}{\partial x} x_{\mathrm{v}}(\tau_i^{\mathrm{m}}; \bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}) + \frac{\partial h_i}{\partial v}(t_i; \bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}) \right), \quad i = 1, \ldots, N^{\mathrm{m}},$$

$$J_{2,i}(\bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}) = \frac{\partial c_i^{\mathrm{pe}}}{\partial x} x_{\mathrm{v}}(t_i; \bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}) + \frac{\partial c_i^{\mathrm{pe}}}{\partial v}(t_i; \bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}).$$

We introduce additional variables $H_1 \in \mathbb{R}^{(n_{\mathrm{p}}+n_{\mathrm{s}}) \times (n_{\mathrm{p}}+n_{\mathrm{s}})}$ and $H_2 \in \mathbb{R}^{n_{\mathrm{s}} \times (n_{\mathrm{p}}+n_{\mathrm{s}})}$ for the symmetric matrix $J_1^T W J_1$ and J_2 along with linearly coupled constraints that fit into the framework of (2.13). The objective then only depends on the newly introduced variables H_1 and H_2.

NLP resulting from DMS parameterization
With the formulations introduced above, we obtain an NLP similar to the DMS-parameterized optimal control problem (2.13):

$$\min_{\substack{q,w,s^{\mathrm{pe}},\\ \bar{s}^x,H_1,H_2}} \quad \Phi\left(\begin{bmatrix}\mathbb{I} & 0\end{bmatrix}\begin{bmatrix}H_1 & H_2^T\\ H_2 & 0\end{bmatrix}^{-1}\begin{bmatrix}\mathbb{I}\\ 0\end{bmatrix}\right) \tag{4.9a}$$

$$\text{s.t.} \quad 0 = \sum_{j=0}^{N^{\mathrm{s}}} \sum_{\substack{i:\\ \tau_j^{\mathrm{s}} \le \tau_i^{\mathrm{m}} < \tau_{j+1}^{\mathrm{s}}}} w_i J_{1,i}(\bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}})^T J_{1,i}(\bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}) - H_1, \tag{4.9b}$$

$$0 = \sum_{j=0}^{N^{\mathrm{s}}} \sum_{\substack{i:\\ \tau_j^{\mathrm{s}} \le t_i < \tau_{j+1}^{\mathrm{s}}}} J_{2,i}(t_i, \bar{s}_j^x, \hat{q}_j) - H_2, \tag{4.9c}$$

$$0 = \bar{y}(\tau_0^{\mathrm{s}}; \hat{q}_0, s_0^{\mathrm{pe}}) - \bar{s}_0^y \tag{4.9d}$$

$$0 = \bar{y}(\tau_{j+1}^{\mathrm{s}}; \bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}) - \bar{s}_{j+1}^y, \qquad j = 0, \dots, N^{\mathrm{s}} - 1 \tag{4.9e}$$

$$0 = \bar{g}(\tau_j^{\mathrm{s}}, \bar{s}_j^x, \hat{q}_j, s_j^{\mathrm{pe}}), \qquad j = 0, \dots, N^{\mathrm{s}} \tag{4.9f}$$

$$0 \le c^{\mathrm{d}}(\tau_{j_i}^{\mathrm{d}}, x(\tau_{j_i}^{\mathrm{d}}; s_j^x, \hat{q}_j, s_j^{\mathrm{pe}}), \hat{q}_j), \tag{4.9g}$$

$$j = 0, \dots, N^{\mathrm{s}} \quad i = 0, \dots, N_j^{\mathrm{d}} - 1$$

$$0 = \sum_{j=0}^{N^{\mathrm{s}}} \sum_{\substack{i:\\ \tau_j^{\mathrm{s}} \le t_i < \tau_{j+1}^{\mathrm{s}}}} c_i^{\mathrm{b}}(t_i, x(t_i; s_j^x, \hat{q}_j, s_j^{\mathrm{pe}}), \hat{q}_j, s_j^{\mathrm{pe}}) \tag{4.9h}$$

$$0 \le w_i \le 1, \qquad i = 0, \dots, N^{\mathrm{m}} \tag{4.9i}$$

$$0 \le L(q, w, s^{\mathrm{pe}}). \tag{4.9j}$$

For every shooting interval, $n_{\mathrm{x}} + n_{\mathrm{x}} \cdot n_{\mathrm{v}}$ additional variables and constraints are introduced. Note that the dimensions of the constraints (4.9b) and (4.9c) and of the corresponding variables H_1 and H_2 are independent of the number of shooting intervals.

4.3. Derivatives of the structured NLPs

The runtime of an SQP method for a problem parameterized by DMS is often dominated by the evaluation of the constraint Jacobian because continuity constraints and their derivatives include possibly expensive calls to numerical integrators. Although the DMS-parameterized OED problem (4.9) could in principle be regarded as an OC problem and solved with the strategies outlined in Sections 2.3.1–2.3.3, much can be gained by analyzing in detail the structures that are due to the characteristics of OED. In particular, the two following aspects are important: First, the dynamic system consists of a nominal and a corresponding variational system, i.e. first-order derivatives are already part of the problem formulation.

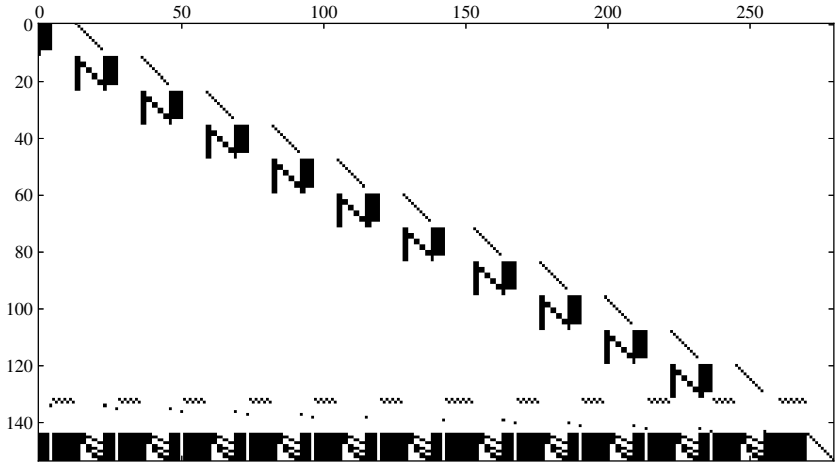


Figure 4.1.: Sparsity pattern of constraint Jacobian for a Lotka-Volterra OED problem with dimensions $n_\mathrm{y} = 2$, $n_\mathrm{p} = 4$, $n_\mathrm{d} = 2$, $n_\mathrm{q} = 50$, $N^\mathrm{m} = 100$. The shooting grid is of size $N^\mathrm{s} = 12$ which yields 280 variables and 154 constraints in the NLP. Note the sparsity within the blocks for the continuity conditions and path constraints, see the discussion around (4.10). The lowermost rows correspond to the linearly coupled constraint (4.9b).

This means whenever the constraints need to be evaluated, derivatives with respect to p and s^pe must also be provided. Ideally, this is handled by integrators that efficiently generate this sensitivity information, e.g., [4, 17, 183]. In Sec. 4.3.1, we analyze the consequences for the constraint Jacobian which can be evaluated at greatly reduced costs. The second important aspect is that due to decoupling, the objective only depends on the additionally introduced variables H_1 and H_2. This allows us to efficiently evaluate first and second-order derivatives of the objective which we will outline in Sec. 4.3.2.

We want to point out that all derivatives are to be understood as directional derivatives in the sense of the forward mode of algorithmic differentiation [112]. Derivatives for the single shooting case (4.4) are analyzed in detail in [15, 134].

4.3.1. Constraint Jacobian

We now describe the structures in the constraint Jacobian of Problem (4.9). An example of a banded Jacobian illustrating the structure induced by OED is shown in Figure 4.1 for a Lotka-Volterra model comprising two nominal states and four parameters. We first described how to efficiently compute the blocks in the banded part of the Jacobian. Then we compute the derivatives of the coupled constraints (4.9b) and (4.9c) that form the lowermost block in the Jacobian.

Derivatives of continuity and consistency constraints

The structure discussed in the following concerns *one block* in the banded part of the Jacobian. In the following we denote by

$$\left[y_{\mathrm{v}}(\tau^{\mathrm{s}}_{j+1})\right]_i$$

the i-th column of the $n_{\mathrm{y}} \times n_{\mathrm{v}}$ sensitivity matrix $y_{\mathrm{v}}(\tau^{\mathrm{s}}_{j+1})$.

Lemma 4.1 (Derivatives of continuity constraints)**.** *The part of the constraint Jacobian of Problem* (4.9) *that corresponds to the derivative of the continuity constraints* (4.9e) *evaluated at τ^{s}_{j+1} with respect to variables for shooting node j, where $j = 0,\dots,N^{\mathrm{s}}-1$, has the following structure:*

$$\frac{\mathrm{d}\bar{y}(\tau^{\mathrm{s}}_{j+1})}{\mathrm{d}\xi_j} = \begin{bmatrix} \frac{\mathrm{d}y}{\mathrm{d}s^y_j} & & & \frac{\mathrm{d}y}{\mathrm{d}s^z_j} & & & \frac{\mathrm{d}y}{\mathrm{d}\hat{q}_j} & \frac{\mathrm{d}y}{\mathrm{d}s^{\mathrm{pe}}_j} & 0 \\ \frac{\mathrm{d}[y_{\mathrm{v}}]_1}{\mathrm{d}s^y_j} & \frac{\mathrm{d}y}{\mathrm{d}s^y_j} & & \frac{\mathrm{d}[y_{\mathrm{v}}]_1}{\mathrm{d}s^z_j} & \frac{\mathrm{d}y}{\mathrm{d}s^z_j} & & \frac{\mathrm{d}[y_{\mathrm{v}}]_1}{\mathrm{d}\hat{q}_j} & \frac{\mathrm{d}[y_{\mathrm{v}}]_1}{\mathrm{d}s^{\mathrm{pe}}_j} & 0 \\ \vdots & & \ddots & \vdots & & \ddots & \vdots & \vdots & \vdots \\ \frac{\mathrm{d}[y_{\mathrm{v}}]_{n_{\mathrm{v}}}}{\mathrm{d}s^y_j} & & \frac{\mathrm{d}y}{\mathrm{d}s^y_j} & \frac{\mathrm{d}[y_{\mathrm{v}}]_{n_{\mathrm{v}}}}{\mathrm{d}s^z_j} & & \frac{\mathrm{d}y}{\mathrm{d}s^z_j} & \frac{\mathrm{d}[y_{\mathrm{v}}]_{n_{\mathrm{v}}}}{\mathrm{d}\hat{q}_j} & \frac{\mathrm{d}[y_{\mathrm{v}}]_{n_{\mathrm{v}}}}{\mathrm{d}s^{\mathrm{pe}}_j} & 0 \end{bmatrix}, \tag{4.10}$$

where we omitted the evaluation argument τ^{s}_{j+1} and the variable vector ξ_j is ordered as follows:

$$s^y_j,\quad s^{y,1}_j,\quad \cdots,\quad s^{y,n_{\mathrm{v}}}_j,\quad s^z_j,\quad s^{z,1}_j,\quad \cdots,\quad s^{z,n_{\mathrm{v}}}_j,\quad \hat{q}_j,\quad s^{\mathrm{pe}}_j \quad \hat{w}_j.$$

Proof. The zero entries in the first row of the matrix (4.10) are due to

Observation 1. The nominal states are independent of the shooting variables for the variational states:

$$\frac{\mathrm{d}y(\tau^{\mathrm{s}}_{j+1})}{\mathrm{d}s^{x,i}_j} = 0, \quad i = 1,\dots,n_{\mathrm{v}}.$$

The sparsity in the rows of (4.10) that correspond to continuity conditions for variational states is due to

Observation 2. Variational states for different parameters v_i and v_k are independent. In particular, this means

$$\frac{\mathrm{d}\left[y_{\mathrm{v}}(\tau^{\mathrm{s}}_{j+1})\right]_i}{\mathrm{d}s^{x,k}_j} = 0, \quad i \neq k.$$

Finally, the derivatives of the continuity conditions for variational states with respect to variational shooting variables can be computed by regarding

Observation 3. By differentiating the right-hand side of the VDAEs (3.24) and (3.25), we see that the derivative of the variational state for a (scalar) variable v_i, $\left[x_\mathrm{v}(t)\right]_i$, with respect to its initial values $s^{x,i}$ satisfies the following $n_\mathrm{x} \times n_\mathrm{x}$-dimensional variational DAE system:

$$\left(\frac{\mathrm{d}\left[\dot{y_\mathrm{v}}(t)\right]_i}{\mathrm{d}s^{x,i}}\right) = \frac{\partial}{\partial\left[x_\mathrm{v}\right]_i}\left(\frac{\partial f}{\partial x}\left[x_\mathrm{v}(t)\right]_i + \frac{\partial f}{\partial v}\right)\cdot\frac{\mathrm{d}\left[x_\mathrm{v}(t)\right]_i}{\mathrm{d}s^{x,i}} = \frac{\partial f}{\partial x}\cdot\frac{\mathrm{d}\left[x_\mathrm{v}(t)\right]_i}{\mathrm{d}s_j^{x,i}}$$
$$0 = \frac{\partial}{\partial\left[x_\mathrm{v}\right]_i}\left(\frac{\partial g}{\partial x}\left[x_\mathrm{v}(t)\right]_i + \frac{\partial g}{\partial v}\right)\cdot\frac{\mathrm{d}\left[x_\mathrm{v}(t)\right]_i}{\mathrm{d}s^{x,i}} = \frac{\partial g}{\partial x}\cdot\frac{\mathrm{d}\left[x_\mathrm{v}(t)\right]_i}{\mathrm{d}s^{x,i}}.$$

This is the same DAE system that describes the sensitivity of the nominal states x with respect to their initial values s^x (cf. Eq. (2.14)) and hence we have that

$$\frac{\mathrm{d}\left[y_\mathrm{v}(\tau_{j+1}^\mathrm{s})\right]_i}{\mathrm{d}s_j^{x,i}} = \frac{\mathrm{d}y(\tau_{j+1}^\mathrm{s})}{\mathrm{d}s_j^x}, \quad i = 1,\ldots,n_\mathrm{v}. \tag{4.11}$$

Furthermore, the rightmost column of the matrix (4.10) is zero because nominal and variational states are independent of the measurement weights w_i. Combining this with observations 1–3 yields the desired structure. □

In particular, Observation 3 implies that we do not need to evaluate any additional state sensitivities when we explicitly discretize the variational equations by multiple shooting. Instead, $\frac{\mathrm{d}y(t)}{\mathrm{d}s^x}$ needs to be computed only once for every shooting interval and then can be used multiple times in the constraint Jacobian. In summary, we see that the columns of matrix (4.10) corresponding to variables $s^{x,i}$ and w can be evaluated without any additional cost! Note that the derivatives $\frac{\mathrm{d}y}{\mathrm{d}(\cdot)}$ and $\frac{\mathrm{d}[y_\mathrm{v}]_i}{\mathrm{d}(\cdot)} = \frac{\mathrm{d}^2 y}{\mathrm{d}v_i\mathrm{d}(\cdot)}$ are sensitivities of first and second order, respectively, and should be supplied by the integrator via IND as outlined in Sec. 2.3.1.

The remaining constraints in one block of the Jacobian are consistency constraints (4.9f) for the nominal and variational algebraic states and the discretized path constraints (4.9g). For the consistency constraints, the same structure as in (4.10) can be observed. To see this, we denote the derivative of g with respect to v by

$$g_\mathrm{v} := g_\mathrm{v}(\tau_j^\mathrm{s}, \bar{s}_j^x, \hat{q}_j, s_j^\mathrm{pe}) := \frac{\mathrm{d}g}{\mathrm{d}v}(\tau_j^\mathrm{s}, \bar{s}_j^x, \hat{q}_j, s_j^\mathrm{pe}) \in \mathbb{R}^{n_z \times n_\mathrm{v}},$$

and by $[g_\mathrm{v}]_i$ the i-th column of g_v.

Lemma 4.2 (Derivatives of consistency constraints)**.** *The part of the constraint Jacobian of Problem* (4.9) *that corresponds to the derivative of the consistency constraints* (4.9f) *evaluated at τ_j^s with respect to variables for shooting node j, where $j = 0,\ldots,N^\mathrm{s}-1$, has the following structure:*

$$\frac{\mathrm{d}\bar{g}(\tau_j^\mathrm{s})}{\mathrm{d}\xi_j} = \begin{bmatrix} \frac{\partial g}{\partial s_j^y} & & & \frac{\partial g}{\partial s_j^z} & & & \frac{\partial g}{\partial \hat{q}_j} & \frac{\partial g}{\partial s_j^\mathrm{pe}} & 0 \\ \frac{\partial [g_\mathrm{v}]_1}{\partial s_j^y} & \frac{\partial g}{\partial s_j^y} & & \frac{\partial [g_\mathrm{v}]_1}{\partial s_j^z} & \frac{\partial g}{\partial s_j^z} & & \frac{\partial [g_\mathrm{v}]_1}{\partial \hat{q}_j} & \frac{\partial [g_\mathrm{v}]_1}{\partial s_j^\mathrm{pe}} & 0 \\ \vdots & & \ddots & \vdots & & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial [g_\mathrm{v}]_{n_\mathrm{v}}}{\partial s_j^y} & & \frac{\partial y}{\partial s_j^y} & \frac{\partial [g_\mathrm{v}]_{n_\mathrm{v}}}{\partial s_j^z} & & \frac{\partial y}{\partial s_j^z} & \frac{\partial [g_\mathrm{v}]_{n_\mathrm{v}}}{\partial \hat{q}_j} & \frac{\partial [g_\mathrm{v}]_{n_\mathrm{v}}}{\partial s_j^\mathrm{pe}} & 0 \end{bmatrix}. \tag{4.12}$$

For the path constraints, we note that they are independent of the shooting variables for the variational states, $s^{x,i}$, $i = 1,\dots,n_\mathrm{v}$ and the measurement weights w.

Lemma 4.3 (Derivatives of discretized path constraints)**.** *The part of the constraint Jacobian of Problem* (4.9) *that corresponds to the derivative of a discretized path constraint* (4.9g) *evaluated at τ_{ji}^d with respect to variables for shooting node j, where $j = 0,\dots,N^\mathrm{s}-1$, has the following structure:*

$$\frac{\mathrm{d}c^\mathrm{d}(\tau_{ji}^\mathrm{d})}{\mathrm{d}\xi_j} = \left[\frac{\partial c^\mathrm{d}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s_j^y},\quad 0,\quad \cdots,\quad 0,\quad \frac{\partial c^\mathrm{d}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s_j^z},\quad 0,\quad \cdots,\quad 0,\quad \frac{\partial c^\mathrm{d}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}\hat{q}_j}+\frac{\partial c^\mathrm{d}}{\partial \hat{q}_j},\quad \frac{\partial c^\mathrm{d}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s_j^\mathrm{pe}},\quad 0\right],$$

where we omitted the evaluation argument τ_{ji}^d and the variable vector ξ_j is ordered as follows:

$$s_j^y,\quad s_j^{y,1},\quad \cdots,\quad s_j^{y,n_\mathrm{v}},\quad s_j^z,\quad s_j^{z,1},\quad \cdots,\quad s_j^{z,n_\mathrm{v}},\quad \hat{q}_j,\quad s_j^\mathrm{pe}\quad \hat{w}_j.$$

In summary, we can obtain a block in the banded Jacobian at greatly reduced cost compared to a naive approach where each block in the Jacobian is assumed to be dense.

Derivative of constraint for decoupling the objective

The coupled constraints (4.9b) and (4.9c) are written in the form of matrix equations within problem (4.9). For the standard form of an NLP, however, the equations need to be vectorized. To this end, we introduce the following notation for symmetric matrices.

Definition 4.4 (Half-vectorization)**.** For a symmetric matrix $H = (H_{ij}) \in \mathbb{R}^{n\times n}$ we define the *half-vectorization* as the $n(n+1)/2$ dimensional column vector obtained by vectorizing the lower triangular part of H:

$$\mathrm{vech}(H) = (H_{1,1}, H_{2,1}, \dots, H_{n,1}, H_{2,2}, \dots, H_{n,2}, \dots, H_{n-1,n-1}, H_{n,n-1}, H_{n,n}) \in \mathbb{R}^{n(n+1)/2}.$$

Using the operators vec and vech, the constraints (4.9b) and (4.9c) read as

$$0 = \sum_{j=0}^{N^\mathrm{s}} \sum_{\substack{i:\\ \tau_j^\mathrm{s}\le\tau_i^\mathrm{m}<\tau_{j+1}^\mathrm{s}}} \mathrm{vech}\left(w_i J_{1,i}(\bar{s}_j^x,\hat{q}_j,s_j^\mathrm{pe})^T J_{1,i}(\bar{s}_j^x,\hat{q}_j,s_j^\mathrm{pe})\right) - \mathrm{vech}\left(H_1\right), \tag{4.13}$$

$$0 = \sum_{j=0}^{N^\mathrm{s}} \sum_{\substack{i:\\ \tau_j^\mathrm{s}\le t_i<\tau_{j+1}^\mathrm{s}}} \mathrm{vec}\left(J_{2,i}(t_i,\bar{s}_j^x,\hat{q}_j)\right) - \mathrm{vec}\left(H_2\right), \tag{4.14}$$

We see that the derivative of (4.13) with respect to the half-vectorized variables $\mathrm{vech}(H_1)$ yields $-\mathbb{I}_{n_\mathrm{v}(n_\mathrm{v}+1)/2}$ in the constraint Jacobian and the derivative of (4.14) with respect to the vectorized variables $\mathrm{vec}(H_2)$ yields $-\mathbb{I}_{n_\mathrm{s}n_\mathrm{v}}$. This implies that both H_1 and H_2 appear only as linear terms within the constraints.

For the derivative of (4.13) and (4.14) with respect to the remaining optimization variables it suffices to analyze the derivatives of the vectors vech $\left(w_i J_{1,i}^T J_{1,i}\right)$ and vec $\left(J_{2,i}\right)$ element-wise. Omitting the arguments, one element of vech $\left(w_i J_{1,i}^T J_{1,i}\right)$ is given by

$$\frac{w_i}{\sigma_i^2} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_k} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_l}, \quad k \geq l.$$

The derivative of this term is nonzero only for variables assigned to the shooting interval j containing τ_i^{m}. Keeping that in mind, we drop the index j in the following. Applying the chain-rule, we obtain the following result.

Lemma 4.5 (Derivatives of vech$(w_i J_{1,i}^T J_{1,i})$)**.** *The derivative of one element of the* $n_{\mathrm{v}} \cdot (n_{\mathrm{v}} + 1)/2$ *vector* vech $\left(w_i J_{1,i}^T J_{1,i}\right)$ *with respect to the optimization variables of Problem* (4.9) *is given by:*

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{w_i}{\sigma_i^2} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_k} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_l}\right) = w_i \left(\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{1}{\sigma_i}\frac{\mathrm{d}h_i}{\mathrm{d}v_k}\right) \cdot \frac{1}{\sigma_i}\frac{\mathrm{d}h_i}{\mathrm{d}v_l} + \frac{1}{\sigma_i}\frac{\mathrm{d}h_i}{\mathrm{d}v_k} \cdot \frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{1}{\sigma_i}\frac{\mathrm{d}h_i}{\mathrm{d}v_l}\right)\right), \tag{4.15}$$

where $k \geq l$ *and* ξ *denotes* s^x*,* q*, or* s^{pe}*. The derivative with respect to a measurement weight is*

$$\frac{\mathrm{d}}{\mathrm{d}w_i}\left(\frac{w_i}{\sigma_i^2} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_k} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_l}\right) = \frac{1}{\sigma_i^2} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_k} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}v_l}. \tag{4.16}$$

Note in particular that the weights w_i enter the constraints only linearly, which means that the second derivative $\nabla^2_{ww}\mathcal{L}(\xi, \lambda)$ vanishes. This corresponds to a small zero diagonal block within each diagonal block of the Hessian.

In the following lemmas, we further analyze the second-order part of Eq. (4.15) for different optimization variables. For better readability, we drop the measurement index i in the following. We also note that $\partial h/\partial v = \left(\partial h/\partial p, 0\right)$, as $\partial h/\partial s^{\mathrm{pe}} = 0$ but we will stick with the notation $\partial h/\partial v$ to avoid complicated notation. The following lemma summarizes formulae for the derivatives with respect to controls, shooting variables for nominal states, and parameterization variables s^{pe}.

Lemma 4.6. *The the second-order derivatives in Eq.* (4.15) *with respect to* q *can be evaluated as*

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}q}\left(\frac{1}{\sigma}\frac{\mathrm{d}h}{\mathrm{d}v_k}\right) =& \frac{\partial}{\partial x}\left(\frac{1}{\sigma}\frac{\partial h}{\partial x}\right)\frac{\mathrm{d}x}{\mathrm{d}q}\left[x_{\mathrm{v}}\right]_k + \frac{\partial}{\partial q}\left(\frac{1}{\sigma}\frac{\partial h}{\partial x}\right)\left[x_{\mathrm{v}}\right]_k. \\ &+ \left(\frac{1}{\sigma}\frac{\partial h}{\partial x}\right)\frac{\mathrm{d}\left[x_{\mathrm{v}}\right]_k}{\mathrm{d}q} + \frac{\partial}{\partial x}\left(\frac{1}{\sigma}\frac{\partial h}{\partial v_k}\right)\frac{\mathrm{d}x}{\mathrm{d}q} + \frac{\partial}{\partial q}\left(\frac{1}{\sigma}\frac{\partial h}{\partial v_k}\right)\end{aligned} \tag{4.17}$$

With respect to $\xi \in \{s^x, s^{\mathrm{pe}}\}$*, the formula simplifies to:*

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{1}{\sigma}\frac{\mathrm{d}h}{\mathrm{d}v_k}\right) = \frac{\partial}{\partial x}\left(\frac{1}{\sigma}\frac{\partial h}{\partial x}\right)\frac{\mathrm{d}x}{\mathrm{d}\xi}\left[x_{\mathrm{v}}\right]_k + \left(\frac{1}{\sigma}\frac{\partial h}{\partial x}\right)\frac{\mathrm{d}\left[x_{\mathrm{v}}\right]_k}{\mathrm{d}\xi} + \frac{\partial}{\partial x}\left(\frac{1}{\sigma}\frac{\partial h}{\partial v_k}\right)\frac{\mathrm{d}x}{\mathrm{d}\xi} \tag{4.18}$$

Proof. We obtain Eq. (4.17) by recalling that

$$\frac{\mathrm{d}h}{\mathrm{d}v_k} = \frac{\partial h}{\partial x}\left[x_\mathrm{v}\right]_k + \frac{\partial h}{\partial v_k}$$

and repeatedly applying the chain rule. Eq. (4.18) is obtained from Eq. (4.17) by noting that the dependency on s^pe and s^x is only implicit through dependency on x, i.e. $\partial h/\partial s^x = \partial h/\partial s^\mathrm{pe} = 0$. □

Again, the derivatives $\frac{\mathrm{d}x}{\mathrm{d}q}$ and $\frac{\mathrm{d}[x_\mathrm{v}]_k}{\mathrm{d}q} = \frac{\mathrm{d}^2 x}{\mathrm{d}v_k \mathrm{d}q}$ are sensitivities of the states that need to be supplied by the integrator, see Sec. 2.3.1. Next, we consider the derivative with respect to shooting variables corresponding to the r-th variational state $s^{x,r}$, $r = 1,\ldots,n_\mathrm{v}$.

Lemma 4.7. *The second-order derivatives in Eq.* (4.15) *with respect to* $s^{x,r}$, $r = 1,\ldots,n_\mathrm{v}$ *can be evaluated as:*

$$\frac{\mathrm{d}}{\mathrm{d}s^{x,r}}\left(\frac{1}{\sigma}\frac{\mathrm{d}h_i}{\mathrm{d}v_k}\right) = w_i \cdot \delta_{rk} \cdot \frac{1}{\sigma}\frac{\partial h_i}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s^x}$$

where δ_{rk} *is the Kronecker delta defined by*

$$\delta_{ij} = \begin{cases} 0 & \text{if } r \neq k \\ 1 & \text{if } r = k \end{cases}.$$

Proof. Following Observation 2 from the proof of Lemma 4.1, we note that

$$\frac{\mathrm{d}}{\mathrm{d}s^{x,r}}\left(\frac{1}{\sigma}\frac{\mathrm{d}h_i}{\mathrm{d}v_k}\right) = 0, \quad k \neq r,$$

which accounts for some additional sparsity in the Jacobian as can be seen in Fig. 4.1 in the lowermost rows. Eq. (4.11) implies that

$$\frac{\mathrm{d}}{\mathrm{d}s^{x,r}}\left(\frac{1}{\sigma}\frac{\mathrm{d}h_i}{\mathrm{d}v_r}\right) = \frac{\mathrm{d}}{\mathrm{d}s^{x,r}}\left(\frac{1}{\sigma}\frac{\partial h_i}{\partial x}\left[x_\mathrm{v}\right]_r + \frac{1}{\sigma}\frac{\partial h_i}{\partial v_r}\right) = \frac{1}{\sigma}\frac{\partial h_i}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s^x}.$$

□

Similar considerations reveal the derivatives of the coupled constraint (4.9c) for J_2. Again, we consider the derivatives of one summand of the coupled constraint element-wise and summarize them in the following Lemma.

Lemma 4.8 (Derivatives of $\mathrm{vec}(J_{2,i})$)**.** *The derivatives of one element of the* $n_\mathrm{s} \cdot n_\mathrm{v}$ *vector* $\mathrm{vec}(J_{2,i})$ *in Eq.* (4.14)*,*

$$\frac{\mathrm{d}(c_i^\mathrm{b}(t_i))_l}{\mathrm{d}v_k} =: \frac{\mathrm{d}c_l^\mathrm{b}}{\mathrm{d}v_k} = \frac{\partial c_l^\mathrm{b}}{\partial x}\left[x_\mathrm{v}\right]_k + \frac{\partial c_l^\mathrm{b}}{\partial v_k}, \quad l = 1,\ldots,n_\mathrm{s}, \quad k = 1,\ldots,n_\mathrm{v},$$

with respect to the optimization variables of Problem (4.9) *are given by:*

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}s^x}\left(\frac{\partial c_l^{\mathrm{b}}}{\partial x}[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial v_k}\right) &= \frac{\partial^2 c_l^{\mathrm{b}}}{\partial x\partial x}\frac{\mathrm{d}x}{\mathrm{d}s^x}\cdot[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial x}\cdot\frac{\mathrm{d}[x_{\mathrm{v}}]_k}{\mathrm{d}s^x}+\frac{\partial^2 c_l^{\mathrm{b}}}{\partial x\partial v_k}\frac{\mathrm{d}x}{\mathrm{d}s^x},\\
\frac{\mathrm{d}}{\mathrm{d}s^{x,r}}\left(\frac{\partial c_l^{\mathrm{b}}}{\partial x}[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial v_k}\right) &= \delta_{rk}\cdot\frac{\partial c_l^{\mathrm{b}}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s^x},\\
\frac{\mathrm{d}}{\mathrm{d}q}\left(\frac{\partial c_l^{\mathrm{b}}}{\partial x}[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial v_k}\right) &= \frac{\partial^2 c_l^{\mathrm{b}}}{\partial x\partial x}\frac{\mathrm{d}x}{\mathrm{d}q}\cdot[x_{\mathrm{v}}]_k+\frac{\partial^2 c_l^{\mathrm{b}}}{\partial q\partial x}\cdot[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial x}\cdot\frac{\mathrm{d}[x_{\mathrm{v}}]_k}{\mathrm{d}q}\\
&\quad+\frac{\partial^2 c_l^{\mathrm{b}}}{\partial x\partial v_k}\frac{\mathrm{d}x}{\mathrm{d}q}+\frac{\partial^2 c_l^{\mathrm{b}}}{\partial q\partial v_k},\\
\frac{\mathrm{d}}{\mathrm{d}s^{\mathrm{pe}}}\left(\frac{\partial c_l^{\mathrm{b}}}{\partial x}[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial v_k}\right) &= \frac{\partial^2 c_l^{\mathrm{b}}}{\partial x\partial x}\frac{\mathrm{d}x}{\mathrm{d}s^{\mathrm{pe}}}\cdot[x_{\mathrm{v}}]_k+\frac{\partial^2 c_l^{\mathrm{b}}}{\partial s^{\mathrm{pe}}\partial x}\cdot[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial x}\cdot\frac{\mathrm{d}[x_{\mathrm{v}}]_k}{\mathrm{d}s^{\mathrm{pe}}}\\
&\quad+\frac{\partial^2 c_l^{\mathrm{b}}}{\partial x\partial v_k}\frac{\mathrm{d}x}{\mathrm{d}s^{\mathrm{pe}}}+\frac{\partial^2 c_l^{\mathrm{b}}}{\partial s^{\mathrm{pe}}\partial v_k},\\
\frac{\mathrm{d}}{\mathrm{d}w_i}\left(\frac{\partial c_l^{\mathrm{b}}}{\partial x}[x_{\mathrm{v}}]_k+\frac{\partial c_l^{\mathrm{b}}}{\partial v_k}\right) &= 0.
\end{aligned}
$$

4.3.2. Objective Derivatives

Because we decoupled the objective (4.9a) by introducing additional variables and constraints, it depends only on H_1 and H_2 (and $s_{N^{\mathrm{s}}}^{\mathrm{pe}}$ in case of the scaled version (3.27)). In particular, no dynamic states are involved in its evaluation. This allows us to derive explicit formulae for the first and second derivative of the objective. Moreover, the second derivative of the objective contains the entire second order information of the problem with respect to H_1 and H_2 because they appear only linearly in the constraints. That means the part of the Hessian of the *Lagrangian* corresponding to H_i is identical with the part of the Hessian of the *objective* corresponding to H_i:

$$\nabla^2_{HH}\mathcal{L}(\xi,\lambda)=\nabla^2_{HH}\Phi(H).$$

Thus, in an SQP method using block diagonal Hessian approximations, the exact Hessian with respect to H_1 and H_2 can be incorporated without evaluating higher order sensitivities of the states. In the following, we only discuss the case of the A-criterion. Derivatives of the other criteria can be obtained in a similar way using derivative formulae given in [134, Ch.6] and [64]. However, this involves more complicated notation and the use of techniques from algorithmic differentiation is recommended [90].

We will now derive the gradient and Hessian of the objective $\Phi=\frac{1}{n_{\mathrm{v}}}\operatorname{tr}$ using directional derivatives of matrix valued functions. Note that we omit the constant factor $1/n_{\mathrm{v}}$ throughout this section.

Definition 4.9 (Matrix-valued directional derivatives)**.** Let a differentiable map $F:\mathbb{R}^{n\times n}\mapsto$

$\mathbb{R}^{n\times n}$ be given and let $A, \Delta A \in \mathbb{R}^{n\times n}$. Then the directional derivative is denoted by

$$\left(\frac{\partial F(A)}{\partial A}\cdot \Delta A\right)_{k,l} := \sum_{i,j=1}^{n} \frac{\partial F(A)_{k,l}}{\partial A_{i,j}} \Delta A_{i,j} = \lim_{h\to 0} \frac{F(A+h\Delta A)_{k,l} - F(A)_{k,l}}{h}$$

for $1 \le k,l \le n$, hence $\frac{F(A)}{A}\cdot \Delta A \in \mathbb{R}^{n\times n}$.

Let a differentiable map $\Phi : \mathbb{R}^{n\times n} \mapsto \mathbb{R}$ be given and let $A, \Delta A \in \mathbb{R}^{n\times n}$. Then the directional derivative is denoted by

$$\frac{\partial \Phi(A)}{\partial A}\cdot \Delta A := \sum_{i,j=1}^{n} \frac{\partial \Phi(A)}{\partial A_{i,j}} \Delta A_{i,j} = \lim_{h\to 0} \frac{\Phi(A+h\Delta A) - \Phi(A)}{h},$$

hence $\frac{\Phi(A)}{A}\cdot \Delta A \in \mathbb{R}$.

Derivatives of the objective with respect to variables H_1 and H_2 can be formulated as directional derivatives with unit directions in the space of symmetric matrices. We define directions $\Delta_{ij} \in \mathbb{R}^{(n_\mathrm{v}+n_\mathrm{s})\times(n_\mathrm{v}+n_\mathrm{s})}$ by

$$(\Delta_{ij})_{k,l} = \begin{cases} 1 & \text{if } (k,l) = (i,j) \text{ or } (k,l) = (j,i) \\ 0 & \text{else} \end{cases}, \quad 1 \le k,l \le n_\mathrm{v}+n_\mathrm{s}.$$

Then derivatives with respect to H_1 correspond to directions Δ_{ij}, $1 \le j \le i \le n_\mathrm{v}$ and derivatives with respect to H_2 to directions Δ_{ij}, $n_\mathrm{v} \le i \le n_\mathrm{v}+n_\mathrm{s}$, $1 \le j \le n_\mathrm{v}$. Instead of distinguishing between H_1 and H_2, we will simply refer to elements of the matrix H defined as the Fisher information matrix augmented by the constraint part, and its inverse C^{aug}:

$$H := \begin{bmatrix} H_1 & H_2^T \\ H_2 & 0 \end{bmatrix}, \quad C^{\mathrm{aug}} := H^{-1} \in \mathbb{R}^{(n_\mathrm{v}+n_\mathrm{s})\times(n_\mathrm{v}+n_\mathrm{s})}.$$

Note that in the case of an unconstrained parameter estimation problem $n_\mathrm{s} = 0$, hence $n_\mathrm{v} = n_\mathrm{p}$ and $C^{\mathrm{aug}} = C \in \mathbb{R}^{n_\mathrm{p}\times n_\mathrm{p}}$.

We summarize some useful derivative formulae in the following Lemma.

Lemma 4.10. *Let $A, \Delta A \in \mathbb{R}^{n\times n}$ and $U \in \mathbb{R}^{n\times m}$ be matrices. Then directional derivatives in direction ΔA of some functions of A are given by:*

$$\frac{\partial \operatorname{tr}(A)}{\partial A}\Delta A = \operatorname{tr}(\Delta A), \tag{4.19}$$

$$\frac{\partial A^{-1}}{\partial A}\Delta A = -A^{-1}\Delta A A^{-1}, \tag{4.20}$$

$$\frac{\partial U^T A U}{\partial A}\Delta A = U^T \Delta A U. \tag{4.21}$$

Proof. See [134, Lemma 6.2.1 and 6.2.3]. □

Objective gradient

We are interested in the derivative of the objective with respect to the newly introduced variables H_1 and H_2. We denote $C^{\text{aug}} = (C_{ij})$ with $i, j = 1, \ldots, n := n_{\text{v}} + n_{\text{s}}$. Then we can express the gradient in terms of the elements of C^{aug} as follows.

Lemma 4.11 (Gradient of the objective)**.** *The gradient of the objective* $\Phi(C) = \operatorname{tr}(C)$ *of Problem* (4.9) *with respect to* H_1 *and* H_2 *is given by*

$$\frac{\mathrm{d}\operatorname{tr}(C)}{\mathrm{d}H_{i,j}} = \begin{cases} -2\sum_{k=1}^{n_{\text{v}}} c_{ki}c_{jk} & \text{if } i > j \\ -\sum_{k=1}^{n_{\text{v}}} c_{ki}^2 & \text{if } i = j, \end{cases}, \quad 1 \le i \le n_{\text{v}} + n_{\text{s}}, \quad 1 \le j \le n_{\text{v}}, \quad j \le i.$$

Proof. Applying the chain rule and using directional derivative calculus, the gradient is equivalent to

$$\frac{\mathrm{d}\operatorname{tr}(C)}{\mathrm{d}H_{i,j}} = \frac{\partial \operatorname{tr}(C)}{\partial C}\frac{\partial C}{\partial C^{\text{aug}}}\frac{\partial C^{\text{aug}}}{\partial H}\Delta_{ij}.$$

Using (4.20) we obtain

$$\frac{\partial \operatorname{tr}(C)}{\partial C}\frac{\partial C}{\partial C^{\text{aug}}}\frac{\partial C^{\text{aug}}}{\partial H}\Delta_{ij} = -\frac{\partial \operatorname{tr}(C)}{\partial C}\frac{\partial C}{\partial C^{\text{aug}}}C^{\text{aug}}\Delta_{ij}C^{\text{aug}}$$

Evaluating the matrix $C^{\text{aug}}\Delta_{ij}C^{\text{aug}}$ for $i > j$ yields:

$$\begin{aligned} C^{\text{aug}}\Delta_{ij}C^{\text{aug}} &= \begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix} \Delta_{ij} \begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix} \\ &= \begin{pmatrix} 0 & C_{1i} & 0 & C_{1j} & 0 \\ 0 & \vdots & 0 & \vdots & 0 \\ 0 & C_{ni} & 0 & C_{nj} & 0 \end{pmatrix} \begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix} \\ &= \begin{pmatrix} C_{1i}C_{j1} + C_{1j}C_{i1} & C_{1i}C_{j2} + C_{1j}C_{i2} & \cdots & C_{1i}C_{jn} + C_{1j}C_{in} \\ C_{2i}C_{j1} + C_{2j}C_{i1} & C_{2i}C_{j2} + C_{2j}C_{i2} & \cdots & C_{2i}C_{jn} + C_{2j}C_{in} \\ \vdots & \vdots & & \vdots \\ C_{ni}C_{j1} + C_{nj}C_{i1} & C_{ni}C_{j2} + C_{nj}C_{i2} & \cdots & C_{ni}C_{jn} + C_{nj}C_{in} \end{pmatrix} \in \mathbb{R}^{n\times n} \end{aligned} \tag{4.22}$$

For $i = j$ this directional matrix is multiplied with $\frac{1}{2}$. Differentiating the projection to the upper left $n_{\text{v}} \times n_{\text{v}}$ part gives (cf. Eq. (4.21)):

$$\begin{aligned} \frac{\partial C}{\partial C^{\text{aug}}}C^{\text{aug}}\Delta_{ij}C^{\text{aug}} &= \frac{\partial}{\partial C^{\text{aug}}}\left(\begin{bmatrix}\mathbb{I} & 0\end{bmatrix} C^{\text{aug}} \begin{bmatrix}\mathbb{I} \\ 0\end{bmatrix}\right) \cdot C^{\text{aug}}\Delta_{ij}C^{\text{aug}} = \begin{bmatrix}\mathbb{I} & 0\end{bmatrix} C^{\text{aug}}\Delta_{ij}C^{\text{aug}} \begin{bmatrix}\mathbb{I} \\ 0\end{bmatrix} \\ &= \begin{pmatrix} C_{1i}C_{j1} + C_{1j}C_{i1} & \cdots & C_{1i}C_{jn_{\text{v}}} + C_{1j}C_{in_{\text{v}}} \\ \vdots & & \vdots \\ C_{n_{\text{v}}i}C_{j1} + C_{n_{\text{v}}j}C_{i1} & \cdots & C_{n_{\text{v}}i}C_{jn_{\text{v}}} + C_{n_{\text{v}}j}C_{in_{\text{v}}} \end{pmatrix} \\ &=: D_{ij} \in \mathbb{R}^{n_{\text{v}}\times n_{\text{v}}} \end{aligned}$$

Finally, noting symmetry of D_{ij}, we obtain

$$\frac{\mathrm{d}\,\mathrm{tr}(C)}{\mathrm{d}H_{i,j}} = -\frac{\partial\,\mathrm{tr}(C)}{\partial C}\cdot D_{ij} = -\,\mathrm{tr}(D_{ij}) = \begin{cases} -2\sum_{k=1}^{n_\mathrm{v}} c_{ki}c_{jk} & \text{if } i > j \\ -\sum_{k=1}^{n_\mathrm{v}} c_{ki}^2 & \text{if } i = j. \end{cases} \tag{4.23}$$

□

Objective Hessian

We now derive explicit expressions for the objective Hessian that can be readily evaluated once C^{aug} is known.

Lemma 4.12 (Hessian of the objective)**.** *The Hessian of the objective* $\Phi(C) = \mathrm{tr}(C)$ *of Problem* (4.9) *with respect to* H_1 *and* H_2 *is given by*

$$\frac{\mathrm{d}^2\,\mathrm{tr}(C)}{\mathrm{d}H_{i,j}\mathrm{d}H_{k,l}} = \begin{cases} 2\sum_{r=1}^{n_\mathrm{v}} \left(C_{jk}C_{lr} + C_{jl}C_{kr}\right)C_{ri} + \left(C_{ik}C_{lr} + C_{il}C_{kr}\right)C_{rj} & \text{if } i > j, k > l \\ 2\sum_{r=1}^{n_\mathrm{v}} \left(C_{ik}C_{lr} + C_{il}C_{kr}\right)C_{ri} & \text{if } i = j, k > l \\ 2\sum_{r=1}^{n_\mathrm{v}} C_{ik}C_{kr}C_{ri} & \text{if } i = j, k = l \end{cases},$$
$$1 \le i,k \le n_\mathrm{v} + n_\mathrm{s}, \quad 1 \le j,l \le n_\mathrm{v}, \quad j \le i, \quad l \le k.$$

Proof. We differentiate the objective gradient (4.23) with respect to $H_{k,l}$. Again, we employ the chain rule and (4.20) to obtain

$$-\frac{\mathrm{d}\,\mathrm{tr}(D_{ij})}{\mathrm{d}H_{k,l}} = -\frac{\partial\,\mathrm{tr}(D_{ij})}{\partial D_{ij}}\frac{\partial D_{ij}}{\partial C^{\mathrm{aug}}}\frac{\partial C^{\mathrm{aug}}}{\partial H}\Delta_{kl} = \frac{\partial\,\mathrm{tr}(D_{ij})}{\partial D_{ij}}\frac{\partial D_{ij}}{\partial C^{\mathrm{aug}}}\underbrace{\left(C^{\mathrm{aug}}\Delta_{kl}C^{\mathrm{aug}}\right)}_{=:\bar{D}_{kl}}.$$

Analyzing the middle and right-hand side terms using (4.21) reveals

$$\begin{aligned}\frac{\partial D_{ij}}{\partial C^{\mathrm{aug}}}\bar{D}_{kl} &= \frac{\partial\left(\begin{bmatrix}\mathbb{I} & 0\end{bmatrix}C^{\mathrm{aug}}\Delta_{ij}C^{\mathrm{aug}}\begin{bmatrix}\mathbb{I}\\0\end{bmatrix}\right)}{\partial C^{\mathrm{aug}}\Delta_{ij}C^{\mathrm{aug}}}\left(\frac{\partial C^{\mathrm{aug}}\Delta_{ij}C^{\mathrm{aug}}}{\partial C^{\mathrm{aug}}}\cdot\bar{D}_{kl}\right)\\ &= \begin{bmatrix}\mathbb{I} & 0\end{bmatrix}\left(\frac{\partial C^{\mathrm{aug}}\Delta_{ij}C^{\mathrm{aug}}}{\partial C^{\mathrm{aug}}}\cdot\bar{D}_{kl}\right)\begin{bmatrix}\mathbb{I}\\0\end{bmatrix}\\ &= \begin{bmatrix}\mathbb{I} & 0\end{bmatrix}\left(\bar{D}_{kl}\Delta_{ij}C^{\mathrm{aug}} + C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl}\right)\begin{bmatrix}\mathbb{I}\\0\end{bmatrix}\end{aligned}$$

Note that $\bar{D}_{kl}\Delta_{ij}C^{\mathrm{aug}} = \left(C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl}\right)^T$, thus we have

$$\frac{\partial\,\mathrm{tr}(D_{ij})}{\partial D_{ij}}\left(\begin{bmatrix}\mathbb{I} & 0\end{bmatrix}\left(\bar{D}_{kl}\Delta_{ij}C^{\mathrm{aug}} + C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl}\right)\begin{bmatrix}\mathbb{I}\\0\end{bmatrix}\right) = 2\sum_{r=1}^{n_\mathrm{v}}\left[C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl}\right]_{rr}.$$

It remains to compute the diagonal elements $\left[C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl}\right]_{rr}$ where we need to distinguish between three cases:

1. $H_{i,j}$ and $H_{k,l}$ off-diagonal elements
2. $H_{i,j}$ diagonal element and $H_{k,l}$ off-diagonal element
3. $H_{i,j}$ and $H_{k,l}$ diagonal elements

Using Eq. (4.22) we recall the definition of $C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl}$ for case 1:

$$C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl} = \begin{pmatrix} 0 & C_{1i} & 0 & C_{1j} & 0 \\ 0 & \vdots & 0 & \vdots & 0 \\ 0 & C_{ni} & 0 & C_{nj} & 0 \end{pmatrix} \begin{pmatrix} C_{1k}C_{l1}+C_{1l}C_{k1} & \cdots & C_{1k}C_{ln}+C_{1l}C_{kn} \\ \vdots & & \vdots \\ C_{nk}C_{l1}+C_{nl}C_{k1} & \cdots & C_{nk}C_{ln}+C_{nl}C_{kn} \end{pmatrix},$$

which gives

$$[C^{\mathrm{aug}}\Delta_{ij}\bar{D}_{kl}]_{rr} = \left(C_{jk}C_{lr}+C_{jl}C_{kr}\right)C_{ri} + \left(C_{ik}C_{lr}+C_{il}C_{kr}\right)C_{rj}.$$

In summary, the objective Hessian is given by

$$\frac{\mathrm{d}^2\,\mathrm{tr}(C)}{\mathrm{d}H_{i,j}\mathrm{d}H_{k,l}} = \begin{cases} 2\sum_{r=1}^{n_{\mathrm{v}}}\left(C_{jk}C_{lr}+C_{jl}C_{kr}\right)C_{ri} + \left(C_{ik}C_{lr}+C_{il}C_{kr}\right)C_{rj} & \text{if } i>j, k>l \\ 2\sum_{r=1}^{n_{\mathrm{v}}}\left(C_{ik}C_{lr}+C_{il}C_{kr}\right)C_{ri} & \text{if } i=j, k>l \\ 2\sum_{r=1}^{n_{\mathrm{v}}} C_{ik}C_{kr}C_{ri} & \text{if } i=j, k=l \end{cases}. \tag{4.24}$$

□

Remark 4.13. If the objective is defined on the scaled covariance matrix (3.27) the entries C_{ij}, $1 \le j \le i \le n_{\mathrm{v}}$, are replaced by $v_i^{-1}v_j^{-1}C_{ij}$. First and second derivatives with respect to $s_{N^{\mathrm{s}}}^{\mathrm{pe}}$ are then easily obtained by differentiating the trace and the terms in Eq. (4.23) elementwise.

We conclude this section with a result for the Hessian of the A-criterion that is of particular interest for optimization algorithms.

Lemma 4.14 (Convexity of the trace)**.** *The function $\Phi(X)$ that maps a positive definite matrix X to the trace of its inverse,*

$$\Phi(X) = \mathrm{tr}(X^{-1}),$$

is convex.

Proof. We show that every restriction of Φ to an arbitrary line is a convex scalar function. Let X be a positive definite symmetric matrix and V a symmetric matrix. To show convexity of Φ, we need to show that

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\Phi(X+tV)\bigg|_{t=0} \ge 0.$$

Repeatedly using the identity [182]

$$(\mathbb{I}+tX^{-1}V)^{-1} = \mathbb{I} - tX^{-1}\left(\mathbb{I}+tVX^{-1}\right)^{-1}V \tag{4.25}$$

we can write

$$\begin{aligned}
(X+tV)^{-1} &= \left(\mathbb{I}+tX^{-1}V\right)^{-1}X^{-1} \\
&= \left(\mathbb{I} - tX^{-1}\left(\mathbb{I}+tVX^{-1}\right)^{-1}V\right)X^{-1} \\
&= \left(\mathbb{I} - tX^{-1}\left(\mathbb{I} - V\left(\mathbb{I}+tX^{-1}V\right)^{-1}tX^{-1}\right)V\right)X^{-1} \\
&= X^{-1} - tX^{-1}VX^{-1} + t^2X^{-1}V\left(\mathbb{I}+tX^{-1}V\right)^{-1}X^{-1}VX^{-1} \\
&= X^{-1} - tX^{-1}VX^{-1} + t^2X^{-1}V\left(\mathbb{I} - tX^{-1}\left(\mathbb{I}+tVX^{-1}\right)V\right)^{-1}X^{-1}VX^{-1} \\
&= X^{-1} - tX^{-1}VX^{-1} + t^2X^{-1}VX^{-1}VX^{-1} + t^3\cdots,
\end{aligned}$$

so

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\Phi(X+tV)\bigg|_{t=0} = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\operatorname{tr}((X+tV)^{-1})\bigg|_{t=0} = 2\operatorname{tr}(X^{-1}VX^{-1}VX^{-1}).$$

Defining $U = X^{-1}V$ we can write $X^{-1}VX^{-1}VX^{-1} = UX^{-1}U^T$ and because X^{-1} is positive definite, $UX^{-1}U^T$ is positive semidefinite, therefore

$$\operatorname{tr}(UX^{-1}U^T) \geq 0.$$

□

4.4. Application to related problems

Here we briefly describe how to efficiently apply the DMS parameterization framework to the multi-experiment setting and the presence of continuous measurements.

4.4.1. Multiple experiments

For the design of multiple experiments in parallel as described in Sec. 3.4.2 we propose the following strategy: We think of the experiments as executed *consecutively* and parameterize the dynamics for each experiment as in the single-experiment case. This yields N^{exp} sets of constraints (4.9d)–(4.9j) and preserves the general derivative structure as described in the previous section.

The linearly coupled constraints for the Fisher matrix (4.9b) and J_2 (4.9c) now span N^{exp} experiments and deserve a closer analysis. Let us denote by $s^{\mathrm{pe},k}$ the variables that parameterize the boundary constraints of the k-th experiment, by h_i^k the i-th measurement

of the k-th experiment and by $c_i^{\mathrm{pe},k}$ the i-th summand of the boundary conditions of the k-th experiment. We further define the parts of J_1 and J_2 corresponding to experiment k by

$$\begin{aligned}
J_{1p,i}^k &= \frac{\mathrm{d}h_i^k}{\mathrm{d}p}, \quad i = 1,\dots,N^{\mathrm{m},k} \\
J_{1s,i}^k &= \frac{\mathrm{d}h_i^k}{\mathrm{d}s^{\mathrm{pe},k}}, \quad i = 1,\dots,N^{\mathrm{m},k} \\
J_{2p}^k &= \sum_{i=1}^{N^{\mathrm{c},k}} \frac{\mathrm{d}c_i^{\mathrm{pe},k}}{\mathrm{d}p} \\
J_{2s}^k &= \sum_{i=1}^{N^{\mathrm{c},k}} \frac{\mathrm{d}c_i^{\mathrm{pe},k}}{\mathrm{d}s^{\mathrm{pe},k}}
\end{aligned}$$

While the parameters p generally yield contributions J_{1p}^k and J_{2p}^k for every experiment k, the variables $s^{\mathrm{pe},k}$ yield a nonzero contribution only for the experiment k for which they are defined. The Fisher matrix has the following arrow-structure:

$$\begin{bmatrix}
\sum_{k=1}^{N^{\mathrm{exp}}} (J_{1p}^k)^T (J_{1p}^k) & (J_{1p}^1)^T (J_{1s}^1) & (J_{1p}^2)^T (J_{1s}^2) & \cdots & (J_{1p}^{N^{\mathrm{exp}}})^T (J_{1s}^{N^{\mathrm{exp}}}) \\
(J_{1p}^1)^T (J_{1s}^1) & (J_{1s}^1)^T (J_{1s}^1) & & & \\
(J_{1p}^2)^T (J_{1s}^2) & & (J_{1s}^2)^T (J_{1s}^2) & & \\
\vdots & & & \ddots & \\
(J_{1p}^{N^{\mathrm{exp}}})^T (J_{1s}^{N^{\mathrm{exp}}}) & & & & (J_{1s}^{N^{\mathrm{exp}}})^T (J_{1s}^{N^{\mathrm{exp}}})
\end{bmatrix}.$$

The matrix J_2 reads as

$$J_2 = \begin{bmatrix}
J_{2p}^1 & J_{2s}^1 & & & \\
J_{2p}^2 & & J_{2s}^2 & & \\
\vdots & & & \ddots & \\
J_{2p}^{N^{\mathrm{exp}}} & & \cdots & & J_{2s}^{N^{\mathrm{exp}}}
\end{bmatrix}.$$

In an efficient implementation, these sparsity structures must be exploited when evaluating the linearly coupled constraints (4.9b) and (4.9c). In particular, variables H_1 and H_2 are only required for the nonzero entries, which reduces the overall dimension of the NLP.

4.4.2. Continuous measurements

When continuous measurements are present as described in Sec. 3.4.4, the Fisher matrix is given by the integral

$$H(t_{\mathrm{f}}) = \int_{t_0}^{t_{\mathrm{f}}} w(t) J_1(t)^T J_1(t) \mathrm{d}t$$

giving rise to the continuous OED problem (3.30). We approximate the integral by the rectangle method on a sufficiently fine measurement grid τ^{m}. On the same grid, we approximate the control function $0 \leq w(t) \leq 1$ by a piecewise constant function with $w(\tau_i^{\mathrm{m}}) = w_i$ which results in

$$H(t_{\mathrm{f}}) \approx \sum_{i=1}^{N^{\mathrm{m}}-1} (\tau_{i+1}^{\mathrm{m}} - \tau_i^{\mathrm{m}}) w_i J_{1,i}^T J_{1,i}.$$

The measurement counting state (3.30d), m, is approximated by

$$m(t_{\mathrm{f}}) = \int_{t_0}^{t_{\mathrm{f}}} w(t)\mathrm{d}t \approx \sum_{i=1}^{N^{\mathrm{m}}-1} (\tau_{i+1}^{\mathrm{m}} - \tau_i^{\mathrm{m}}) w_i.$$

We define new measurement weights $\tilde{w}_i$, $i = 1, \ldots, N^{\mathrm{m}} - 1$, as

$$\tilde{w}_i = (\tau_{i+1}^{\mathrm{m}} - \tau_i^{\mathrm{m}}) w_i, \qquad 0 \leq \tilde{w}_i \leq \tau_{i+1}^{\mathrm{m}} - \tau_i^{\mathrm{m}}.$$

This enables us to use the shooting methods introduced in this chapter without further modification for problems that involve continuous measurements.

4.5. Problem modifications

In this section, we briefly present several modifications of problem (4.9) that change the properties of the resulting NLP and possibly the behavior of the optimization algorithm.

4.5.1. Regularization of design variables

Badly chosen grids for discretization of the controls and potential measurements, τ^{c} and τ^{m}, may lead to ill-conditioned optimization problems. In this case, the following two regularization procedures for w and q can be helpful. Note that both regularization schemes preserve the DMS induced block diagonal structure in the Hessian.

Regularization to promote integer measurement weights
In [171] the use of an L_1 regularization term to promote sparsity in the measurement weights w_i is suggested. Thus, the objective becomes

$$\Phi(C) + \varepsilon_w \sum_{i=1}^{N^{\mathrm{m}}} w_i.$$

The penalty parameter ε_w has a practical interpretation: An optimal sampling design never performs measurements when the information gain is below the penalization parameter ε_w. The information gain is defined as $F(t_{\mathrm{f}})^{-1} F(t) F(t_{\mathrm{f}})^{-1}$, where $F(t)$ denotes the Fisher information matrix incorporating all measurements until time t, see Sec. 3.3.1. In other words, a measurement is not selected if it does not sufficiently reduce the objective although the measurement constraint would permit it.

Control regularization
Some optimal control problems have singular regions, where the optimal value of the control cannot be determined uniquely. This can lead to numerical instabilities. A remedy is to penalize excessive switching by a quadratic term in the objective. The regularized objective reads as

$$\Phi(C) + \tfrac{\varepsilon_q}{2} \left\| q - q^{\mathrm{ref}} \right\|_2^2,$$

where q^{ref} is a reference value for q and ε_q is a small regularization parameter.

4.5.2. Nonlinear transformation of the objective

In [150], Mommer et al. propose a nonlinear transformation to address the problem of slow convergence of SQP methods for OED problems. Their so-called *left preconditioner* is defined as

$$\psi : (0,\infty) \to (-\infty, 0), \quad \psi(x) = -x^{-2}.$$

This means, the objective (3.28a), $\Phi(C)$, is substituted by

$$\psi(\Phi(C)) = -\frac{1}{\Phi(C)^2}.$$

The objective derivative can easily be evaluated by using the derivative of Φ and applying the chain rule:

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\psi(\Phi(C)) = \frac{2}{\Phi(C)^3} \cdot \frac{\mathrm{d}}{\mathrm{d}\xi}\Phi(C).$$

Using this transformation, we indeed observed accelerated local convergence for a Lotka–Volterra example, see Ch. 12.

4.5.3. Objective decoupling with pseudo states

Alternative formulations of problem (4.9) have been considered in [124, 136]. To simplify notation, we write down problem (4.9) for the case of an unconstrained parameter estimation problem and an ODE system. Let us also assume that some prior information on the parameters is available at t_0 in the form of a symmetric positive definite matrix H_0. Then the problem reads as

$$\min_{q,w,\bar{s}^y,H} \quad \Phi\left(H^{-1}\right) \tag{4.26a}$$

$$\text{s.t.} \qquad 0 = H_0 + \sum_{j=0}^{N^{\mathrm{s}}} \sum_{\substack{i:\\ \tau_j^{\mathrm{s}} < \tau_i^{\mathrm{m}} \le \tau_{j+1}^{\mathrm{s}}}} w_i J_{1,i}^T J_{1,i} - H, \tag{4.26b}$$

$$0 = \bar{y}(\tau_0^{\mathrm{s}}; \hat{q}_0) - \bar{s}_0^y \tag{4.26c}$$

$$0 = \bar{y}(\tau_{j+1}^{\mathrm{s}}; \bar{s}_j^y, \hat{q}_j) - \bar{s}_{j+1}^y, \qquad j = 0, \ldots, N^{\mathrm{s}} - 1 \tag{4.26d}$$

$$+\text{further constraints as in (4.9)}$$

Two possibilities to replace the linearly coupled constraint (4.26b) are given as follows:

1. We introduce N^{s} matrix-valued variables $H_j \in \mathbb{R}^{n_{\mathrm{p}} \times n_{\mathrm{p}}}$, $j = 1, \ldots, N^{\mathrm{s}}$ and corresponding constraints to describe the *evolution of the information matrix* on the shooting intervals:

$$0 = H_j + \sum_{\substack{i: \\ \tau_j^{\mathrm{s}} < \tau_i^{\mathrm{m}} \leq \tau_{j+1}^{\mathrm{s}}}} w_i J_{1,i}^T J_{1,i} - H_{j+1}, \quad j = 0, \ldots, N^{\mathrm{s}} - 1. \tag{4.27}$$

2. At every grid point τ_j^{s} the covariance matrix taking into account all measurements up to time τ_j^{s} is the inverse of H_j. From (4.27) we can thus obtain a recursion formula to describe the *evolution of the covariance matrix*:

$$0 = \left(C_j^{-1} + \sum_{\substack{i: \\ \tau_j^{\mathrm{s}} < \tau_i^{\mathrm{m}} \leq \tau_{j+1}^{\mathrm{s}}}} w_i J_{1,i}^T J_{1,i} \right)^{-1} - C_{j+1}, \quad j = 0, \ldots, N^{\mathrm{s}} - 1, \tag{4.28}$$

   where $C_0 = H_0^{-1}$ exists by assumption.

The corresponding objectives read $\Phi\left(H_{N^{\mathrm{s}}}^{-1}\right)$ and $\Phi\left(C_{N^{\mathrm{s}}}\right)$, respectively. These formulations resemble the treatment of dynamic states in DMS, only C and H are given by difference equations rather than differential equations. This motivates the name *pseudo states* for H_j and C_j and *pseudo continuity conditions* for (4.27) and (4.28).

Note that the constraints (4.27) are purely linear in H_j and will always be satisfied in an SQP method if feasible initial values for H_j are provided. Hence, Newton's method in exact arithmetics for this formulation will deliver the same iterates as for problem (4.26).

Remark 4.15. It is interesting to note that Eq. (4.27) and (4.28) can also be derived from the continuous measurement formulation for H. We have the following ODE for H:

$$\frac{\mathrm{d}}{\mathrm{d}t} H(t) = w(t) J_1(t)^T J_1(t) \tag{4.29}$$

From this we can derive the following ODE for $C(t) = H(t)^{-1}$:

$$\frac{\mathrm{d}}{\mathrm{d}t} C(t) = \frac{\mathrm{d}}{\mathrm{d}t} H(t)^{-1} = -H^{-1} \left(\frac{\mathrm{d}}{\mathrm{d}t} H(t) \right) H^{-1} = -C(t) w(t) J_1(t)^T J_1(t) C(t). \tag{4.30}$$

If we let $w(\tau_i^{\mathrm{m}})$ in (4.29) and (4.30) be the Dirac delta function, we obtain the difference equations (4.27) and (4.28) as solutions.

Remark 4.16. Numerical experience with the two pseudo state formulations is reported in [124]. Pseudo states for the Fisher matrix are competitive with our standard formulation (4.9), however, they incur additional overhead for the linear algebra as more variables and constraints are present in the NLP. The formulation with pseudo states for the covariance matrix (4.28) has been found to be unsuitable for practical use because it gives rise to ill-conditioned linear systems.

4.5.4. Telescoping sum objective

A way to transform a Mayer-type objective in optimal control problems parameterized by DMS is to expand the objective by means of a telescoping sum. This was implemented within the multiple shooting software MUSCOD [34, 163] but has not been documented. Consider first the finite-dimensional NLP resulting from a general OC problem:

$$\begin{aligned}
\min_{q,s^y} \quad & \Phi\left(s^y_{N^\mathrm{s}}\right) && \text{(4.31a)}\\
\text{s.t.} \quad & 0 = y(\tau^\mathrm{s}_0;\hat{q}_0) - s^y_0, && \text{(4.31b)}\\
& 0 = y(\tau^\mathrm{s}_{j+1};s^y_j,\hat{q}_j) - s^y_{j+1}, \qquad j = 0,\ldots,N^\mathrm{s}-1 && \text{(4.31c)}\\
& \quad +\text{further constraints as in (2.13)}
\end{aligned}$$

At a solution, it holds in particular that $s^y_{j+1} = y(\tau^\mathrm{s}_{j+1};s^y_j,\hat{q}_j)$. Thus the objective can be equivalently formulated as

$$\Phi\left(s^y_{N^\mathrm{s}}\right) = \sum_{j=1}^{N^\mathrm{s}} \Phi\left(y(\tau^\mathrm{s}_j)\right) - \sum_{j=1}^{N^\mathrm{s}-1} \Phi(s^y_j). \tag{4.32}$$

For this formulation, the variable $s^y_{N^\mathrm{s}}$ can be omitted. Eq. (4.32) is a telescoping sum; all terms cancel out except one. The objective function value and the primal variables at the solution do not change, but the objective gradient as well as the Lagrange multipliers do, so an SQP method will in general take different intermediate steps.

Application to OED

The telescoping sum formulation (4.32) can be applied to the pseudo state formulations of OED involving pseudo continuity conditions (4.27) or (4.28). The transformed objective functions read as

$$\Phi\left(H^{-1}_{N^\mathrm{s}}\right) = \sum_{j=0}^{N^\mathrm{s}-1} \Phi\left(\left(H_j + \sum_{\substack{i:\\ \tau^\mathrm{s}_j < \tau^\mathrm{m}_i \le \tau^\mathrm{s}_{j+1}}} w_i J^T_{1,i} J_{1,i}\right)^{-1}\right) - \sum_{j=1}^{N^\mathrm{s}-1} \Phi(H^{-1}_j),$$

for the formulation with pseudo states of the form (4.27), and

$$\Phi\left(C_{N^\mathrm{s}}\right) = \sum_{j=0}^{N^\mathrm{s}-1} \Phi\left(\left(C^{-1}_j + \sum_{\substack{i:\\ \tau^\mathrm{s}_j < \tau^\mathrm{m}_i \le \tau^\mathrm{s}_{j+1}}} w_i J^T_{1,i} J_{1,i}\right)^{-1}\right) - \sum_{j=1}^{N^\mathrm{s}-1} \Phi(C_j),$$

if pseudo states of the form (4.28) are used.

Remark 4.17. Numerical experience with the telescoping sum for the pseudo state formulations (4.27) and (4.28) has been disappointing. We conjecture that this is due to the fact that evaluation of the objective takes place where not enough information is accumulated through measurements. Then, the results highly depend on the regularization term H_0.

Part III.

Sequential quadratic programming

Chapter 5.

Preliminary considerations: SQP and multiple shooting

In the previous chapters, we have shown how to transform OED and OC problems to NLPs using DMS. The following chapters deal with the design and implementation of an efficient sequential quadratic programming (SQP) method for their solution. In this introductory chapter, we recapitulate the special class of NLPs that we want to solve and motivate the development of our method with indefinite Hessian approximations by means of two model examples.

5.1. Requirements for an SQP method for direct multiple shooting

Recall the NLP (4.9) resulting from a DMS approach for OED. It has the following properties, which mostly apply to the case of OC problems as well:

- **Sparse and structured Jacobian.** The Jacobian has a banded (sub-)structure that is induced by the continuity conditions and the discretized path- and consistency constraints. There is additional sparsity within each block due to the special structure of OED.
- **Expensive derivatives.** The evaluation of constraint derivatives is usually computationally expensive because for OED it requires evaluation of second-order sensitivities of the nonlinear dynamics, cf. Sec. 2.3.1. The evaluation of the Hessian would require third-order sensitivities and sophisticated IND schemes for their efficient evaluation.
- **Sparse, block-diagonal Hessian.** The Hessian is sparse with a block-diagonal structure because of the partial separability of the Lagrangian. The lower right block can be computed cheaply compared to all other blocks because no sensitivities are involved.
- **Nonconvexity.** The problems have a relatively small constraint nullspace because of the large number of equality constraints and potentially active control bounds. Only on the constraint nullspace is the Hessian required to be positive semidefinite at the solution. The full Hessian depends, e.g., on the curvature of the states and is usually indefinite, which implies that the problems are nonconvex.

From these properties, we derive the following requirements for an efficient SQP method:

1. Sparsity needs to be exploited to reduce the cost for the linear algebra. This issue is addressed in Chapter 8 where sparsity is exploited by means of a Schur complement approach for quadratic programming.
2. Exact Hessians are too costly, instead they need to be approximated by suitable quasi-Newton updates.
3. The quasi-Newton updates need to maintain the block-diagonal structure to preserve sparsity. Tailored quasi-Newton updates are discussed in Chapter 7.

The aspect of nonconvexity has received little attention in the context of DMS. In the following sections, we analyze two model problems to show that using only positive definite Hessians can result in poor convergence. Hence, an additional requirement is:

4. The globalization strategy and the QP solver must accept indefinite Hessian approximations.

Note that this requirement is substantial for both the globalization strategy and the QP solver: Most line search methods rely on the positive definiteness of the Hessian to guarantee descent in a merit function [154]. We present a filter line search globalization that can handle indefinite Hessians in Chapter 6. Furthermore, while convex QPs can be solved in polynomial time [138], allowing indefinite Hessians makes the resulting QP nonconvex and solving it is an NP-hard problem [49, 162].

5.2. Multiple shooting and the lifted Newton method

Multiple shooting can be seen as an application of the *lifted Newton* method introduced in [5]. The idea is the following: We consider the problem of solving a nonlinear system of equations:

$$F(y) = 0, \tag{5.1}$$

where the evaluation of the function $F \in \mathcal{C}^1(\mathbb{R}^{n_\mathrm{u}}, \mathbb{R}^{n_\mathrm{u}})$ is given in the form of a possibly complex algorithm with several intermediate variables. Denoting these intermediate variables by $x_i \in \mathbb{R}^{n_i}$ for $i = 1, 2, \ldots m$ and disregarding further internal structure, we summarize the algorithm in the generic form

$$x_i := f_i(y, x_1, x_2, \ldots, x_{i-1}) \quad \text{for} \quad i = 1, 2, \ldots, m.$$

The final output $F(y)$ is given by

$$f_F(y, x_1, x_2, \ldots, x_m). \tag{5.2}$$

It is straightforward to see that the original system (5.1) is equivalent to the "lifted" system of nonlinear equations

$$G(y,x) = 0 \tag{5.3}$$

with $n_{\mathrm{x}} = \sum_{i=1}^{m} n_i$, $x = (x_1, x_2, \ldots, x_m)$ and where $G \in \mathcal{C}^1(\mathbb{R}^{n_{\mathrm{u}}} \times \mathbb{R}^{n_{\mathrm{x}}}, \mathbb{R}^{n_{\mathrm{u}}} \times \mathbb{R}^{n_{\mathrm{x}}})$ is given by

$$G(u,x) = \begin{pmatrix} f_1(y) - x_1 \\ f_2(y,x_1) - x_2 \\ \vdots \\ f_m(y,x_1,x_2,\ldots,x_{m-1}) - x_m \\ f_F(y,x_1,x_2,\ldots,x_m) \end{pmatrix}. \tag{5.4}$$

The lifted Newton method now solves the original n_{u}-dimensional problem (5.1) by applying Newton's method to the lifted $(n_{\mathrm{u}} + n_{\mathrm{x}})$-dimensional problem (5.3). If structure-exploiting methods like those proposed in [5] are used, the costs of one iteration in the original and in the lifted space are similar. On the other hand, Newton's method in the lifted space often exhibits a significantly improved local convergence rate compared to the original, smaller space.

Going from direct single shooting to DMS can be seen as a lifting of the evaluation of the objective function. We introduce shooting variables and corresponding constraints that ensure equivalence of the solutions. The same structure as in (5.4) is then found in the constraints of the optimization problem. Note, however, that a Newton-type method in optimization is applied to the KKT conditions. There, for every shooting variable we introduce a Lagrange multiplier for the corresponding continuity condition. So for every additional shooting variable the iteration space for Newton's method grows by two. Therefore, in the iteration space of the optimization problem, (5.4) appears only as substructure but the idea is the same: We iterate in a higher-dimensional space to obtain better convergence properties while using structure-exploiting linear algebra that make the additional computational cost for one iteration in the higher-dimensional system negligible.

In this work, we explicitly allow the shooting grid τ^{s} and the control grid τ^{c} to be different. If we translate this in terms of the lifted Newton method it means that for a given control grid τ^{c} different liftings can be constructed, each represented by a different shooting grid τ^{s}. This yields a family of NLPs that all have the same solutions but that will cause different behavior of the same quasi-Newton method. This is a special case of a more general concept that is given by the following definition.

Definition 5.1 (Equivalent NLPs)**.** Consider two nonlinear programming problems

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_1}} \quad & \varphi_1(x) \\ \text{s.t.} \quad & c_1(x) = 0, \end{aligned} \qquad \begin{aligned} \min_{y \in \mathbb{R}^{n_2}} \quad & \varphi_2(y) \\ \text{s.t.} \quad & c_2(y) = 0, \end{aligned}$$

with constraint functions $c_1 : \mathbb{R}^{n_1} \to \mathbb{R}^{m_1}$ and $c_2 : \mathbb{R}^{n_2} \to \mathbb{R}^{m_2}$ for which LICQ holds. We denote the feasible sets by Ω_1 and Ω_2. Assume further, that $n_1 \leq n_2$ and $m_1 \leq m_2$. We call the two NLPs *equivalent*, if the following is true:

1. For local solutions x^* and y^*, the dimensions of the constraint nullspaces are identical, i.e., $n_1 - m_1 = n_2 - m_2$.

2. There exists a bijection $h : \Omega_2 \to \Omega_1$ such that

$$\varphi_1(h(y)) = \varphi_2(y), \quad \text{for all } x \in \Omega_2.$$

Setting up an OED or OC problem with a fixed control grid but different multiple shooting grids yields a family of equivalent NLPs in the sense of Def. 5.1. However, it is not clear how the selection of a shooting grid or lifting influences the convergence behavior of the solution method. In [5] it is shown for a model root finding problem that Newton's method with exact derivatives yields a faster local convergence rate for the lifted problem than for the original one.

In the following, we consider NLPs and investigate how a quasi-Newton method with positive definite Hessians is affected by lifting. We start with a theoretical analysis of an academic example that shows the typical behavior we often see in DMS for OED problems. Then an OED model problem is constructed and analyzed. Numerical experiments confirm the theoretical findings in a more realistic setting.

5.3. Nonconvexity in block-structured problems

In this section we investigate the local convergence behavior of quasi-Newton methods on two model problems with block-diagonal Hessians which were first discussed in a paper [125] that was prepared within the PhD project. Quasi-Newton methods approximate the exact Hessians by simple low-rank updates. For block-diagonal Hessians these updates can be applied to every block independently. Thus the sparse block structure is maintained which would be destroyed by a full-space update. A detailed discussion of quasi-Newton approximations and modifications follows in Chapter 7.

For the following examples we use the positive definite block-BFGS update with the standard damping strategy due to Powell [165] and the block-SR1 update, that may become indefinite, to illustrate how positive definite Hessian approximations can prevent fast local convergence when the Hessian contains negative eigenvalues. We use the standard full-memory forms of these updates that can be found in many textbooks, e.g., [154, Ch. 18.3].

5.3.1. Academic example

We consider the simple unconstrained, convex minimization problem

$$\min_{x_1 \in \mathbb{R}} \quad \tfrac{1}{2} x_1^2.$$

After adding a second variable x_2 along with the constraint $x_1 = x_2$, we obtain the equivalent problem

$$\min_{x_1,x_2\in\mathbb{R}} \quad x_1^2 - \tfrac{1}{2}x_2^2 \tag{5.5a}$$

$$\text{s.t.} \quad x_1 - x_2 = 0. \tag{5.5b}$$

Because the Lagrangian function of the NLP (5.5) is partially separable, its Hessian,

$$\nabla^2_{xx}\mathcal{L}(x,\lambda) = \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix},$$

is block diagonal. Furthermore, it has a negative eigenvalue in the range space of (5.5b). We now study the convergence behavior of different quasi-Newton methods.

Lemma 5.2. *Consider a full-step quasi-Newton SQP method for the solution of* (5.5) *with iterates* $(x^{[k]}, \lambda^{[k]})$*, and corresponding Hessian approximations* $B^{[k]}$ *for* $k = 0, 1, 2, \ldots$*. Let* $B^{[0]} = \mathbb{I}$ *and* $x^{[0]}$ *and* $\lambda^{[0]}$ *be arbitrary. Then the following holds:*

1. *After one iteration, the linear constraint* (5.5b) *is satisfied, i.e.* $x_1^{[k]} = x_2^{[k]}$*,* $k \geq 1$*.*
2. *If* $B^{[k]} = \nabla^2_{xx}\mathcal{L}(x^{[k]}, \lambda^{[k]})$*, the NLP* (5.5) *is solved after one iteration with solution* $x_1^* = x_2^* = \lambda^* = 0$*.*
3. *If block-SR1 updates are used, the two Hessians blocks are recovered after one iteration, leading to convergence in two iterations.*
4. *If block-BFGS updates are used, we obtain for* $k \geq 1$*:*

$$B^{[k]} = \begin{pmatrix} 2 & 0 \\ 0 & a^{[k]} \end{pmatrix}, \quad a^{[k]} > 0,$$

with $\{a^{[k]}\} \to 0$*.*

Proof. 1 and 2 are easily verified by computing one Newton step. 3 follows from the secant equation $B^{[k+1]}\delta^{[k]} = \gamma^{[k]}$ that motivates the update formulae (cf. Eq. (7.1)). 4 follows from the BFGS damping strategy: The first block is exactly recovered after one iteration. For the second block, the undamped update would yield -1, that means the damping strategy becomes active. It interpolates between the current, positive approximation and the unmodified updates to guarantee a positive definite matrix, see Sec. 7.2. This yields a sequence $a^{[k]} > 0$, with $\{a^{[k]}\} \to 0$. □

Let us now further analyze the convergence behavior of the block-BFGS method.

Theorem 5.3. *A full-step quasi-Newton method with block-BFGS updates applied to problem* (5.5) *converges* Q-linearly *with convergence rate* $R = \frac{1}{2}$*.*

Proof. Let us assume $k \geq 1$. Then by Lemma 5.2 we know that $x_1^{[k]} = x_2^{[k]}$ and

$$B^{[k]} = \begin{pmatrix} 2 & 0 \\ 0 & a^{[k]} \end{pmatrix}, \quad a^{[k]} > 0.$$

The iterate $x^{[k+1]} = x^{[k]} + \Delta x^{[k]}$ and $\lambda^{[k+1]}$ can be computed by solving the KKT system of the corresponding QP. Suppressing the index k, this reads as

$$\left(\begin{array}{cc|c} 2 & & 1 \\ & a & -1 \\ \hline 1 & -1 & \end{array}\right) \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \lambda \end{pmatrix} = \begin{pmatrix} -2x_1 \\ x_1 \\ 0 \end{pmatrix}. \tag{5.6}$$

This yields

$$\Delta x_1 = \Delta x_2 = -\frac{x_1}{2+a}, \qquad \lambda = -2x_1 \frac{1+a}{2+a}.$$

The rate of convergence for the components of x, x_1 and x_2, is given by

$$\frac{\left|x_i^{[k+1]} - x_i^*\right|}{\left|x_i^{[k]} - x_i^*\right|} = \left|x_i^{[k]} - \frac{x_i^{[k]}}{2+a^{[k]}}\right| \cdot \left|\frac{1}{x_i^{[k]}}\right| = 1 - \frac{1}{2+a^{[k]}} =: R$$

Because $a^{[k]} > 0$, this implies that the block-BFGS method converges linearly for problem (5.5) with convergence rate $R = \frac{1}{2}$. □

This result means that if, e.g., $x^{[0]} = 1$ we need 40 iterations to achieve an accuracy of 10^{-12} because $x^{[40]} \geq 2^{-40} \approx 9.1 \cdot 10^{-13}$; a substantial difference compared to the block-SR1 method that only needs 2 iterations. Although problem (5.5) is technically not a lifted problem in the sense of lifted Newton, we may conclude that approximating negative definite blocks by positive definite Hessians can impair local convergence even for simple problems.

Note that for this example, damping is necessary because the individual blocks are approximated independently. The full-space BFGS could proceed without damping. Its behavior for Example (5.5) can be summarized as follows:

Lemma 5.4. *Consider a full-step quasi-Newton SQP method applied to Problem* (5.5) *with initial Hessian approximation $B^{[0]} = \mathbb{I}$ and feasible starting values $x_1^{[0]} = x_2^{[0]} \neq 0$. Then the Hessian approximation after the first step obtained by a full-space BFGS update is the positive definite matrix*

$$B^{[1]} = \begin{pmatrix} 4.5 & -2.5 \\ -2.5 & 1.5 \end{pmatrix},$$

and the solution of (5.5) *is recovered after one additional iteration.*

Proof. We compute the first step by solving the KKT system (5.6) with $\mathbb{I}$ in the upper left block. This yields the step

$$\Delta x^{[0]} = -\frac{1}{2}\left(x_1^{[0]}, \quad x_1^{[0]}\right)^T.$$

and the new iterate

$$x^{[1]} = \frac{1}{2}\left(x_1^{[0]}, \quad x_1^{[0]}\right)^T.$$

For the BFGS update, we need the difference of the gradient of the Lagrangian, $\gamma^{[0]}$. Because the constraint is linear, we can ignore the current iterate of λ and obtain

$$\gamma^{[0]} = \nabla\mathcal{L}(x^{[1]}, \lambda^{[1]}) - \nabla\mathcal{L}(x^{[0]}, \lambda^{[1]}) = \left(-x_1^{[0]}, \quad \tfrac{1}{2}x_1^{[0]}\right)^T.$$

We now apply the BFGS formula to compute $B^{[1]}$. Omitting the index $^{[0]}$, it reads as

$$B^{[1]} = B - \frac{B\Delta x\Delta x^T B}{\Delta x^T B\Delta x} + \frac{\gamma\gamma^T}{\gamma^T\Delta x}$$

With $\Delta x = \Delta x^{[0]}$, $\gamma = \gamma^{[0]}$ as given above and $B = \mathbb{I}$ we obtain the matrix $B^{[1]}$ as desired.

We now compute the next iterate of the quasi-Newton method by solving the KKT system

$$\left(\begin{array}{cc|c} 4.5 & -2.5 & 1 \\ -2.5 & 1.5 & -1 \\ \hline 1 & -1 & \end{array}\right)\begin{pmatrix}\Delta x_1^{[1]} \\ \Delta x_2^{[1]} \\ \lambda^{[2]}\end{pmatrix} = \frac{1}{2}\begin{pmatrix}-2x_1^{[0]} \\ x_1^{[0]} \\ 0\end{pmatrix}.$$

This yields

$$\Delta x_1^{[1]} = \Delta x_2^{[1]} = -\frac{1}{2}x_1^{[0]}, \qquad \lambda^{[2]} = 0,$$

and thus the solution $x_1^{[2]} = x_2^{[2]} = 0$ of problem (5.5). □

Note that for the full-space BFGS update, damping is not necessary because the curvature condition

$$\gamma^{[0]^T}\Delta x^{[0]} = \tfrac{1}{4}x_1^{[0]} > 0$$

is satisfied which implies that the BFGS update produces a positive definite matrix. This is not the case when applying the update independently to the diagonal blocks.

This simple example shows that the full space BFGS can be very efficient although the matrices it produces may be very different from the true Hessian. However, on larger problems with a block-diagonal Hessian it is often unable to accumulate enough curvature information and it does not represent a suitable way to solve problems parameterized by DMS, see the results in Sec. 10.4.

5.3.2. An OED model problem

We now discuss a minimal, but prototypical OED example that allows an analytical investigation. We consider a simple form of the NLP (4.9) that results from a DMS parameterization of an OED problem. Assume that $\tau^{\mathrm{s}} = \tau^{\mathrm{m}} = \tau^{\mathrm{c}}$, the controls are discretized piecewise constant, and that the ODE states can observed directly with all $w_i = 1$. Furthermore, assume that no additional nonlinear constraints are present. The NLP then reads as

$$\begin{aligned}
\min_{q_j, s_j, S_j, H} \quad & \operatorname{tr}(H^{-1}) && && \text{(5.7a)}\\
\text{s.t.} \quad & 0 = y(\tau_{j+1}; s_j, q_j) - s_{j+1} && 0 \le j \le N^{\mathrm{s}} - 1, && \text{(5.7b)}\\
& 0 = y_0 - s_0, && && \text{(5.7c)}\\
& 0 = y_{\mathrm{p}}(\tau_{j+1}; s_j, S_j, q_j) - S_{j+1} && 0 \le j \le N^{\mathrm{s}} - 1, && \text{(5.7d)}\\
& 0 = S_0, && && \text{(5.7e)}\\
& b_\ell \le q_j \le b_u && 0 \le j \le N^{\mathrm{s}}, && \text{(5.7f)}\\
& 0 = \textstyle\sum_{j=0}^{N^{\mathrm{s}}} S_j^T S_j - H. && && \text{(5.7g)}
\end{aligned}$$

We now consider the ODE system

$$\dot{y}(t) = p \cdot u(t), \quad y(t_0) = 0, \tag{5.8}$$

with a scalar parameter p and given estimate $\hat{p} = 10$ on the time horizon $[t_0, t_{\mathrm{f}}] = [0, 1]$. Let the control u be constrained by $0 \le u(t) \le 2$. The variational differential equation corresponding to (5.8) is

$$\dot{y_{\mathrm{p}}}(t) = u(t), \quad y_{\mathrm{p}}(t_0) = 0.$$

The ODE solutions on the shooting intervals $[\tau_j^{\mathrm{s}}, \tau_{j+1}^{\mathrm{s}}]$, $0 \le j < N^{\mathrm{s}}$ have the representations

$$y(\tau_{j+1}^{\mathrm{s}};\ \tau_j^{\mathrm{s}}, s_j, q_j) = s_j + p q_j (\tau_{j+1}^{\mathrm{s}} - \tau_j^{\mathrm{s}}), \tag{5.9a}$$

$$y_{\mathrm{p}}(\tau_{j+1}^{\mathrm{s}};\ \tau_j^{\mathrm{s}}, s_j, S_j, q_j) = S_j + q_j (\tau_{j+1}^{\mathrm{s}} - \tau_j^{\mathrm{s}}). \tag{5.9b}$$

The global minimum of this problem is $\operatorname{tr}((H^*)^{-1}) = 6.49 \cdot 10^{-2}$. It is attained at $q_j = 2$, $0 \le j < N^{\mathrm{c}}$. This can be seen by applying a maximum principle [171]. This means that the system reveals the most information about the parameter p if we excite it by the maximum amount possible.

If we insert Equations (5.9) into the NLP (5.7), we note that the only nonlinear constraint is (5.7g); its Lagrange multiplier is denoted by $\lambda_{\mathrm{H}} \in \mathbb{R}^{n_{\mathrm{p}}}$. We define the variables

$$\xi = \begin{bmatrix} \xi_1^T & \xi_2^T & \cdots & \xi_{N^{\mathrm{s}}}^T & H \end{bmatrix}^T, \qquad \xi_j = \begin{pmatrix} s_j & S_j & q_j \end{pmatrix}^T, \quad j = 0, \ldots, N^{\mathrm{s}}.$$

The Hessian of the Lagrangian of (5.7) has the following diagonal block structure:

$$\begin{aligned}
\nabla^2_{\xi,\xi} \mathcal{L}(\xi, \lambda) &= \operatorname{diag}\left(\nabla^2_{\xi_0,\xi_0} \mathcal{L}(\xi, \lambda), \ldots, \nabla^2_{\xi_{N^{\mathrm{s}}},\xi_{N^{\mathrm{s}}}} \mathcal{L}(\xi, \lambda), \nabla^2_{H,H} \mathcal{L}(\xi, \lambda) \right),\\
\nabla^2_{\xi_j,\xi_j} \mathcal{L}(\xi, \lambda) &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2\lambda_{\mathrm{H}} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \nabla^2_{H,H} \mathcal{L}(\xi, \lambda) = 2H^{-3}.
\end{aligned} \tag{5.10}$$

N^{s}	τ^{s}	BFGS	SR1
2	$(0,1)$	32	25
3	$(0,0.5,1)$	38	19
6	$(0,0.2,0.4,0.6,0.8,1)$	31	15
11	$(0,0.1,0.2,\ldots,1)$	41	12

Table 5.1.: Number of SQP iterations taken to solve example (5.7) to a KKT tolerance of 10^{-9}. N^{s} is the number of multiple shooting nodes. Iteration counts are reported for a fullstep SQP method with block-BFGS and block-SR1 updates. Identity is taken as initial approximation.

While $2H^{-3}$ is positive close to the optimal solution, $2\lambda_{\mathrm{H}}$ is not. In our example, we have $\lambda_{\mathrm{H}}^{*} = -4.22 \cdot 10^{-3}$. Consequently, the performance of the SQP algorithm can be impaired significantly when the approximation of the Hessian is forced to be positive definite on every block.

To illustrate this, we choose a grid of 11 equidistant points on $[0,1]$, $\tau_j = 0.1\,j$, with $0 \leq j \leq 10$. As initial guess, we set $q_j = 0.6$ and we initialize s_j and S_j such that all continuity conditions (5.9) are satisfied. The initial Hessian approximation is identity. With this setup, our full-step SQP implementation finds the solution in 41 iterations with damped block-BFGS updates. In all iterations, 10 of the block-BFGS updates are damped to preserve positive definiteness of the approximation, or they are skipped entirely due to ill-conditioning as described in Section 7.2. In comparison, a full-step SR1-SQP algorithm needs only 12 iterations to converge.

We now keep the control discretization fixed and choose coarser multiple grids. This results in a family of equivalent NLPs as discussed in the previous section. Because of the coarser discretization, the Hessian has fewer diagonal blocks. Table 5.1 shows the number of SQP iterations taken for these NLPs. We see that the negative impact of block-BFGS damping is most severe for the finest multiple shooting grid. At the same time, the number of SQP iterations decreases with finer grids for the block-SR1 updates. Here, more diagonal blocks allow the high-rank block updates to introduce more curvature information per iteration which provides rapid local convergence.

Chapter 6.

A filter line search SQP method with indefinite Hessians

In this chapter, we present a new line search SQP method that can work with indefinite Hessian matrices. It is based on the filter line-search globalization proposed by Wächter and Biegler in [198]. Our algorithm has an inner loop in which several, possibly nonconvex, QPs are solved until a positive definiteness condition motivated by the proof of global convergence is satisfied. We first give a general description of the algorithm. Then we recapitulate the backtracking line search procedure from [198] and include two heuristics that have proven efficient in practice. Finally, we give details about our feasibility restoration phase including a new, efficient heuristic for problems arising in DMS. A paper describing the SQP algorithm is submitted for publication [125].

6.1. The algorithm

In this and the following chapters we are interested in solving nonlinear programs of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \varphi(x) && \text{(6.1a)}\\ \text{s.t.} \quad & c(x) = 0, && \text{(6.1b)}\\ & b_\ell \le x \le b_u. && \text{(6.1c)} \end{aligned}$$

where $\varphi : \mathbb{R}^n \to \mathbb{R}$, $c : \mathbb{R}^n \to \mathbb{R}^m$, and $b_\ell, b_u \in \mathbb{R}^n \cup \{-\infty, \infty\}^n$. NLPs involving general inequality constraints $c_i(x) \ge 0$ can always be transformed into this form by introducing slack variables $s_i \le 0$ and setting $c_i(x) + s_i = 0$.

As outlined in Section 1.2, we start with an initial guess $x^{[0]}$ and compute iterates $x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}$ in SQP iteration $k \ge 0$. The step size $\alpha^{[k]}$ is determined by a filter line search described in Sec. 6.2 below. The search direction $d^{[k]}$ is given as the solution of the quadratic program QP($B^{[k]}$):

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T B^{[k]} d + {g^{[k]}}^T d && \text{(6.2a)}\\ \text{s.t.} \quad & A^{[k]} d + c^{[k]} = 0, && \text{(6.2b)}\\ & b_\ell \le x^{[k]} + d \le b_u. && \text{(6.2c)} \end{aligned}$$

In the QP (6.2), $g^{[k]}$ is the objective gradient $\nabla\varphi(x^{[k]})$, $c^{[k]}$ is the evaluation of the constraints $c(x^{[k]})$, the matrix $A^{[k]}$ is the constraint Jacobian $\nabla c(x^{[k]})^T$, and the matrix $B^{[k]}$ is an approximation of the Hessian of the Lagrangian function. We define $\lambda^{[k]}$ as the current iterate of the Lagrange multipliers and $\bar{\lambda}^{[k]} \in \mathbb{R}^{m+n}$ as the Lagrange multipliers of (6.2). The algorithm is stopped with solution $(x^{[k]}, \lambda^{[k]})$ if the following convergence criterion is satisfied

$$\frac{\left\|\nabla\mathcal{L}(x^{[k]}, \lambda^{[k]})\right\|_\infty}{1 + \left\|\lambda^{[k]}\right\|_\infty} < \varepsilon_{\text{opt}} \qquad \text{and} \qquad \frac{\left\|c(x^{[k]})\right\|_\infty}{1 + \left\|x^{[k]}\right\|_\infty} < \varepsilon_{\text{feas}} \tag{6.3}$$

with prescribed tolerances $\varepsilon_{\text{opt}}, \varepsilon_{\text{feas}} > 0$.

A key question is under which conditions a solution to QP($B^{[k]}$) exists and when we can expect the line search to find an acceptable step size along this solution. An answer is given in the global convergence proof for the filter line search [198]:

Under the assumption that the approximations $B^{[k]}$ are uniformly positive definite on a certain subspace, it is guaranteed that a solution of (6.2) exists and that the resulting search direction has descent properties that are required for the filter line search. Uniform positive definiteness means that the smallest eigenvalue is bounded away from zero for the entire sequence of reduced Hessians. In order to formulate the positive-definiteness condition, let us define $\mathcal{S}^{[k]}$ as the set of bounds that are active at *both* $x^{[k]}$ *and* $x^{[k]} + d^{[k]}$:

$$\mathcal{S}^{[k]} := \left\{ 1 \le i \le n \mid (x_i^{[k]} = b_{\ell,i} \text{ or } x_i^{[k]} = b_{u,i}) \text{ and } d_i^{[k]} = 0 \right\}. \tag{6.4}$$

We denote by $\mathbb{I}_{\mathcal{S}^{[k]}} \in \mathbb{R}^{|\mathcal{S}^{[k]}| \times n}$ the identity matrix with the rows deleted whose indices are not in $\mathcal{S}^{[k]}$. We define

$$A_{\mathcal{S}}^{[k]} := \begin{bmatrix} A^{[k]} \\ \mathbb{I}_{\mathcal{S}^{[k]}} \end{bmatrix}.$$

With these definitions in place, we can formally state the assumption.

Assumption 6.1. The Hessian approximations $B^{[k]}$ are uniformly positive definite on the null space of $A_{\mathcal{S}}^{[k]}$.

In other words, there exists a constant $M_B > 0$ so that for all k we have

$$\lambda_{\min}\left(Z^{[k]T} B^{[k]} Z^{[k]}\right) \ge M_B,$$

where $Z^{[k]}$ is a basis of the null space of $\mathcal{A}_{\mathcal{S}}^{[k]}$ and $\lambda_{\min}$ denotes the smallest eigenvalue. This assumption is necessary to ensure sufficient decrease of the objective at points with small infeasibility. If the method converges to a solution that satisfies the strong second-order conditions (see Thm. 1.8), the active set finally becomes unchanged and $Z^{[k]}$ is a basis for the active constraints. Global convergence can be shown if, among further assumptions such as boundedness of the functions and well-definedness of the restoration phase, Assumption 6.1 (named (G3*) in [198]) holds.

Similar to the practical approaches taken in [194, 199], we do not guarantee uniform positive definiteness for the entire sequence $\{B^{[k]}\}$, because this would require an excessively large amount of time for the computation of eigenvalues. Instead, our SQP method guarantees positive definiteness of $B^{[k]}$ on the null space of $A_{\mathcal{S}}^{[k]}$ in every individual iteration k separately. We start with a possibly indefinite approximation of the Hessian and attempt to solve (6.2). If (6.2) is infeasible, the method switches to the restoration phase. Otherwise, we obtain a solution $d^{[k]}$ and check whether $B^{[k]}$ is positive definite on the null space of $A_{\mathcal{S}}^{[k]}$. In this case, the filter line search is continued with $d^{[k]}$. If $B^{[k]}$ is not positive definite on the null space of $A_{\mathcal{S}}^{[k]}$, a different Hessian approximation is chosen and a new direction $d^{[k]}$ is computed. In Section 7.1, possible sequences of approximations are discussed.

Algorithm 6 gives the full description of this SQP method. The backtracking line search in Step 3 and the feasibility restoration phase in Step 5 are described in Sec. 6.2 and Sec. 6.3, respectively. The algorithm is similar to Algorithm I in [198], with an additional inner loop in Step 2. Within this loop, indexed by $l \geq 0$, different Hessian approximations are tried.

6.2. Filter line search procedure

The step size $\alpha^{[k]}$ is determined by the filter line search described in [198]. It is also implemented in the interior point code IPOPT that is described in [199]. Its global and local convergence properties are analyzed in [198] and [197], respectively.

Broadly speaking, a step is accepted if it sufficiently reduces either the objective value $\varphi(x)$ or the constraint violation $\eta(x) := \|c(x)\|_\infty$, see Sec. 1.2.5. In a series of trial step sizes $1 = \alpha^{[k,0]} > \alpha^{[k,1]} > \cdots > \alpha^{[k,l_{\max}]}$, a new iterate $x^{[k+1,l]} = x^{[k]} + \alpha^{[k,l]} d^{[k]}$ is accepted if it leads to sufficient progress towards either goal compared to the current iterate, i.e., if

$$\eta(x^{[k+1,l]}) \leq (1 - \beta_\eta^{\mathcal{F}})\eta(x^{[k]}), \quad \text{or} \tag{6.5a}$$

$$\varphi(x^{[k+1,l]}) \leq \varphi(x^{[k]}) - \beta_\varphi^{\mathcal{F}} \eta(x^{[k]}) \tag{6.5b}$$

with fixed constants $\beta_\varphi^{\mathcal{F}}, \beta_\eta^{\mathcal{F}} \in (0,1)$. Figure 6.1 illustrates condition (6.5) and the concept of a filter. The above criterion is replaced by requiring sufficient progress in the objective function, whenever the current iterate is sufficiently feasible, i.e., we have $\eta(x^{[k]}) \leq \eta^{\min}$, with $\eta^{\min} \in (0, \infty]$, and the following switching condition

$$\nabla\varphi(x^{[k]})^T d^{[k]} < 0 \qquad \text{and} \qquad \alpha^{[k,l]} \left[-\nabla\varphi(x^{[k]})^T d^{[k]}\right]^{s_\varphi} > \delta_\eta \left[\eta(x^{[k]})\right]^{s_\eta} \tag{6.6}$$

with constants $\delta_\eta > 0$, $s_\eta > 1$, $s_\varphi \geq 1$ holds. If $\eta(x^{[k]}) \leq \eta^{\min}$ and (6.6) is true for the current step size $\alpha^{[k,l]}$, the trial point has to satisfy the Armijo condition

$$\varphi(x^{[k+1,l]}) \leq \varphi(x^{[k]}) + \eta_\varphi \alpha^{[k,l]} \nabla\varphi(x^{[k]})^T d^{[k]} \tag{6.7}$$

instead of (6.5), in order to be acceptable. Here, $\eta_\varphi \in (0, \frac{1}{2})$ is a constant.

The algorithm maintains a filter $\mathcal{F}^{[k]} \subset \{(\eta, \varphi) \in \mathbb{R}^2 \mid \eta \geq 0\}$ for each iteration k. The filter contains those combinations of constraint violation values η and objective function

Algorithm 6: Filter Line Search SQP Algorithm with Indefinite Hessian Matrices

Given: Termination tolerances $\varepsilon_{\text{opt}}, \varepsilon_{\text{feas}} > 0$, starting point $(x^{[0]}, \lambda^{[0]})$, initial Hessian $B^{[0]}$. Evaluate $\varphi(x^{[0]}), c^{[0]}, g^{[0]}, A^{[0]}$. Set $k = 0$.

1. *Check convergence.* Stop if $(x^{[k]}, \lambda^{[k]})$ satisfies the KKT-tolerance given by (6.3).
2. *Compute search direction.* Define a sequence of trial Hessian matrices $B^{[k,0]}, \ldots, B^{[k,l_{\max}]}$ where $B^{[k,l_{\max}]}$ is positive definite.
   For $l = 0, 1, \ldots, l_{\max}$
   a) Solve QP($B^{[k,l]}$) to obtain a primal-dual critical point $(d^{[k,l]}, \bar{\lambda}^{[k,l]})$.
   b) If QP($B^{[k,l]}$) is infeasible, go to 5.
   c) If a critical point of QP($B^{[k,l]}$) is found and if $B^{[k,l]}$ is positive definite on $A_S^{[k]}$, set $(d^{[k]}, \bar{\lambda}^{[k]}) = (d^{[k,l]}, \bar{\lambda}^{[k,l]})$, set $B^{[k]} = B^{[k,l]}$, and go to 3.

   End for.
3. *Backtracking line search.* Try to find a step size $\alpha^{[k]}$ by the filter line search described in Algorithm 7. If the trial step size becomes too small, go to 5.
4. *Next iteration.*
   Set
   $$x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}, \qquad \lambda^{[k+1]} = (1 - \alpha^{[k]})\lambda^{[k]} + \alpha^{[k]} \bar{\lambda}^{[k]}.$$
   Evaluate
   $$\varphi(x^{[k+1]}),\ g^{[k+1]},\ c^{[k+1]},\ A^{[k+1]}.$$
   For the computation of quasi-Newton updates, store
   $$\gamma^{[k]} = \nabla \mathcal{L}(x^{[k+1]}, \lambda^{[k+1]}) - \nabla \mathcal{L}(x^{[k]}, \lambda^{[k+1]}), \qquad \delta^{[k]} = x^{[k+1]} - x^{[k]}.$$
   Set $k \leftarrow k + 1$, and go to 1.
5. *Feasibility restoration phase.* If possible, compute $x^{[k+1]}$ that is accepted by the filter and go to 1. Otherwise, stop and declare the problem as first-order locally infeasible.

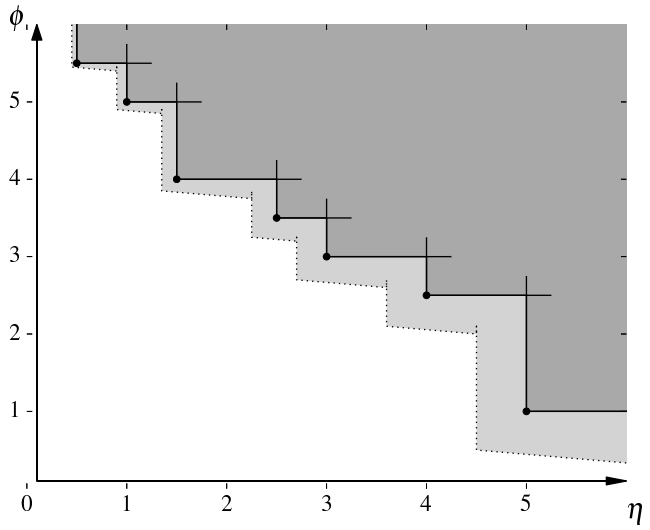


Figure 6.1.: A filter defined by a set of iterates (dots) in the (φ, η) plane. The dotted line is a slanting envelope defined by $\beta_\eta^{\mathcal{F}} = \beta_\varphi^{\mathcal{F}} = 0.1$, that represents condition (6.5). The dark gray area corresponds to the set of dominated points that, together with the light gray area, corresponds to the prohibited region defined by the filter.

values φ that are *prohibited* for a successful trial point in iteration k. That means, a trial point $x^{[k+1,l]}$ is rejected during the line search, if $(\eta(x^{[k+1,l]}), \varphi(x^{[k+1,l]})) \in \mathcal{F}^{[k]}$. We then say that the trial point is not acceptable to the current filter. We initialize the filter to

$$\mathcal{F}^{[0]} = \{(\eta, \varphi) \in \mathbb{R}^2 \mid \eta \geq \eta^{\max}\}$$

for some $\eta^{\max}$. This means that the algorithm will never allow trial points that have a constraint violation larger than $\eta^{\max}$. Later, the filter is augmented in some iterations, using the update formula

$$\mathcal{F}^{[k+1]} := \mathcal{F}^{[k]} \cup \left\{(\eta, \varphi) \in \mathbb{R}^2 \mid \eta \geq (1 - \beta_\eta^{\mathcal{F}})\eta(x^{[k]}) \text{ and } \varphi \geq \varphi(x^{[k]}) - \beta_\varphi^{\mathcal{F}}\eta(x^{[k]})\right\}, \tag{6.8}$$

i.e., pairs are added which do not decrease neither the constraint violation nor the objective function sufficiently.

A feasibility restoration phase may be necessary to generate iterates that reduce the constraint violation and are acceptable to the filter when the step size $\alpha^{[k]}$ in direction $d^{[k]}$ falls below a certain threshold. Details on the implemented restoration phase are given in Sec. 6.3 below.

6.2.1. Second-order correction steps

A well-known difficulty in many methods for constrained nonlinear programming is the so-called *Maratos effect*. It was first observed by Maratos [149] and describes the behavior

of a globalization strategy that can interfere with the full step arbitrarily close to a solution thus impeding fast local convergence. One way to overcome this difficulty are *second-order correction steps (SOCS)*. A second-order correction step aims to reduce infeasibility by re-solving the QP (6.2) with a quadratic approximation of the constraints instead of a linear one. Following [154, Ch. 18.3], the QP to be solved is

$$\min_{d\in\mathbb{R}^n} \quad \tfrac{1}{2} d^T B^{[k]} d + {g^{[k]}}^T d \tag{6.9a}$$

$$\text{s.t.} \quad A^{[k]} d + c(x^{[k]} + d^{[k,j]}) - A^{[k]} d^{[k,j]} = 0, \tag{6.9b}$$

$$b_\ell \le x^{[k]} + d \le b_u. \tag{6.9c}$$

for $j = 0, 1, \ldots, j_{\max}$, where $d^{[k,0]} := d^{[k]}$. Note that this requires no additional evaluation of constraint derivatives, only of the constraint function which has to be done anyway in the line search. Furthermore, a QP solver can re-use factorizations of the first QP solved to obtain $d^{[k]}$, which allows efficient computation of SOCS. We denote by $d^{[k]}_{\text{soc}} := d^{[k,j+1]}$ a solution of the QP (6.9).

In the filter line search, we compute a SOCS $d^{[k]}_{\text{soc}}$ if the first trial step size $\alpha^{[k,0]}$ has been rejected and if $\eta(x^{[k+1,0]}) \ge \eta(x^{[k]})$. Then we check if the resulting trial point $x^{[k+1]}_{\text{soc}} := x^{[k]} + d^{[k]}_{\text{soc}}$ is acceptable to the filter and satisfies the filter acceptance criteria. If this trial point passes the tests, it is accepted as the new iterate. Otherwise, we apply additional second-order corrections $j = 1, 2, \ldots$, unless the correction step has not decreased the constraint violation by a fraction $\kappa_{\text{soc}} \in (0, 1)$ or a maximum number $j_{\max}$ of second-order corrections has been performed. In that case, the original search direction $d^{[k]}$ is restored and the regular backtracking line search is resumed with a shorter step size $\alpha^{[k,1]} = \frac{1}{2}\alpha^{[k,0]}$.

Algorithm 7 gives a full description of the filter line search procedure.

6.2.2. Practical modifications

In practice, two modifications of Algorithm 6 have proven to be efficient. First, whenever a QP($B^{[k,l]}$) is infeasible in Step 2b, we reset the Hessian to a scaled identity and attempt to resolve QP($B^{[k,l]}$). Although in theory this does not change the feasible set, we found that in practice a QP is often reported infeasible due to rounding errors caused by ill-conditioned Hessians. The same is done if the step length of the line search becomes too short (Step 3). Here, an ill-conditioned Hessian can produce an inaccurate QP solution and the line search is not guaranteed to find an acceptable step in this direction.

The second modification addresses the fact that the line search is sometimes unable to find a step when the algorithm is very close to a solution and the current value of the convergence criterion (6.3) is already close to the prescribed tolerances ε_{opt} and $\varepsilon_{\text{feas}}$. Then the step will be very small and the progress is sometimes deemed not sufficient by the line search. In this case, we allow the algorithm to take the full step if it reduces the optimality tolerance

$$\frac{\|\nabla \mathcal{L}(x, \lambda)\|_\infty}{1 + \|\lambda_\infty\|}$$

by at least a factor $\kappa_{\text{kkt}} \in (0, 1]$.

Algorithm 7: Backtracking filter line search.

Constant line search parameters: $\alpha_{\min} > 0$,$j_{\max} \in \mathbb{N}$, $\beta_\eta^{\mathcal{F}}, \beta_\varphi^{\mathcal{F}} \in (0,1)$, $\delta_\eta > 0$, $s_\eta > 1$, $s_\varphi \geq 1$, $\eta_\varphi \in (0, \frac{1}{2})$, $\eta^{\min} > 0$, $\eta^{\max} \in (\eta(x^{[0]}), \infty]$, and $\kappa_{\text{soc}} \in (0,1)$.
Given: Direction $d^{[k]}$ obtained from QP($B^{[k]}$) (6.2) at SQP Iteration k. Set $\alpha^{[k,0]} = 1$ and $l = 0$.

1. *Compute the new trial point.* Set $x^{[k+1,l]} := x^{[k]} + \alpha^{[k,l]} d^{[k]}$.
2. *Check acceptability to the filter.* If $(\eta(x^{[k+1,l]}), \varphi(x^{[k+1,l]})) \in \mathcal{F}^{[k]}$, reject the trial step and go to 4.
3. *Check sufficient decrease with respect to the current iterate.*
   - *Case I:* $\eta(x^{[k]}) \leq \eta^{\min}$ *and* (6.6) *holds:* If (6.7) holds, accept the trial step $x^{[k+1,l]}$ and go to 10. Otherwise continue at 4.
   - *Case II:* $\eta(x^{[k]}) > \eta^{\min}$ *or* (6.6) *is not satisfied:* If (6.5) holds, accept the trial step $x^{[k+1,l]}$ and go to 10. Otherwise continue at 4.
4. *Initialize the second-order correction.* If $l > 0$ or $\eta(x^{[k+1,l]}) < \eta(x^{[k]})$, skip the second-order correction (SOC) and continue at 9. Otherwise, set $d^{[k,0]} := d^{[k]}$, initialize the SOC counter $j = 0$, and initialize $\eta_{\text{soc}}^{\text{old}} = \eta(x^{[k]})$.
5. *Compute the second-order correction.* Compute $d_{\text{soc}}^{[k]} = d^{[k,j+1]}$ by solving (6.9) and set $x_{\text{soc}}^{[k+1]} = x^{[k]} + d_{\text{soc}}^{[k]}$.
6. *Check acceptability to the filter (in SOC).* If $(\eta(x_{\text{soc}}^{[k+1]}), \varphi(x_{\text{soc}}^{[k+1]})) \in \mathcal{F}^{[k]}$, reject the trial step and go to 9.
7. *Check sufficient decrease with respect ot the current iterate (in SOC).*
   - *Case I:* $\eta(x^{[k]}) \leq \eta^{\min}$ *and* (6.6) *holds (for* $\alpha^{[k,0]}$*):* If (6.7) holds with "$x^{[k+1,l]}$" replaced by "$x_{\text{soc}}^{[k+1]}$", accept the trial step $x_{\text{soc}}^{[k+1]}$ and go to 10. Otherwise continue at 8.
   - *Case II:* $\eta(x^{[k]}) > \eta^{\min}$ *or* (6.6) *is not satisfied (for* $\alpha^{[k,0]}$*):* If (6.5) holds with "$x^{[k+1,l]}$" replaced by "$x_{\text{soc}}^{[k+1]}$", accept the trial step $x_{\text{soc}}^{[k+1]}$ and go to 10. Otherwise continue at 8.
8. *Next second-order correction.* If $j = j_{\max}$ or $\eta(x_{\text{soc}}^{[k+1]}) > \kappa_{\text{soc}} \eta_{\text{soc}}^{\text{old}}$, abort SOC and continue at 9. Otherwise, increase the SOC counter $j = j + 1$, set $\eta_{\text{soc}}^{\text{old}} = \eta(x_{\text{soc}}^{[k+1]})$ and go back to 5.
9. *Choose new trial step size.* Set $\alpha_{[k,l+1]} = \frac{1}{2}\alpha^{[k,l]}$ and $l = l + 1$. If the trial step size becomes too small, i.e. $\alpha^{[k,l]} < \alpha_{\min}$ try to find an acceptable iterate using a feasibility restoration phase.
10. *Accept the trial point and augment the filter if necessary.* Terminate with step size $\alpha^{[k]} := \alpha^{[k,l]}$ and set $d^{[k]} := d_{\text{soc}}^{[k]}$ if SOC was accepted. If (6.6) or (6.7) do not hold for $\alpha^{[k]}$, augment the filter using (6.8). Otherwise leave the filter unchanged, i.e., set $\mathcal{F}^{[k+1]} := \mathcal{F}^{[k]}$.

6.3. Feasibility restoration phase

Whenever a QP is infeasible or the step size in the line search becomes too small, the algorithm resorts to a feasibility restoration phase (Step 5 of Algorithm 6) to find an iterate that is closer to the feasible region and acceptable for the filter. The feasibility restoration phase also has the purpose to detect local infeasibility. In most algorithms, this is done by minimizing some measure of the constraint violation $\eta(x)$ and we have also included such a procedure in our method. However, minimizing the constraint violation is computationally expensive, as discussed in Sec 6.3.2. In the following section we present a simple and efficient restoration heuristic for problems arising in lifted methods such as DMS that is always invoked before resorting to the standard restoration phase.

6.3.1. A feasibility restoration heuristic for problems arising in direct multiple shooting

The main purpose of the feasibility restoration phase is to deliver a new iterate $x^{[k+1]}$ that provides sufficient reduction, i.e. satisfies (6.5), and is acceptable to the filter, i.e. $(\varphi(x^{[k+1]}), \eta(x^{[k+1]})) \notin \mathcal{F}^{[k+1]}$. The global convergence proof [198] makes no assumptions on the particular procedure. Thus for OED and OC problems parameterized by multiple shooting we can try to reduce the overall constraint violation simply by choosing the shooting variables such that they satisfy the continuity conditions. Algorithm 8 describes the procedure for a simple optimal control problem that has the following form:

$$\min_{s,q} \quad \varphi(s_{N^s}) \tag{6.10a}$$

$$\text{s.t.} \quad y(t_j; q_{j-1}, s_{j-1}) - s_j = 0, \quad j = 1, \dots, N^s \tag{6.10b}$$

$$c(q,s) \leq 0 \tag{6.10c}$$

For OED problems, the variables H_1 and H_2 in the coupled constraints (4.9b) and (4.9c) are also included in the loop. We note that Algorithm 8 does not require evaluation of constraint derivatives, which makes it very efficient. In the algorithm, acceptability to the filter is checked after integrating the states on one shooting interval. This is to ensure that the algorithm terminates with an iterate $x^{[k+1]}$ that is in some sense close to the current iterate $x^{[k]}$. Another possibility is to try to fulfill *all* continuity constraints and then check acceptability to the filter.

This restoration heuristic may fail to produce an acceptable iterate for two reasons: First, numerical integration of the states may not be possible over the whole interval for the given controls $q^{[k]}$, that means some continuity constraints cannot be satisfied. Second, by satisfying continuity conditions, other constraints (6.10c) may be violated such that the overall infeasibility of the new iterate is not sufficiently reduced. In both cases, we resort to the feasibility restoration phase described in the next section.

Algorithm 8: Feasibility restoration heuristic for problems arising in DMS.

Given: Iterate $x^{[k]} := (q_0^{[k]}, \ldots, q_{N^s-1}^{[k]}, s_0^{[k]}, \ldots, s_{N^s}^{[k]})$, current filter $\mathcal{F}^{[k+1]}$.
Set $q^{[k+1]} = q^{[k]}$, $s_0^{[k+1]} = s_0^{[k]}$.
For $l = 1, \ldots, N^s$

1. Try to set $s_l^{[k+1]} = y(t_l; q_{l-1}^{[k+1]}, s_{l-1}^{[k+1]})$ by integration.
   Set $x^{[k+1,l]} = (q^{[k+1]}, s_0^{[k+1]}, \ldots, s_l^{[k+1]}, s_{l+1}^{[k]}, \ldots, s_{N^s}^{[k]})$.
2. If integration fails, set $s_l^{[k+1]} = s_l^{[k]}$, $l = l+1$ and go to 1.
3. If $(\varphi(x^{[k+1,l]}), \eta(x^{[k+1,l]})) \notin \mathcal{F}^{[k+1]}$, stop with new iterate $x^{[k+1]} := x^{[k+1,l]}$.
   Otherwise, set $l = l+1$ and go to 1.

End for.

6.3.2. Minimizing constraint violation

If the restoration heuristic described by Algorithm 8 fails, we attempt to minimize the constraint violation by applying the SQP method to the following NLP:

$$\min_{x \in \mathbb{R}^n, s \in \mathbb{R}^m} \quad \tfrac{1}{2}\|s\|_2^2 + \tfrac{\zeta}{2}\|D^{[k]}(x - x^{[k]})\|_2^2 \tag{6.11a}$$

$$\text{s.t.} \quad c(x) - s = 0, \tag{6.11b}$$

$$b_\ell \le x \le b_u. \tag{6.11c}$$

Here, an m-dimensional vector of slack variables s is introduced that represents the violation of the m nonlinear constraints. A term is included in the objective function that penalizes the deviation from the current iterate $x^{[k]}$, where $\zeta > 0$ is the weighting parameter, and the scaling matrix $D^{[k]}$ is defined by

$$D^{[k]} = \operatorname{diag}(\min(1, 1/|x_1^{[k]}|), \ldots, \min(1, 1/|x_n^{[k]}|)).$$

The required derivatives of (6.11) can be computed straightforward using the derivatives of c: The constraint Jacobian and objective gradient with respect to x and s are given by

$$\left[\nabla c(x)^T,\ -\mathbb{I}\right] \qquad \text{and} \qquad \left[\left[(D^{[k]})^2(x - x^{[k]})\right]^T,\ s^T\right]^T,$$

respectively.

Problem (6.11) is of the form (6.1), therefore we apply Algorithm 6. In practice, the problem is not solved until a prescribed accuracy is achieved but after every iteration we check if the current iterate in the restoration problem is acceptable for the filter $\mathcal{F}^{[k+1]}$ of the original problem. We start the restoration phase with x initialized by the current iterate of the original problem $x^{[k]}$ and s initialized by $s = c(x^{[k]})$. Therefore, the initial point in the restoration phase is feasible and we can expect the first iterate to reduce $\frac{1}{2}\|s\|_2^2$.

The restoration phase terminates succesfully, if a point acceptable for the filter $\mathcal{F}^{[k+1]}$ is found. If Algorithm 6 for problem (6.11) itself tries to revert to a restoration phase, the SQP method is terminated. If the restoration problem is solved to optimality using the tolerances of the original problem before the regular method can be resumed, the SQP method terminates with the message that the problem seems locally infeasible.

Chapter 7.

Hessian approximations

The Hessian matrix in an SQP method has major influence on the speed of convergence. In Chapter 5 it is shown how positive definite Hessian approximations can prevent fast local convergence in case of nonconvexity of the underlying problem. This motivates the development of Algorithm 6 that allows indefinite Hessians. This chapter deals with specific Hessian approximations to be used with Algorithm 6 for NLPs arising in DMS. We first discuss possibilities how to choose a sequence of Hessians in every SQP iteration. An indefinite SR1 approximation is used whenever this is possible and a positive definite BFGS approximation is provided as fallback. We then give details on how to compute the SR1 and BFGS quasi-Newton approximations for block-diagonal Hessians. Finally, we show how to apply a sizing strategy to the block-BFGS approximations, addressing some of their shortcomings. Here, the approximate Hessian is multiplied by an appropriate scalar before updating that shifts its spectrum such that it overlaps with the spectrum of the exact Hessian.

7.1. Choice of the Hessian sequence

The loop in Step 2 of Algorithm 6 requires a sequence of Hessian approximations $B^{[k,l]}$, $l = 0, 1, \ldots, l_{\max}$ during SQP iteration k. On the one hand, we want to use indefinite matrices that permit the approximation of negative curvature in the constraint range space so that the issues pointed out in Section 5.3 can be prevented. On the other hand, we prefer an early termination of the loop to keep the number of QP iterations small.

In general, we consider a sequence of convex combinations of a "desired" Hessian $B_{\text{des}}^{[k]}$ and a positive definite approximation $B_{\text{pd}}^{[k]}$:

$$B^{[k,l]} = (1-\mu^{[l]})B_{\text{des}}^{[k]} + \mu^{[l]}B_{\text{pd}}^{[k]} \quad \text{with } 0 = \mu^{[0]} < \mu^{[1]} < \ldots < \mu^{[l_{\max}]} = 1.$$

This guarantees termination of the inner loop because a positive definite matrix certainly satisfies Assumption 6.1. Possible choices for B_{des} are the exact Hessian, $\nabla^2_{xx}\mathcal{L}(x^{[k]}, \lambda^{[k]})$, or an approximation by SR1 updates $B_{\text{SR1}}^{[k]}$. For the positive definite approximation $B_{\text{pd}}^{[k]}$ possibilities are a damped BFGS approximation, $B_{\text{BFGS}}^{[k]}$, or a scaled identity matrix $\sigma^{[k]}\mathbb{I}$.

Another strategy is to start with a possibly indefinite $B_{\text{des}}^{[k]}$ and add multiples of the identity:

$$B^{[k,l]} = B_{\text{des}}^{[k]} + \mu^{[l]}\mathbb{I} \quad \text{with } 0 = \mu^{[0]} < \mu^{[1]} < \mu^{[2]} < \ldots.$$

The interior point code IPOPT [199] and LOQO [195] use this scheme to correct the inertia of indefinite Hessians. A number of related convexification schemes have recently been presented by [96].

Parallelization of the inner loop
We note that these strategies permit a parallel implementation of the loop, that is, the problems QP($B^{[k,l]}$) can be solved simultaneously. To detail this process, let us drop the index k and simply write QP(l):=QP($B^{[k,l]}$). Furthermore, we define a function Ξ with $\Xi(l) \in \{-1,0,1\}$, where $\Xi(l) = -1$ means QP(l) has not yet terminated, $\Xi(l) = 0$ means QP(l) has terminated and Assumption (6.1) is violated, and $\Xi(l) = 1$ means that QP(l) has terminated and Assumption (6.1) is satisfied, i.e., a direction admissible for the line search has been found.

Suppose now that for some index $\bar{l}$ problem QP($\bar{l}$) has terminated. Then all running QP instances can be terminated immediately if one of the following conditions is met:

- Case 1, $\Xi(\bar{l}) = 0$: Let l^* be the smallest index with $\Xi(l^*) = 1$. If $\Xi(l) = 0$ for all $l < l^*$, start the line search with the solution produced by QP(l^*). If no index l^* with $\Xi(l^*) = 1$ exists, or if some QP(l) with $l < l^*$ has not yet terminated, continue.
- Case 2, $\Xi(\bar{l}) = 1$: If $\Xi(l) = 0$ for all $l < \bar{l}$, set $l^* = \bar{l}$ and start the line search with the solution produced by QP($\bar{l}$). If $\Xi(l) \neq 0$ for some $l < \bar{l}$, i.e., one of the QPs has not yet terminated, or there exists a successfully terminated QP that is closer to the desired Hessian B_{des}, continue.

All other solutions are then discarded, in particular solutions with $\Xi(l) = 1$, but $l > l^*$.

Hessian approximations for OED problems
Our main interest lies in approximating the Hessian of OED problems that are block-structured due to DMS. Because full exact Hessians for OED problems are usually not available, we strive to make use of a blockwise SR1 approximation $B^{[k]}_{\text{SR1}}$ whenever this is possible, and resort to a blockwise BFGS approximation $B^{[k]}_{\text{BFGS}}$ otherwise. As we discussed in Sec. 4.3.2, we can also incorporate partial information of the exact Hessian of the OED objective. Instead of approximating the diagonal block that corresponds to the variables for the Fisher matrix, $\nabla^2_{HH}\mathcal{L}(x^{[k]}, \lambda^{[k]})$, we can compute the exact Hessian for this block efficiently using the entries of the current covariance matrix C. As the OED objective is convex, this block is positive definite and thus compatible with Assumption 6.1.

7.2. Partitioned quasi-Newton updates

Quasi-Newton updates incorporate recently observed curvature information of the Lagrangian into the existing Hessian by a simple rank-one or rank-two correction. This is done by requiring that the approximation for the next iteration satisfies the *secant equation*

$$B^{[k+1]}\delta^{[k]} = \gamma^{[k]}, \tag{7.1}$$

where $\delta^{[k]}$ is the current step and $\gamma^{[k]}$ is the difference of the gradients of the Lagrangian:

$$\delta^{[k]} := x^{[k+1]} - x^{[k]}$$
$$\gamma^{[k]} := \nabla_x \mathcal{L}(x^{[k+1]}, \lambda^{[k+1]}) - \nabla_x \mathcal{L}(x^{[k]}, \lambda^{[k+1]}).$$

Many rank-two update formulae can be derived from the secant equation (7.1), see, e.g., [190] for an overview. However, it is generally agreed that the BFGS update is the most effective one in practice [154, 190] which is why we restrict our discussion of rank-two updates to the BFGS update. Note that there is only a single symmetric rank-one update satisfying the secant equation.

The block-diagonal structure of the true Hessian of the Lagrangian is maintained by means of a *partitioned* quasi-Newton update. Following the approach suggested in [34, 163], we approximate each block separately by a suitable update formula. This can be seen as a special case of partitioned quasi-Newton updates proposed in [110, 111] and leads to a high-rank update in each SQP iteration. The updates apply the full-space formulae to the appropriate subvectors of $\delta^{[k]}$ and $\gamma^{[k]}$. Likewise, the scaling and damping procedures described below are carried out for each block independently.

In the following, we refrain from introducing an additional index to indicate a block. Rather, the quantities $\delta^{[k]}$, $\gamma^{[k]}$, $B^{[k]}_{(\cdot)}$, $\theta^{[k]}$, and $\sigma^{[k]}_{(\cdot)}$ refer to any one of the diagonal blocks in the Hessian for the remainder of this section. Formulae should hence be read as being applied to each block individually, and to all blocks at the same time.

7.2.1. SR1 Update

Our method tries to use the SR1 update whenever possible. It has appealing theoretical properties and has been reported to be very efficient in practice, see below. In particular, it generates very good approximations to the exact Hessian and is thus able to adequately reflect negative curvature in the Lagrangian which leads to rapid local convergence. The SR1 update is defined by the formula

$$B^{[k+1]}_{\mathrm{SR1}} = B^{[k]}_{\mathrm{SR1}} + \frac{(\gamma^{[k]} - B^{[k]}_{\mathrm{SR1}}\delta^{[k]})(\gamma^{[k]} - B^{[k]}_{\mathrm{SR1}}\delta^{[k]})^T}{(\gamma^{[k]} - B^{[k]}_{\mathrm{SR1}}\delta^{[k]})^T\delta^{[k]}}, \quad k = 0, 1, 2, \ldots. \tag{7.2}$$

The choice of the initial matrix $B^{[0]}_{\mathrm{SR1}}$ is described Section 7.2.4. Note that the update can break down if the denominator $\gamma^{[k]} - B^{[k]}_{\mathrm{SR1}}\delta^{[k]}$ vanishes. In this case, there exists no symmetric rank-one correction to the current approximation that satisfies the secant equation. Following the rule in [154], we guard against denominators that are close to zero and skip the update whenever

$$\left|(\gamma^{[k]} - B^{[k]}_{\mathrm{SR1}}\delta^{[k]})^T\delta^{[k]}\right| < \varepsilon_{\mathrm{SR1}} \cdot \|\gamma^{[k]} - B^{[k]}_{\mathrm{SR1}}\delta^{[k]}\|_2 \cdot \|\delta^{[k]}\|_2.$$

In our implementation, we choose $\varepsilon_{\mathrm{SR1}} = 10^{-8}$.

Theoretical properties

The SR1 update was first derived in the first paper on quasi-Newton methods [54] by Davidon in 1959 but has later been re-discovered several times. The fact that the denominator in (7.2) can vanish and that positive definiteness is not preserved has often been cited as a disadvantage and rank-two updates such as BFGS are often used instead. However, with the advent of trust-region methods, the SR1 update gained popularity, see [40, 46, 129, 147]. In particular, strong theoretical properties indicate that it can outperform the BFGS update in some situations. In the following, we state two important results that motivate the choice of the SR1 update to approximate nonconvex Hessians. The first well-known result has first been proven by [71].

Theorem 7.1 (Quadratic termination of SR1)**.** *Consider n SR1 updates using difference vectors $\delta^{[k]}$ and $\gamma^{[k]}$ for $k = 0, 1, 2, \ldots, n-1$, where $\gamma^{[k]} = H\delta^{[k]}$ and H is symmetric. If $B_{\text{SR1}}^{[0]}$ is symmetric, and if for $k = 0, 1, \ldots, n-1$ the denominators in* (7.2) *are non-zero, and the vectors $\delta^{[k]}$ are linearly independent, then $B_{\text{SR1}}^{[n]} = H$.*

Proof. We show by induction that

$$B_{\text{SR1}}^{[k]}\delta^{[j]} = \gamma^{[j]}, \quad j = 0, \ldots, k-1, \tag{7.3}$$

where $0 \leq k \leq n$. For $k = 0$, the condition is vacuous and hence true. Assume now that (7.3) is true for some k, such that $0 \leq k \leq n-1$. The definition of $B_{\text{SR1}}^{[k+1]}$ gives

$$B_{\text{SR1}}^{[k+1]}\delta^{[j]} = B_{\text{SR1}}^{[k]}\delta^{[j]} + \frac{r^{[k]}{r^{[k]}}^T\delta^{[j]}}{{r^{[k]}}^T\delta^{[k]}}, \tag{7.4}$$

where $r^{[k]} := \gamma^{[k]} - B_{\text{SR1}}^{[k]}\delta^{[k]}$. For $j = k$ the right-hand side is $B_{\text{SR1}}^{[k]}\delta^{[k]} + r^{[k]}$ which is equal to $\gamma^{[k]}$ by definition of $r^{[k]}$. For $j < k$ it follows from (7.3) that $B_{\text{SR1}}^{[k]}\delta^{[j]} = \gamma^{[j]}$, and, using symmetry of $B_{\text{SR1}}^{[k]}$, that

$${r^{[k]}}^T\delta^{[j]} = (\gamma^{[k]} - B_{\text{SR1}}^{[k]}\delta^{[k]})^T\delta^{[j]} = {\gamma^{[k]}}^T\delta^{[j]} - {\delta^{[k]}}^T B_{\text{SR1}}^{[k]}\delta^{[j]} = {\gamma^{[k]}}^T\delta^{[j]} - {\delta^{[k]}}^T\gamma^{[j]}$$

Because $\gamma^{[j]} = H\delta^{[j]}$ for all $j = 0, 1, \ldots, n-1$, it follows for $j < k$ that ${r^{[k]}}^T\delta^{[j]} = 0$. Thus for both $j = k$ and $j < k$ it has been shown that $B_{\text{SR1}}^{[k]}\delta^{[j]} = \gamma^{[j]}$ and hence (7.3) has been established for $k+1$ replacing k. Hence by induction, (7.3) is true for all $k = 0, \ldots, n$. For $k = n$, and using $\gamma^{[j]} = H\delta^{[j]}$, (7.3) can be written as

$$B_{\text{SR1}}^{[n]}\delta^{[j]} = H\delta^{[j]}, \quad j = 0, \ldots, n-1$$

or as

$$B_{\text{SR1}}^{[n]}\Delta = H\Delta$$

where Δ is an $n \times n$ matrix with columns $\delta^{[j]}$, $j = 0, 1, \ldots, n-1$. But Δ is nonsingular by the linear independence assumption, so it follows that $B_{\text{SR1}}^{[n]} = H$. □

A consequence of this theorem is that if SR1 updates are used to minimize a convex quadratic function, then, in exact arithmetics, iteration n is a Newton iteration which will locate the minimizer exactly. A key feature of the proof is the establishment of the so-called hereditary property (7.3), in which secant conditions from previous iterations remain satisfied by subsequent $B_{\mathrm{SR1}}^{[k]}$ matrices.

A number of results concerning convergence of matrices generated by the SR1 update is established in [46]. In particular, the following theorem is of interest.

Theorem 7.2 (Convergence of SR1 approximations)**.** *Consider a quasi-Newton method using SR1 updates that generates a sequence of iterates $\{x^{[k]}\}$ for the unconstrained minimization problem*

$$\min_{x\in\mathbb{R}^n} \quad \varphi(x).$$

Assume that $\varphi \in \mathcal{C}^2$, its Hessian $\nabla^2\varphi$ is Lipschitz continuous, and that $\{x_k\}$ converges to a finite limit point x^*. Assume further that the sequence of steps $\{\delta^{[k]}\}$ is uniformly linearly independent, that is, there exist k_0 and $m \geq n$ such that, for every $k \geq k_0$ one can choose n distinct indices $k \leq k_1 < \cdots < k_n \leq k+m$ such that the minimum singular value of the $n \times n$ matrix

$$\begin{bmatrix} \frac{\delta^{[k_1]}}{\|\delta^{[k_1]}\|} & \cdots & \frac{\delta^{[k_n]}}{\|\delta^{[k_n]}\|} \end{bmatrix}$$

is bounded away from zero. Finally, assume that the denominators in (7.2) *are all nonzero.*

Then there exists a constant $C > 0$ such that, for $k \geq k_0$,

$$\left\| B_{\mathrm{SR1}}^{[k+m+1]} - \nabla^2\varphi(x^*) \right\| \leq C\varepsilon^{[k]} \tag{7.5}$$

where

$$\varepsilon^{[k]} = \max\{ \left\| x^{[l]} - x^* \right\| \mid k \leq l \leq k+m+1 \} \tag{7.6}$$

and

$$\lim_{k\to\infty} \left\| B_{\mathrm{SR1}}^{[k]} - \nabla^2\varphi(x^*) \right\| = 0. \tag{7.7}$$

This theorem implies that the sequence of matrices $\{B_{\mathrm{SR1}}^{[k]}\}$ converges to the true Hessian $\nabla^2\varphi(x^*)$. Furthermore, the estimate (7.5) provides some indication of the speed of convergence. Through further analysis, one can show that (7.5) can be replaced by

$$\left\| B_{\mathrm{SR1}}^{[k+m+1]} - \nabla^2\varphi(x^*) \right\| \leq C_2 \left\| x^{[k]} - x^* \right\| \tag{7.8}$$

under some assumptions. This means that fast convergence of $\{x^{[k]}\}$ to x^* implies fast convergence of $\{B_{\mathrm{SR1}}^{[k]}\}$ to $\nabla^2\varphi(x^*)$.

7.2.2. Damped BFGS Update

Starting from an initial approximation $B_{\text{BFGS}}^{[0]}$, the BFGS update is given by the formula

$$B_{\text{BFGS}}^{[k+1]} = B_{\text{BFGS}}^{[k]} - \frac{B_{\text{BFGS}}^{[k]}\delta^{[k]}\delta^{[k]^T}B_{\text{BFGS}}^{[k]}}{\delta^{[k]^T}B_{\text{BFGS}}^{[k]}\delta^{[k]}} + \frac{\gamma^{[k]}\gamma^{[k]^T}}{\gamma^{[k]^T}\delta^{[k]}}, \quad k = 0, 1, \ldots. \tag{7.9}$$

Possible choices for $B^{[0]}$ are discussed below in Sec. 7.2.4.

The BFGS update maintains positive definiteness whenever the *curvature condition*, $\delta^{[k]^T}\gamma^{[k]} > 0$ is satisfied. While in unconstrained optimization this is the case in a neighborhood of a solution, it is not true for constrained problems as pointed out in Section 5.3. Therefore, we use Powell's damping strategy [165] that makes it possible to perform the BFGS update even in a direction of negative curvature. We compute damping parameters $\theta^{[k]}$ from

$$\theta^{[k]} = \begin{cases} \dfrac{0.8\delta^{[k]^T}B_{\text{BFGS}}^{[k]}\delta^{[k]}}{\delta^{[k]^T}B_{\text{BFGS}}^{[k]}\delta^{[k]} - \delta^{[k]^T}\gamma^{[k]}}, & \text{if } \delta^{[k]^T}\gamma^{[k]} < 0.2\delta^{[k]^T}B_{\text{BFGS}}^{[k]}\delta^{[k]}, \\ 1, & \text{else.} \end{cases}$$

With this, we define $\overline{\gamma}^{[k]}$ by

$$\overline{\gamma}^{[k]} = \theta^{[k]}\gamma^{[k]} + (1-\theta^{[k]})B_{\text{BFGS}}^{[k]}\delta^{[k]},$$

and use $\overline{\gamma}^{[k]}$ in place of $\gamma^{[k]}$ to compute the damped update from Equation (7.9). Note that $\theta^{[k]} = 1$ gives the unmodified BFGS update. We also note that the secant condition (7.1) does not hold anymore when the damping strategy is applied. It does hold for the modified vector $\overline{\gamma}^{[k]}$ instead.

An additional safeguard is required because the update described above is applied blockwise, i.e., $\delta^{[k]}$ is in fact only a subvector of the full-space step. This implies that $\delta^{[k]}$ and hence the denominators $\delta^{[k]^T}H^{[k]}\delta^{[k]}$ and $\gamma^{[k]^T}\delta^{[k]}$ can be zero even though the full-space step is not. We skip the update for the current block whenever $\delta^{[k]^T}H^{[k]}\delta^{[k]} < \varepsilon_{\text{QN}}$ or $\gamma^{[k]^T}\delta^{[k]} < \varepsilon_{\text{QN}}$. In our implementation, we set $\varepsilon_{\text{QN}} = 10^{-14}$.

Theoretical properties

Although the BFGS update is universally considered to be the best quasi-Newton update in practice [154, 190], it lacks the strong theoretical background of the SR1 update, and convergence results for the BFGS update typically require much stronger assumptions. In particular, most existing results are stated under the assumption that exact line searches are used. An analogon to Thm 7.1 is given in the following theorem, see [55].

Theorem 7.3 (Quadratic termination of BFGS)**.** *Consider a quasi-Newton method using BFGS updates with symmetric positive definite initial matrix $B_{\text{BFGS}}^{[0]}$ that generates a sequence of iterates $\{x^{[k]}\}$ for the unconstrained quadratic minimization problem*

$$\min_{x\in\mathbb{R}^n} \quad \varphi(x) := \tfrac{1}{2}x^T Hx + g^T x,$$

where H is assumed to be positive definite. Furthermore, assume that an exact line search is used to produce the iterates $\{x^{[k]}\}$, that is, step lengths $\alpha^{[k]}$ are chosen such that they minimize the function along the search directions $-{B^{[k]}_{\mathrm{BFGS}}}^{-1}\nabla\varphi(x^{[k]})$. Then there is an integer $0 \leq l \leq n$ such that $x^{[l]} = x^* = -H^{-1}g$ and if $l = n$, $B^{[k]}_{\mathrm{BFGS}} = H$.

Compare this to the result of Thm 7.1, where the steps $\delta^{[k]}$ can be defined in an almost arbitrary manner. If step lengths $\alpha = 1$ are used, it can be shown that the BFGS matrices need *not* converge to the exact Hessian, and examples for this undesirable behavior can be found even in the strictly convex quadratic case [86].

7.2.3. Limited-memory storage and build-up

In practice, a *limited-memory* update is often used instead of a full memory update. That means old curvature information is consistently discarded and only the M most recent updates are taken into account, where M is relatively small, say, $M \leq 20$. We noticed in our experiments that the full-memory BFGS updates often lead to more SQP iterations than their limited-memory counterparts, although theoretically the rate of convergence is only linear [145] in the limited memory case. We attribute this to the fact that "old" curvature information can result in poor Hessian approximations and prevent fast progress of the algorithm. On the other hand, the constant M should not be chosen too low for SR1 updating because then the available secant information may not be enough for the SR1 accuracy properties to manifest, see [147].

Usually, limited-memory methods avoid the explicit construction of the Hessian by making use of compact representation schemes, that allow efficient computation of matrix–vector products [41]. However, the `qpOASES` QP solver that is used in our implementation requires that the elements of the Hessian matrix are provided explicitly. As a consequence, we compute the dense quasi-Newton matrices for each block. More specifically, we store the M most recent vectors

$$\{\delta^{[k-1]}, \ldots, \delta^{[k-M]}\},\ \{\gamma^{[k-1]}, \ldots, \gamma^{[k-M]}\},$$

where $M = \min(\tilde{M}, k)$ and $\tilde{M}$ is a fixed number. Starting with an initial approximation B^{k-M}, we explicitly construct the blocks of the matrix $B^{[k]}$ by forming the sum of the M most recent updates. This approach is efficient because the blocks are relatively small, and the computational cost for constructing the dense matrices is negligible compared to evaluation of sensitivities and solution of the QP.

7.2.4. Initial approximation

For the limited-memory updates, we choose the initial matrix $B^{[k-M]}$ as the identity matrix multiplied by a scaling factor $\sigma^{[k]}$. In the first SQP iteration, we set $\sigma^{[0]} = 1$. After this, $\sigma^{[k]}$

is one of the following quantities:

$$\sigma_{\mathrm{SP}}^{[k]} = \frac{{\gamma^{[k]}}^T \gamma^{[k]}}{{\gamma^{[k]}}^T \delta^{[k]}}, \qquad \sigma_{\mathrm{Mean}}^{[k]} = \sqrt{\frac{{\gamma^{[k]}}^T \gamma^{[k]}}{{\delta^{[k]}}^T \delta^{[k]}}}, \qquad \sigma_{\mathrm{OL}}^{[k]} = \frac{{\gamma^{[k]}}^T \delta^{[k]}}{{\delta^{[k]}}^T \delta^{[k]}}. \tag{7.10}$$

Note that the initial matrix $B^{[0]}$ for the full-memory update is multiplied by one of the above after the first step. The factor σ_{SP} is due to Shanno and Phua [185], the factor σ_{OL} is due to Oren and Luenberger [157], and σ_{Mean} is the geometric mean of σ_{SP} and σ_{OL}. Note that $\sigma_{\mathrm{SP}} \geq \sigma_{\mathrm{Mean}} \geq \sigma_{\mathrm{OL}}$. To avoid numerical instabilities we enforce $\sigma_{(\cdot)} > \varepsilon_\sigma$ whenever the denominator becomes too small. We set $\varepsilon_\sigma = 10^{-14}$ in our implementation.

7.3. Sizing of quasi-Newton updates

Aside from scaling the initial approximation $B^{[0]}$, *self-scaling variable metric (SSVM) methods*, first developed in [156, 157, 158], try to compensate for poor scaling by multiplying the approximation $B^{[k]}$ with a scalar $\sigma^{[k]}$ before performing the update. For example, a self-scaling BFGS update is computed by the formula

$$B_{\mathrm{BFGS}}^{[k+1]} = \sigma^{[k]} \left[B_{\mathrm{BFGS}}^{[k]} - \frac{B_{\mathrm{BFGS}}^{[k]} \delta^{[k]} {\delta^{[k]}}^T B_{\mathrm{BFGS}}^{[k]}}{{\delta^{[k]}}^T B_{\mathrm{BFGS}}^{[k]} \delta^{[k]}} \right] + \frac{\gamma^{[k]} {\gamma^{[k]}}^T}{{\gamma^{[k]}}^T \delta^{[k]}},$$

where $\sigma^{[k]}$ is given by, e.g.,

$$\sigma^{[k]} = \frac{{\gamma^{[k]}}^T \delta^{[k]}}{{\delta^{[k]}}^T B^{[k]} \delta^{[k]}},$$

cf. Eq. (7.10).

Although theoretical and numerical evidence suggest that the SSVM philosophy of scaling in every iteration is inferior to scaling only at the first iteration, see [155, 185], the concept has been refined and successfully applied by various authors [2, 3, 50, 206]. We note that the literature on the topic is almost exclusively concerned with unconstrained optimization.

Our motivation for using scaling techniques for quasi-Newton approximations is that numerical experiments with BFGS updates indicate that the approximations tend to accumulate many large eigenvalues. This causes the algorithm to take unnecessarily small steps and prevents progress towards a solution. A remedy is provided by the concept of *selective sizing* as proposed by Contreras and Tapia [50] that in turn was inspired by the SSVM methods. In this section, we will first provide some theoretical motivation for the selective sizing strategy. The notation using Rayleigh quotients has been first developed in a Bachelor's thesis [144] that was supervised within this project. Afterwards we describe the selective sizing strategy that we implement in our method.

7.3.1. Theoretical motivation

By *sizing* we mean multiplying the approximate Hessian by an appropriate scalar before it is updated. The term sizing, as opposed to scaling, was used in [56] to emphasize the fact that multiplying by a scalar shifts the spectrum of the matrix. The goal of sizing is to shift the eigenvalues of the Hessian approximation such that they better approximate those of the exact Hessian. We now define precisely what we mean by this.

Definition 7.4 (Convex spectrum)**.** By the *convex spectrum* of a collection of $n \times n$ matrices $H_1, \ldots, H_m$, denoted $\operatorname{conspec}(H_1, \ldots, H_m)$, we mean the convex hull of the eigenvalues of $H_1, \ldots, H_m$.

When H is symmetric, all eigenvalues are real and the convex spectrum is an interval. A representation of the convex spectrum of a matrix H is given by the Rayleigh quotient.

Definition 7.5 (Rayleigh quotient)**.** For a symmetric matrix $H \in \mathbb{R}^{n\times n}$, the *Rayleigh quotient* is defined as

$$R_H(x) := \frac{x^T A x}{x^T x}, \quad x \neq 0.$$

A well-known property of the Rayleigh quotient is

$$\lambda_{\min} \leq R_H \leq \lambda_{\max},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalue of H, respectively. Let us now define the notion of sizing for Hessian matrices.

Definition 7.6 (Sizing)**.** Consider a $\mathcal{C}^2$ function $\phi : \mathbb{R}^n \to \mathbb{R}$. We say that the scalar σ *sizes* $B \in \mathbb{R}^{n\times n}$ relative to the Hessian of ϕ if there exists $x_1, x_2, \ldots, x_m \in \mathbb{R}^m$ such that

$$\operatorname{conspec}(\sigma B) \cap \operatorname{conspec}(\nabla^2\phi(x_1), \nabla^2\phi(x_2), \ldots, \nabla^2\phi(x_m)) \neq \emptyset.$$

We call the integer m the degree of the sizing factor σ.

We are interested in a sizing factor σ that sizes a BFGS approximation relative to the exact Hessian of the Lagrangian. Note that σ is defined in terms of evaluation of the exact Hessian—which is not accessible—at several points. Our only information about the exact Hessian comes from the vector $\gamma^{[k]}$ that satisfies the secant equation

$$B^{[k+1]}\delta^{[k]} = \gamma^{[k]}.$$

The idea is to use available information $\delta^{[k-i]}$ and $\gamma^{[k-i]}$ from previous iterations $i = 1, \ldots, m$ to construct sizing factors of degree m.

Theorem 7.7. *Consider a $\mathcal{C}^2$ function $\phi : \mathbb{R}^n \to \mathbb{R}$ and a sequence of matrices $\{B^{[k]}\}$ constructed by the BFGS update to approximate its Hessian. Then σ defined by*

$$\sigma \cdot \beta^{[k]} = \sum_{i=1}^{m} \alpha_i \cdot \frac{\delta^{[k-i+1]^T} \gamma^{[k-i+1]}}{\delta^{[k-i+1]^T} \delta^{[k-i+1]}}, \tag{7.11}$$

where

$$\alpha_i > 0, \quad \sum_{i=1}^{m} \alpha_i = 1 \quad \textit{and} \quad \beta^{[k]} \in \operatorname{conspec}(B^{[k]}).$$

constitutes a sizing factor of degree m, that sizes the current Hessian approximation $B^{[k]}$ relative to the exact Hessian.

Proof. Consider the function $\psi : \mathbb{R}^n \to \mathbb{R}$ defined by:

$$\psi(x) := \delta^{[k]^T} \nabla \mathcal{L}(x, \lambda^{[k+1]}).$$

By definition of $\gamma^{[k]}$ we know that

$$\delta^{[k]^T} \gamma^{[k]} = \psi(x^{[k+1]}) - \psi(x^{[k]})$$

We apply the mean-value theorem to obtain

$$\begin{aligned} \psi(x^{[k+1]}) - \psi(x^{[k]}) &= \nabla \psi(x^{[k]} + \theta(x^{[k+1]} - x^{[k]}))^T \delta^{[k]} \\ &= \delta^{[k]^T} \nabla^2 \mathcal{L}(x^{[k]} + \theta \delta^{[k]}, \lambda^{[k+1]}) \delta^{[k]}, \quad \theta \in [0,1]. \end{aligned}$$

This means that the Rayleigh quotient

$$R_{\nabla^2 \mathcal{L}(x^{[k]} + \theta \delta^{[k]}, \lambda^{[k+1]})}(\delta^{[k]}) = \frac{\delta^{[k]^T} \gamma^{[k]}}{\delta^{[k]^T} \delta^{[k]}}$$

is a point in the convex spectrum of the exact Hessian of $\mathcal{L}$ at a point $x^{[k]} + \theta \delta^{[k]}$. □

The scalar $\beta^{[k]}$ can be obtained by evaluating the Rayleigh quotient of $B^{[k]}$ at a suitable point. For example, σ_{OL} is a sizing factor of degree 1:

$$\sigma_{\mathrm{OL}}^{[k]} = \frac{\delta^{[k]^T} \gamma^{[k]}}{\delta^{[k]^T} B^{[k]} \delta^{[k]}}.$$

Centered Oren–Luenberger sizing factor

A sizing factor of degree 2 is the centered Oren–Luenberger sizing factor σ_{COL} due to [50]. It uses two pieces of approximate Hessian information and can be derived as follows. Using formula (7.11), we require that

$$\sigma_{\mathrm{COL}}^{[k]}(\alpha) \cdot \left[(1-\alpha) R_{B^{[k]}}(\delta^{[k-1]}) + \alpha R_{B^{[k]}}(\delta^{[k]}) \right] = (1-\alpha) R_{\nabla^2 \mathcal{L}}(\delta^{[k-1]}) + \alpha R_{\nabla^2 \mathcal{L}}(\delta^{[k]}),$$

where we use the compact notation $R_{\nabla^2 \mathcal{L}}(\delta^{[k]}) = R_{\nabla^2 \mathcal{L}(x^{[k]} + \theta \delta^{[k]}, \lambda^{[k+1]})}(\delta^{[k]})$. Here, we use a convex combination of $R_{B^{[k]}}(\delta^{[k-1]})$ and $R_{B^{[k]}}(\delta^{[k]})$ to obtain $\beta^{[k]}$ in formula (7.11). Observing that $B^{[k]} \delta^{[k-1]} = \gamma^{[k-1]}$, this yields

Definition 7.8 (COL sizing factor)**.** The *centered Oren–Luenberger (COL) sizing factor* at iteration k for a parameter $\alpha \in (0,1)$ is defined as

$$\sigma_{\mathrm{COL}}^{[k]}(\alpha) = \frac{(1-\alpha)\gamma^{[k-1]^T}\delta^{[k-1]}/\delta^{[k-1]^T}\delta^{[k-1]} + \alpha\gamma^{[k]^T}\delta^{[k]}/\delta^{[k]^T}\delta^{[k]}}{(1-\alpha)\gamma^{[k-1]^T}\delta^{[k-1]}/\delta^{[k-1]^T}\delta^{[k-1]} + \alpha\delta^{[k]^T}B^{[k]}\delta^{[k]}/\delta^{[k]^T}\delta^{[k]}}. \tag{7.12}$$

Note that $\alpha = 1$ yields the Oren-Luenberger factor $\sigma_{\mathrm{OL}}^{[k]}$. Similarly, sizing factors of higher degree can be conceived, see [144].

7.3.2. Selective sizing strategy

Recall that our initial goal was to prevent the accumulation of large eigenvalues in quasi-Newton approximations that may lead to unnecessarily small steps. This is accomplished by the *selective sizing strategy*, that has been first described in [50] for unconstrained problems. It uses the COL sizing factor (7.12) whenever it is smaller than 1, thus effectively reducing the eigenvalues of $B^{[k]}$.

More precisely, we choose $\alpha^{[k]} = \min(\frac{1}{2}, \tau_{\mathrm{COL}}\|\delta^{[k]}\|_\infty)$ and size $B^{[k]}$ before updating by $\sigma_{\mathrm{COL}}^{[k]}(\alpha^{[k]})$ whenever $0 < \varepsilon_{\mathrm{COL}} < \sigma_{\mathrm{COL}}^{[k]}(\alpha^{[k]}) < 1$. Here, $\varepsilon_{\mathrm{COL}}$ is a safeguard to prevent near-singular approximations. Moreover the constant $\tau_{\mathrm{COL}} > 0$ guarantees that $\alpha^{[k]} \to 0$ and thus $\sigma_{\mathrm{COL}}^{[k]} \to 1$ as the solution is approached. In our implementation, we choose $\tau_{\mathrm{COL}} = 10^4$ and $\varepsilon_{\mathrm{COL}} = 0.1$. Note that we replace $\gamma^{[k]}$ by $\bar{\gamma}^{[k]}$ in (7.12) whenever BFGS damping is active.

The computational overhead introduced by this strategy is very small: For each block of the Hessian, we only have to store the scalars $\delta^{[k-1]^T}\delta^{[k-1]}$ and $\gamma^{[k-1]^T}\delta^{[k-1]}$ from the previous iteration and perform one additional multiplication.

The selective sizing strategy performed very well in our numerical tests and was consistently better than the traditional strategy of sizing in only the first iteration and the SSVM philosophy of sizing in every iteration with a sizing factor of degree 1, see Chapter 10. Thus the question arises if more can be gained by using sizing factors of higher degree. Strategies based on factors of degree 3 are examined in [144]. However, they offer no significant improvement over the selective sizing strategy based on the COL sizing factor which is why we do not discuss them here. We attribute this to the fact that "too old" information $\delta^{[k-i]}$, $\gamma^{[k-i]}$ are less useful for approximating the current Hessian.

Chapter 8.

Solution of sparse and nonconvex quadratic programs

This chapter deals with the efficient solution of the sparse and nonconvex quadratic subproblems arising in Algorithm 6. The method we use is based on the parametric active set method [21, 68] implemented in `qpOASES` [69, 70]. We first outline the parametric quadratic programming algorithm before describing new developments that enable the solution of sparse and nonconvex quadratic programs, including a Schur complement approach, a mechanism for monitoring the inertia to deal with nonconvexity, and some practical issues. In particular, we show how the method can be used to efficiently check positive definiteness on $A_{\mathcal{S}}$, which is required by Algorithm 6 in every SQP iteration. These new developments for sparse and nonconvex problems were first described in the paper [125].

8.1. A parametric active set method

In this section we first establish some basic notation for (parametric) quadratic programming and review the optimality conditions for quadratic programs. Then we state the parametric quadratic programming algorithm, the basis for the new developments that enable us to efficiently solve the sparse nonconvex subproblems in our SQP method. Most of the content is taken from [70], but the notation has been adjusted to our SQP framework.

8.1.1. Optimality conditions

We are interested in solving quadratic programs of the following form:

$$\min_{d\in\mathbb{R}^n} \tfrac{1}{2} d^T B d + g^T d \tag{8.1a}$$

$$\text{s.t. } Ad + c = 0, \tag{8.1b}$$

$$b_\ell \leq d \leq b_u. \tag{8.1c}$$

The first-order necessary optimality conditions for (8.1) are a special case of the KKT conditions (1.2) and state that if d^* is a solution of (8.1), there exists a vector of Lagrange

multipliers or dual variables $\lambda^* \in \mathbb{R}^{m+n}$ such that

$$Bd^* + g - \begin{bmatrix} A^T & \mathbb{I}_n \end{bmatrix} \lambda^* = 0, \tag{8.2a}$$
$$Ad^* + c = 0, \tag{8.2b}$$
$$b_\ell \leq d^* \leq b_u, \tag{8.2c}$$
$$\lambda^*_{m+i} \geq 0, \text{ if } d^*_i = b_{\ell,i}, \tag{8.2d}$$
$$\lambda^*_{m+i} \leq 0, \text{ if } d^*_i = b_{u,i}, \tag{8.2e}$$
$$\lambda^*_{m+i} = 0, \text{ if } b_{\ell,i} < d^*_i < b_{u,i}. \tag{8.2f}$$

A pair (d^*, λ^*) that satisfies (8.2) is called a KKT point. In the convex case this is a global solution of (8.1). Let us further define the set of active variable bounds as a subset of $\bar{n} := \{1, \ldots, n\}$, $\mathcal{A} := \{i \in \bar{n} \mid d^*_i = b_{\ell,i} \text{ or } d^*_i = b_{u,i}\}$ and let $\mathbb{I}_\mathcal{A}$ be the matrix whose columns are the unit vectors $e_i \in \mathbb{R}^n$ with $i \in \mathcal{A}$. We define the matrix of active constraints as

$$A_\mathcal{A} := \begin{bmatrix} A^T & \mathbb{I}_\mathcal{A} \end{bmatrix}^T. \tag{8.3}$$

Second-order necessary optimality conditions in active set form can then be stated as follows:

Theorem 8.1 (QP second-order necessary conditions)**.** *Suppose $(d^*, \lambda^*) \in \mathbb{R}^{n+m+n}$ is a KKT point and a local minimizer of the QP* (6.2)*. Let $A_\mathcal{A}$ have full row rank and let the columns of the matrix $Z_\mathcal{A}$ form a basis of the null space of $A_\mathcal{A}$. Then the reduced Hessian $Z_\mathcal{A}^T B Z_\mathcal{A}$ is positive semidefinite.*

The following second order sufficient condition can be used to verify local optimality for (d^*, λ^*): If strict complementarity is satisfied (see Def. 1.6) and the reduced Hessian is positive definite, then (d^*, λ^*) is a local minimizer of (8.1).

8.1.2. The parametric quadratic programming paradigm

The idea behind parametric active set methods is to trace optimal solutions on an affine-linear homotopy path between two QP instances, parameterized by $\tau \in [0, 1]$. This yields a one-parametric family of QPs with affine-linear functions $g(\tau)$, $c(\tau)$, $b_\ell(\tau)$, and $b_u(\tau)$:

$$\min_{d \in \mathbb{R}^n} \tfrac{1}{2} d^T B d + g(\tau)^T d \tag{8.4a}$$
$$\text{s.t. } Ad + c(\tau) = 0, \tag{8.4b}$$
$$b_\ell(\tau) \leq d \leq b_u(\tau). \tag{8.4c}$$

For fixed τ, denote an optimal primal-dual solution by $(d(\tau), \lambda(\tau))$. It can be shown that optimal solutions depend piecewise affine-linearly but not necessarily continuously on τ, see [21]. The active set is constant on each linear segment. Parametric active set algorithms follow $(d(\tau), \lambda(\tau))$ by moving from the beginning of one segment to the next. A working set $\mathcal{W}(\tau)$ of active variables is maintained and is updated accordingly.

In our case, every iteration l of the loop in Step 2 of the SQP Algorithm 6 requires the solution of QP($B^{[k,l]}$). The parametric QP we want to solve reads as

$$\min_d \ \tfrac{1}{2} d^T B^{[k,l]} d + g(\tau)^T d \tag{8.5a}$$
$$\text{s.t. } A^{[k]} d + c(\tau) = 0, \tag{8.5b}$$
$$b_\ell(\tau) \le d \le b_u(\tau). \tag{8.5c}$$

The lower and upper bounds $b_\ell(\tau)$ and $b_u(\tau)$, the gradient vector $g(\tau)$ and the constraint vector $c(\tau)$ are the following affine-linear functions parameterized by $\tau \in [0,1]$:

$$b_\ell(\tau) = b_\ell - x^{[k-1]}(1-\tau) - x^{[k]}\tau, \qquad b_u(\tau) = b_u - x^{[k-1]}(1-\tau) - x^{[k]}\tau,$$
$$g(\tau) = \tilde{g}^{[k-1]}(1-\tau) + g^{[k]}\tau, \qquad c(\tau) = \tilde{c}^{[k-1]}(1-\tau) + c^{[k]}\tau.$$

The parametric data in $\tau = 0$ is initialized with

$$\tilde{g}^{[k-1]} = g^{[k-1]} + (B^{[k-1]} - B^{[k,l]})d^{[k-1]} - (A^{[k-1]} - A^{[k]})^T \bar{\lambda}^{[k-1]},$$
$$\tilde{c}^{[k-1]} = c^{[k-1]} + (A^{[k-1]} - A^{[k]})d^{[k-1]}.$$

This makes is possible to warm-start the solver in $\tau = 0$ from the known optimal solution $(d^{[k-1]}, \bar{\lambda}^{[k-1]})$ of the previous SQP iteration. The solution of (8.4) in $\tau = 1$ is the solution of QP($B^{[k,l]}$) sought for.

8.1.3. The algorithm

For the parametric quadratic programming algorithm we compute iterates $w^{[\nu]} = (d^{[\nu]}, \lambda^{[\nu]})$, and $\tau^{[\nu]}$, $\nu = 0, 1, 2, \ldots$, with $0 = \tau^{[0]} < \tau^{[1]} < \cdots < \tau^{[N]} = 1$ until a solution of (8.4) is found. In particular, conditions 8.1 are satisfied at a solution. We maintain a working set of variables that are active at one of their bounds:

$$\mathcal{W}^{[\nu]} \subseteq \{ i \in \bar{n} \mid d_i^{[\nu]} = b_{\ell,i}(\tau^{[\nu]}) \text{ or } d_i^{[\nu]} = b_{u,i}(\tau^{[\nu]}) \}.$$

We denote the complement of $\mathcal{W}^{[\nu]}$ in $\bar{n}$ by $(\mathcal{W}^{[\nu]})^{\complement}$. The number of elements in $\mathcal{W}^{[\nu]}$ is denoted by $\left|\mathcal{W}^{[\nu]}\right| \le n$. For the remainder of this section, we assume that the Hessian B is positive semidefinite. The nonconvex case is discussed in Sec. 8.3 below. Algorithm 9 shows the main steps of the parametric quadratic programming algorithm. We now explain several steps in more detail.

Computing the step direction (Step 2)
We denote by $\mathbb{I}_{\mathcal{W}^{[\nu]}} \in \mathbb{R}^{|\mathcal{W}^{[\nu]}| \times n}$ the identity matrix with the rows deleted whose indices are not in $\mathcal{W}^{[\nu]}$. We define

$$A_{\mathcal{W}^{[\nu]}} := \begin{bmatrix} A \\ \mathbb{I}_{\mathcal{W}^{[\nu]}} \end{bmatrix},$$

Algorithm 9: Parametric quadratic programming algorithm for positive semi-definite Hessians.

1. Start with an optimal primal–dual solution $w^{[0]} = (d^{[0]}, \lambda^{[0]})$ and associated working set $\mathcal{W}^{[0]}$ of the previously solved QP. Set $\tau^{[0]} = 0$, $\nu = 0$.
2. Determine the step direction $\Delta w^{[\nu]} = (\Delta d^{[\nu]}, \Delta\lambda^{[\nu]})$ using the current working set $\mathcal{W}^{[\nu]}$.
3. Determine the maximum homotopy step length $\Delta\tau^{[\nu]}$ and possibly the index of a blocking bound, p, or a blocking multiplier sign change, q, with the ratio tests

$$\Delta\tau_{\mathrm{p}} = \min_{i\in\mathcal{W}^{\complement}} \alpha_i^{\mathrm{p}}, \quad \text{where } \alpha_i^{\mathrm{p}} = \begin{cases} -\frac{d_i - b_{\ell,i}(\tau)}{\Delta d_i}, & \text{if } \Delta d_i < 0 \\ \frac{b_{u,i}(\tau) - d_i}{\Delta d_i}, & \text{if } \Delta d_i > 0 \\ \infty, & \text{otherwise.} \end{cases}$$

$$\Delta\tau_{\mathrm{d}} = \min_{i\in\mathcal{W}} \alpha_i^{\mathrm{d}}, \quad \text{where } \alpha_i^{\mathrm{d}} = \begin{cases} -\frac{\lambda_{m+i}}{\Delta\lambda_{m+i}}, & \text{if } \operatorname{sgn}(\lambda_{m+i}) \neq \operatorname{sgn}(\Delta\lambda_{m+i}) \\ \infty, & \text{otherwise.} \end{cases}.$$

   Set $\Delta\tau = \min\{\Delta\tau_p, \Delta\tau_d\}$ and $p = \arg\min_{i\in\mathcal{W}^{\complement}} \alpha_i^{\mathrm{p}}$ or $q = \arg\min_{i\in\mathcal{W}^{\complement}} \alpha_i^{\mathrm{d}}$.
4. If $\Delta\tau^{[\nu]} \geq 1 - \tau^{[\nu]}$, then stop with $w^{[\nu+1]} = w^{[\nu]} + (1 - \tau^{[\nu]})\Delta w^{[\nu]}$ as the solution of (8.4).
5. Set $\tau^{[\nu+1]} = \tau^{[\nu]} + \Delta\tau^{[\nu]}$, $w^{[\nu+1]} = w^{[\nu]} + \Delta\tau^{[\nu]}\Delta w^{[\nu]}$, and $\mathcal{W}^{[\nu+1]} = \mathcal{W}^{[\nu]}$.
6. If bound of primal variable $d_p^{[\nu+1]}$ is blocking:
   (a) Set $\mathcal{W}^{[\nu+1]} = \mathcal{W}^{[\nu+1]} \cup \{p\}$.
   (b) If the new working set $\mathcal{W}^{[\nu+1]}$ is linearly dependent, find an exchange constraint index q or stop due to infeasibility of (8.4) for all $\tau > \tau^{[\nu+1]}$. Adjust dual variables $\lambda^{[\nu+1]}$ and set $\mathcal{W}^{[\nu+1]} = \mathcal{W}^{[\nu+1]} \setminus \{q\}$.
7. If sign change of dual variable $\lambda_{m+q}^{[\nu+1]}$ is blocking:
   (a) Set $\mathcal{W}^{[\nu+1]} = \mathcal{W}^{[\nu+1]} \setminus \{q\}$
   (b) If B has nonpositive curvature on the null space of $\mathcal{W}^{[\nu+1]}$, find an exchange bound index p or stop due to unboundedness of (8.4) for all $\tau > \tau^{[\nu+1]}$. Adjust primal variables $d^{[\nu+1]}$ and set $\mathcal{W}^{[\nu+1]} = \mathcal{W}^{[\nu+1]} \cup \{p\}$.
8. Set $\nu = \nu + 1$ and continue with Step 2.

and denote by $\lambda_{\mathcal{W}}^{[\nu]}$ the corresponding multipliers. Furthermore, we define the right-hand side vector $b_{\mathcal{W}}(\tau^{[\nu]})$ as

$$b_{\mathcal{W}}(\tau^{[\nu]}) = \begin{bmatrix} -c(\tau^{[\nu]}) \\ \mathbb{I}_{\mathcal{W}^{[\nu]}} b(\tau^{[\nu]}) \end{bmatrix},$$

where the vector $b(\tau^{[\nu]})$ consists of the entries

$$(b(\tau^{[\nu]}))_i = \begin{cases} (b_\ell(\tau^{[\nu]}))_i, & \text{if } d_i^{[\nu]} = (b_\ell(\tau^{[\nu]}))_i \\ (b_u(\tau^{[\nu]}))_i, & \text{if } d_i^{[\nu]} = (b_u(\tau^{[\nu]}))_i \\ 0 \quad , & \text{otherwise.} \end{cases}$$

We can then determine the step direction $(\Delta d^{[\nu]}, \Delta\lambda_{\mathcal{W}}^{[\nu]})$ by solving

$$\begin{bmatrix} B & A_{\mathcal{W}^{[\nu]}}^T \\ A_{\mathcal{W}^{[\nu]}} & \end{bmatrix} \begin{bmatrix} \Delta d^{[\nu]} \\ -\Delta\lambda_{\mathcal{W}}^{[\nu]} \end{bmatrix} = \begin{bmatrix} -(g(1) - g(\tau^{[\nu]}) \\ b_{\mathcal{W}}(1) - b_{\mathcal{W}}(\tau^{[\nu]}) \end{bmatrix}. \tag{8.7}$$

For $i \neq \mathcal{W}^{[\nu]}$, we set $\Delta\lambda_i^{[\nu]} = 0$.

Determining linear dependence (Step 6)
The new working set $\mathcal{W}^{[\nu+1]}$ is formed by addition of a new bound p to the working set $\mathcal{W}$, which can lead to rank deficiency of $A_{\mathcal{W}^{[\nu+1]}}$ and thus loss of invertibility in Eq. (8.7). We can check for linear dependency of $e_p \in \mathbb{R}^n$ on $A_{\mathcal{W}^{[\nu]}}$ by solving

$$\begin{bmatrix} B & A_{\mathcal{W}^{[\nu]}}^T \\ A_{\mathcal{W}^{[\nu]}} & \end{bmatrix} \begin{bmatrix} s \\ \zeta \end{bmatrix} = \begin{bmatrix} e_p \\ 0 \end{bmatrix}. \tag{8.8}$$

e_p is linearly dependent on $A_{\mathcal{W}^{[\nu]}}$ if and only if $s = 0$. In this case, an exchange bound must leave the working set to resolve linear dependency. To determine the bound, see that Eq. (8.8) yields

$$0 = -e_p + A_{\mathcal{W}^{[\nu]}}^T \zeta. \tag{8.9}$$

Multiplying Eq. (8.9) by μ—where $\mu \geq 0$ if the new bound p is active at its upper bound, and $\mu \leq 0$ if it is active at its lower bound—and adding this as a special form of zero to the stationarity conditions (8.2a) yields

$$\begin{aligned} B(d^{[\nu]} + \Delta\tau\Delta d^{[\nu]}) + g(\tau^{[\nu+1]}) &= A_{\mathcal{W}^{[\nu]}}^T \lambda_{\mathcal{W}}^{[\nu]} \\ &= -\mu e_p + A_{\mathcal{W}^{[\nu]}}^T (\lambda_{\mathcal{W}}^{[\nu]} + \mu\zeta) \end{aligned} \tag{8.10}$$

All coefficients on the right-hand side of Eq. (8.10) are also valid choices $\tilde{\lambda}$ for the dual variables as long as they have the correct sign. Hence, we need to compute the index q and multiplier μ of the exchange constraint. We determine μ^* by the following ratio test:

$$\mu^* := \min_{i \in \mathcal{W}^{[\nu+1]}} \alpha_i^{\mathrm{d}}, \quad \text{where } \alpha_i^{\mathrm{d}} = \begin{cases} -\frac{\lambda_{m+i}}{\zeta_{m+i}}, & \text{if } \operatorname{sgn}(\lambda_{m+i}) \neq \operatorname{sgn}(\mu\zeta_{m+i}) \\ \infty, & \text{otherwise.} \end{cases} \tag{8.11}$$

If $\mu^* = \infty$, then the parametric QP is infeasible beyond $\tau^{[\nu+1]}$, in particular in $\tau = 1$ by convexity of the feasible set. Otherwise, let q be a minimizing index of the ratio set und let $\lambda^{[\nu+1]} := \tilde{\lambda}$ where

$$\tilde{\lambda}_{m+p} = -\mu^*, \quad \tilde{\lambda}_i = \lambda_i^{[\nu+1]} + \mu^* \zeta_i \tag{8.12}$$

Furthermore, $\tilde{\lambda}_{m+q}$ vanishes by construction of μ^*. Removing constraint q from $\mathcal{W}^{[\nu+1]}$ restores linear independence, see [21] for a proof.

Determining zero curvature (Step 7)
The new working set $\mathcal{W}^{[\nu+1]}$ is formed by removal of bound q from $\mathcal{W}^{[\nu]}$. The dimension of the null space of $A_{\mathcal{W}^{[\nu+1]}}$ grows by one compared to $A_{\mathcal{W}^{[\nu]}}$. This may lead to exposition of a direction of zero curvature in the null space, which implies loss of invertibility in Eq. (8.7). Directions of zero curvature can be detected by solving

$$\begin{bmatrix} B & A_{\mathcal{W}^{[\nu]}}^T \\ A_{\mathcal{W}^{[\nu]}} & \end{bmatrix} \begin{bmatrix} s \\ \zeta \end{bmatrix} = \begin{bmatrix} 0 \\ -e_{m+q} \end{bmatrix}. \tag{8.13}$$

with $e_{m+q} \in \mathbb{R}^{m+\left|\mathcal{W}^{[\nu]}\right|}$. B is singular on the null space of $A_{\mathcal{W}^{[\nu+1]}}$ if and only if $\zeta = 0$, see [21]. The zero curvature disappears if a suitable exchange bound is added to the working set instead. To determine the exchange bound, note that s solves

$$Bs = 0, \quad e_{m+q}s = -1, \quad A_{\mathcal{W}^{[\nu+1]}}s = 0$$

and all points $\tilde{d} = d^{[\nu+1]} + \sigma s$, $\sigma > 0$ are also optimal solutions if $\tilde{d}$ is primal feasible. We can determine the largest σ from the ratio tests

$$\sigma = \min_{i \in (\mathcal{W}^{[\nu+1]})^{\complement}} \alpha_i^{\mathrm{p}} = \begin{cases} -\frac{d_i^{[\nu+1]} - b_{\ell,i}(\tau)}{s_i}, & \text{if } s_i < 0 \\ \frac{b_{u,i}(\tau) - d_i^{[\nu+1]}}{s_i}, & \text{if } s_i > 0 \\ \infty, & \text{otherwise.} \end{cases}$$

If $\sigma = \infty$, then the parametric QP is unbounded beyond $\tau^{[\nu+1]}$, and in particular in $\tau = 1$. Otherwise, let p be a minimizing index of the ratio set, and let $d^{[\nu+1]} := d^{[\nu+1]} + \sigma s$. By construction of σ, the constraint row p is active in $d^{[\nu+1]}$ and can be added to the working set. Again, we refer to [21] for a proof.

8.2. Linear algebra

The main computational burden associated with solving a QP (8.1) is the solution of the KKT systems in Eq. (8.7) and (8.8) or (8.13) in every iteration ν of the parametric quadratic programming algorithm. First we describe how the dimension of the system can be reduced from $n+m+|\mathcal{W}|$ to $n+m-|\mathcal{W}|$ by exploiting the fact that inequalities are only given as simple bounds on the variables. Then we describe a Schur complement approach that allows to employ efficient sparse solvers for the large KKT matrix at the first iteration and maintain dense updates for a small Schur complement as the iterations proceed.

8.2.1. Simple bounds

For the solution of the $n+m+|\mathcal{W}|$ system (8.7) during an iteration ν, consider first a permutation and partition of the primal variables into free and fixed variables, $\Delta d = (\Delta d_F, \Delta d_X)$ according to the current working set $\mathcal{W}$. Similarly, we partition $\Delta\lambda_{\mathcal{W}} = (\Delta\lambda_A, \Delta\lambda_X)$ into multipliers for linear constraints and active variable bounds. The generic right-hand side vectors $g = (g_F, g_X)$ and $b = (b_A, b_X)$ as well as Hessian H and Jacobian A are permuted and partitioned accordingly,

$$B = \begin{pmatrix} B_{FF} & B_{FX}^T \\ B_{FX} & B_{XX} \end{pmatrix}, \qquad A = \begin{pmatrix} A_F & A_X \end{pmatrix}.$$

Then system (8.7) reads as

$$\begin{bmatrix} B_{FF} & B_{FX}^T & A_F^T & \\ B_{FX} & B_{XX} & A_X^T & \mathbb{I} \\ A_F & A_X & & \\ & \mathbb{I} & & \end{bmatrix} \begin{bmatrix} \Delta d_F \\ \Delta d_X \\ -\Delta\lambda_A \\ -\Delta\lambda_X \end{bmatrix} = \begin{bmatrix} g_F \\ g_X \\ b_A \\ b_X \end{bmatrix}.$$

From this system, we immediately see that $\Delta d_X = b_X$. We can now obtain Δd_F and $\Delta\lambda_A$ by solving the reduced KKT system

$$\begin{bmatrix} B_{FF} & A_F^T \\ A_F & \end{bmatrix} \begin{bmatrix} \Delta d_F \\ -\Delta\lambda_A \end{bmatrix} = \begin{bmatrix} g_F - B_{FX}^T b_X \\ b_A - A_X b_X \end{bmatrix}. \tag{8.14}$$

Finally, we recover the multipliers for the fixed variables as

$$\Delta\lambda_X = B_{FX}\Delta d_F + H_{XX} b_X - A_X^T \Delta\lambda_A - g_X.$$

Systems (8.8), and (8.13) are also solved using the reduced system (8.14) and appropriate right-hand sides.

8.2.2. Schur complement approach

Every active-set iteration ν in the parametric active-set QP method requires the solution of linear systems of the form (8.14). We denote the reduced KKT matrix from (8.14) in iteration ν by

$$K^{[\nu]} := \begin{bmatrix} B^{[\nu]} & {A^{[\nu]}}^T \\ A^{[\nu]} & \end{bmatrix}, \tag{8.15}$$

where $H^{[\nu]}$ is the matrix obtained from H by deleting the rows and columns that correspond to the indices in $\mathcal{W}^{[\nu]}$. Similarly, $A^{[\nu]}$ is obtained from A by deleting the columns corresponding to the indices in $\mathcal{W}^{[\nu]}$.

In many application, in particular our problems from DMS, this matrix is sparse, cf. Fig 2.2. It can be efficiently factored using a symmetric indefinite LBL^T-factorization. In

our implementation, this is accomplished by the sparse multifrontal solver `MA57` [63] with approximate minimum degree ordering computed by `MC47` [6]. Other direct methods for sparse systems that could be used in this context include `PARDISO` [175] and `MUMPS` [7].

Computing a new LBL^T factorization every time the working set changes would be inefficient. Instead we follow the Schur complement approach described by [27, 94, 97, 104], which computes a factorization of (8.15) only for the initial working set $\mathcal{W}^{[0]}$ set and then maintains and updates factors for a Schur complement as the iterations proceed.

Forming $K^{[\nu]}$ by augmentation

When the working set changes, the KKT matrix changes by a single row and column. Instead of factoring the matrix (8.15) again, the first matrix $K^{[0]}$ may be bordered in a way that reflects the changes to the working set during a set of ν subsequent iterations. The solution of systems involving (8.15) is then found by using a fixed factorization of $K^{[0]}$, and a factorization of a smaller matrix of order at most 2ν.

To see this, let us first consider the case where a variable with index p enters the working set in the first iteration. Then $K^{[1]}$ is obtained by bordering $K^{[0]}$ with the p-th unit row and column vector $e_p \in \mathbb{R}^n$:

$$K^{[1]} = \left[\begin{array}{cc|c} H^{[0]} & {A^{[0]}}^T & e_p \\ A^{[0]} & 0 & 0 \\ \hline e_p^T & 0 & 0 \end{array}\right]$$

Now assume that a variable with index $q \neq p$ leaves the working set in the second iteration. Then $K^{[1]}$ is bordered as follows:

$$K^{[2]} = \left[\begin{array}{ccc|c} H^{[0]} & {A^{[0]}}^T & e_p & H_{\mathcal{W}^{[0]},q} \\ A^{[0]} & 0 & 0 & A_q \\ e_p^T & 0 & 0 & 0 \\ \hline H_{\mathcal{W}^{[0]},q}^T & A_q^T & 0 & H_{qq} \end{array}\right],$$

where A_q is the q-th columns of A, $H_{\mathcal{W}^{[0]},q}$ is the q-th column of H with the elements corresponding to the working set deleted, and H_{qq} is the q-th diagonal element of H. If p leaves or q enters the working set again in a future iteration, we simply delete the corresponding rows and columns.

In summary, after ν iterations, the KKT system is maintained as a symmetric augmented system of the form

$$K^{[\nu]} = \begin{bmatrix} K^{[0]} & M^{[\nu]} \\ {M^{[\nu]}}^T & N^{[\nu]} \end{bmatrix}, \ \nu = 1, 2, \ldots. \tag{8.16}$$

where $N^{[\nu]}$ is of dimension of at most 2ν because when a bound is added there may be an exchange bound that is removed due to linear dependency. Conversely, when a bound is removed there may be another bound added due to zero curvature.

Solving sytems involving $K^{[\nu]}$
Suppose we now want to solve a system involving $K^{[\nu]}$

$$\begin{bmatrix} K^{[0]} & M^{[\nu]} \\ {M^{[\nu]}}^T & N^{[\nu]} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} r \\ t \end{bmatrix}$$

with appropriately chosen vectors α, β, r, t to represent one of the systems (8.7), (8.8), or (8.13). The key insight is that a solution with $K^{[\nu]}$ only requires the solution of the following systems:

$$K^{[0]}u = r, \qquad S^{[\nu]}\beta = t - {M^{[\nu]}}^T u, \qquad K^{[0]}\alpha = r - M^{[\nu]}\beta,$$

where $S^{[\nu]}$ is the *Schur complement of* $K^{[0]}$ *in* $K^{[\nu]}$:

$$S^{[\nu]} = N^{[\nu]} - {M^{[\nu]}}^T {K^{[0]}}^{-1} M^{[\nu]}. \tag{8.17}$$

This can be done at greatly reduced cost compared to a new factorization of $K^{[\nu]}$: The LBL^T-factors for $K^{[0]}$ are computed only once at the beginning, and we maintain a dense QR factorization of the small Schur complement $S^{[\nu]}$.

The question remains how to maintain and update the Schur complement. For this, consider an iteration $\nu+1$ with a single change in the working set such that either a bound is added that has not been removed before or a bound is removed that has not been added before. Then the matrices $M^{[\nu+1]}$ and $N^{[\nu+1]}$ have the form

$$M^{[\nu+1]} = \begin{bmatrix} M^{[\nu]} & m \end{bmatrix}, \qquad N^{[\nu+1]} = \begin{bmatrix} N^{[\nu]} & v \\ v^T & \sigma \end{bmatrix},$$

with m, v, and σ depending on the type of change in the working set. Following the definition of the Schur complement (8.17), the new Schur complement $S^{[\nu+1]}$ is given by

$$S^{[\nu+1]} = \begin{bmatrix} N^{[\nu]} & v \\ v^T & \sigma \end{bmatrix} - \begin{bmatrix} {M^{[\nu]}}^T \\ m^T \end{bmatrix} {K^{[0]}}^{-1} \begin{bmatrix} M^{[\nu]} & m \end{bmatrix} = \begin{bmatrix} S^{[\nu]} & u \\ u^T & \gamma \end{bmatrix} \tag{8.18}$$

with $K^{[0]}t = m$, $u = v - {M^{[\nu]}}^T u$ and $\gamma = \sigma - m^T t$. Note that another solve with $K^{[0]}$ is required to update S.

Thus the new Schur complement is constructed from the old one by bordering it with one row and column. On the other hand, if a working set change reverts an earlier update of the Schur complement, the new Schur complement shrinks by one row and one column. For completeness we have included Algorithm 10, which details the procedure of updating a dense QR factorization of the symmetric matrix S. The algorithm is standard, see [103]. Note that other factorizations for S are possible.

In practice, the KKT matrix $K^{[\nu]}$ is refactorized from time to time and replaces the factorizations of $K^{[0]}$ and $S^{[\nu]}$, e.g. when the dimension of $S^{[\nu]}$ exceeds a given threshold `nSmax` and the cost of maintaining the dense factors of $S^{[\nu]}$ becomes too high.

Algorithm 10: Updating QR factors of the Schur complement.

Goal: Compute QR factors of the Schur complement $S^{[\nu+1]}$ of dimension $n_S^{[\nu+1]}$ after a working set change using existing QR factors for $S^{[\nu]} = Q^{[\nu]}R^{[\nu]}$.
Case 1. $S^{[\nu+1]}$ is $S^{[\nu]}$ bordered by a row and a column, $n_S^{[\nu+1]} = n_S^{[\nu]} + 1$

1. Set

$$Q^{[\nu+1]^T} S^{[\nu+1]} = \begin{bmatrix} Q^{[\nu]^T} & \\ & 1 \end{bmatrix} \begin{bmatrix} S^{[\nu]} & u \\ u^T & \gamma \end{bmatrix}$$
$$= \begin{bmatrix} Q^{[\nu]^T} S^{[\nu]} & Q^{[\nu]^T} u \\ u^T & \gamma \end{bmatrix} = \begin{bmatrix} R^{[\nu]} & Q^{[\nu]^T} \\ u^T & \gamma \end{bmatrix} = R^{[\nu+1]}$$

2. Apply $n_S^{[\nu+1]} - 1$ Givens rotations to zero components of u^T in the last row of $R^{[\nu+1]}$:

$$G^T(n_S^{[\nu+1]} - 1, n_S^{[\nu+1]}) \cdots G^T(1, n_S^{[\nu+1]}) \cdot Q^{[\nu+1]^T} \cdot S^{[\nu+1]} =$$
$$G^T(n_S^{[\nu+1]} - 1, n_S^{[\nu+1]}) \cdots G^T(1, n_S^{[\nu+1]}) \cdot R^{[\nu+1]}$$

Case 2. $S^{[\nu+1]}$ is $S^{[\nu]}$ with row and column $\bar{j}$ deleted, $n_S^{[\nu+1]} = n_S^{[\nu]} - 1$

1. Delete column $\bar{j}$ to obtain $n_S^{[\nu]} \times n_S^{[\nu+1]}$ matrices R_1 and S_1. Apply $n_S^{[\nu+1]} - \bar{j}$ Givens rotations to bring R_1 to triangular form:

$$G^T(n_S^{[\nu+1]} - 1, n_S^{[\nu+1]}) \cdots G^T(\bar{j}, \bar{j}+1) \cdot Q^{[\nu]^T} \cdot S_1 =$$
$$G^T(n_S^{[\nu+1]} - 1, n_S^{[\nu+1]}) \cdots G^T(\bar{j}, \bar{j}+1) \cdot R_1$$

2. Simultaneous permutation of $Q^{[\nu]}$ and S_1: Move row $\bar{j}$ to position $n_S^{[\nu]}$

3. Delete row $n_S^{[\nu]}$. Apply $n_S^{[\nu+1]}$ Givens rotations to zero the first $n_S^{[\nu+1]}$ elements of the last column of $Q^{[\nu]^T}$. The last element will be α with $\alpha \in \{-1, 1\}$. Note that the leading $n_S^{[\nu+1]}$ elements of the last row of $Q^{[\nu]^T}$ will be zero automatically due to orthogonality.

$$G^T(n_S^{[\nu]}, 1) \cdot G^T(n_S^{[\nu]}, n_S^{[\nu+1]}) \cdot Q^{[\nu]^T} \cdot \begin{bmatrix} S^{[\nu+1]} \\ u^T \end{bmatrix} = \begin{bmatrix} Q^{[\nu+1]^T} & 0 \\ 0 & \alpha \end{bmatrix} \begin{bmatrix} S^{[\nu+1]} \\ u^T \end{bmatrix}$$
$$= G^T(n_S^{[\nu]}, 1) \cdot G^T(n_S^{[\nu]}, n_S^{[\nu+1]}) \cdot R_1 = \begin{bmatrix} R^{[\nu+1]} \\ v^T \end{bmatrix}.$$

$Q^{[\nu+1]}$ and $R^{[\nu+1]}$ are the new factors.

4. If row permutation (i.e. $n_S^{[\nu+1]} - \bar{j}$) was odd and $\alpha = 1$ or row permutation was even and $\alpha = -1$: Change the sign of the first column of $Q^{[\nu+1]}$ and the first row of $R^{[\nu+1]}$ to maintain a $Q^{[\nu+1]}$ with $\det(Q^{[\nu+1]}) = 1$

8.3. Handling nonconvexity

The discussion so far has only been concerned with positive semidefinite Hessians. An important feature of our SQP Algorithm 6 is that indefinite Hessians are allowed as long as they are positive definite on the null space of $A_{\mathcal{S}}$. In this section, we show how Step 7 of Algorithm 9 must be modified to handle indefinite Hessians. We first summarize the *flipping bounds* strategy that is invoked by `qpOASES` whenever the nullspace picks up a negative eigenvalue. Afterwards we describe two possibilities how negative curvature can be detected when the nullspace grows. The first one is specifically tailored to the Schur complement approach by monitoring the inertia of the Schur complement. The second one is a generalization of the nonzero curvature test described above. We implemented the first strategy in `qpOASES` for our numerical tests.

8.3.1. Flipping bounds strategy

The flipping bounds strategy relies on the fact that all variables have finite bounds. If one or more bounds are missing, $b_{\ell,i} = -\infty$ or $b_{u,i} = \infty$, we employ a *far bounds* strategy [70]. There, infinite or large entries $b_{\ell,i}$ and $b_{u,i}$ are temporarily set to finite, moderately large values, $M_{\ell,i} \approx -10^6$ and $M_{u,i} \approx 10^6$, respectively. Then we solve the QP with far bounds and distinguish two cases:

1. If no far bounds are active at the solution, this is the solution of the original QP.
2. If the QP with far bounds is infeasible or at least one far bound is active, we enlarge $M_{\ell,i}$ and $M_{u,i}$ by a growth factor, say 10^3, and solve again, using efficient hot starts.

If case 1 has not occured and the value of the far bounds exceed a very large value considered infinity, say 10^{20}, the problem is either declared infeasible, if the last QP solved was infeasible, or unbounded, if the last QP contained an active far bound. Note that the current working set and matrix factorizations can be reused in the sequence of QPs with growing far bounds, which allows the far bounds strategy to be carried out very efficiently.

We may now assume all variables have finite bounds. Consider the following situation: A bound q leaves the working set, i.e., a sign change in the dual variables is blocking, and no other bound enters, and we detect that by removing this bound a negative eigenvalue appears in the null space, see Sec. 8.3.2 below for details. In this case we introduce a jump in the QP homotopy by moving the opposite bound of $d_q^{[\nu]}$ such that it becomes active immediately. This is easily achieved by setting

$$b_{\ell,q}(\tau^{[\nu]}) := b_{u,q}(\tau^{[\nu]}), \quad \text{if } \lambda_q^{[\nu]} \leq 0 \text{ and } \Delta\lambda_q^{[\nu]} > 0$$
$$b_{u,q}(\tau^{[\nu]}) := b_{\ell,q}(\tau^{[\nu]}), \quad \text{if } \lambda_q^{[\nu]} \geq 0 \text{ and } \Delta\lambda_q^{[\nu]} < 0,$$

and adding q again to the working set. This way, primal and dual feasibility is retained and we can proceed with the iterations. In particular, the factorization from the previous iteration stays valid. We next describe how to detect if removing a bound exposes negative curvature in the null space.

8.3.2. Detecting negative curvature by monitoring the inertia of the KKT matrix

One possibility to ensure that $B^{[\nu]}$ stays positive definite on the null space of $A^{[\nu]}$ throughout the working set iterations is to monitor the inertia $\mathrm{In}(K^{[\nu]}) = (n_+, n_-, n_0)$ of $K^{[\nu]}$, where n_+, n_-, and n_0 are the number of positive, negative, and zero eigenvalues of $K^{[\nu]}$. Our strategy is similar to the one proposed in [104] and is based on two results that can be proven by using Sylvester's law of inertia.

Theorem 8.2 (Gould, [105])**.** *Let B be an $n \times n$ symmetric matrix, A an $m \times n$ matrix of full row rank, K the KKT matrix of* (8.1) *partitioned as in* (8.15)*, and Z a nullspace basis for A. Then*

$$\mathrm{In}(K) = \mathrm{In}\begin{bmatrix} B & A^T \\ A & \end{bmatrix} = \mathrm{In}(Z^T B Z) + (m, m, 0)$$

Theorem 8.3 (Haynsworth, [119])**.** *Let K be the KKT matrix of* (8.1) *partitioned as in* (8.16)*. Then*

$$\mathrm{In}(K^{[\nu]}) = \mathrm{In}(K^{[0]}) + \mathrm{In}(S^{[\nu]}).$$

Theorem 8.2 allows to deduce the inertia of the reduced Hessian from the inertia of the KKT matrix. Theorem 8.3 allows to monitor the inertia of $K^{[\nu]}$ during modifications of the Schur complement. As the iterations proceed, we always keep the reduced Hessian positive definite such that the second-order optimality conditions stated in Thm. 8.1 are satisfied.

We start with an initial KKT matrix $K^{[0]}$ with correct inertia. The case of incorrect inertia is addressed in Section 8.3.4 below. Thm 8.3 implies that the inertia of $K^{[\nu]}$ is correct if and only if

$$\mathrm{In}(S^{[\nu]}) = (\sigma_-, \sigma_+, 0), \tag{8.19}$$

where σ_+ is the number of variables added and σ_- is the number of variable removed from the set $\mathcal{W}^{[\nu]}$ since the most recent LBL^T-factorization of the KKT matrix. The inertia of $K^{[\nu]}$ can thus be determined from the inertia of $S^{[\nu]}$, and it can be tracked efficiently by observing the sign of the determinant of $S^{[\nu]}$ during every modification of $S^{[\nu]}$. The determinant is easily computable: We construct the QR factors of $S^{[\nu]}$ such that $\det Q = 1$ always holds for the orthogonal matrix Q. Then $\det S^{[\nu]}$ is just the product of the diagonal entries of the triangular factor R.

To see how the sign of the determinant changes, let $S^{[\nu+1]}$ and $S^{[\nu]}$ have dimensions $n_S^{[\nu+1]}$ and $n_S^{[\nu]}$, respectively. A negative eigenvalue can turn up in the null space when we remove a variable from the working set. For the Schur complement, one of three cases can occur:

- **Case 1: The Schur complement grows.**

  The bound has been in the initial working set and is now to be removed. The Schur complement is bordered by one row and column, and the new Schur complement is

given by

$$S^{[\nu+1]} = \begin{pmatrix} S^{[\nu]} & u \\ u^T & \gamma \end{pmatrix}.$$

Its determinant evaluates to $\det S^{[\nu+1]} = \det S^{[\nu]} \cdot (\gamma - u^T {S^{[\nu]}}^{-1} u)$. If $\operatorname{sgn}(\det S^{[\nu+1]}) \neq \operatorname{sgn}(\det S^{[\nu]})$, the Schur complement gained a negative eigenvalue which is not consistent with the necessary condition (8.19). We set $S^{[\nu+1]} = S^{[\nu]}$ and the flipping bounds strategy is invoked.

- **Case 2: The Schur complement shrinks.**

  The bound has not been in the initial working set but has subsequently been added—causing the Schur complement to grow and gain a negative eigenvalue according to (8.19)—and is now to be removed again. Suppose row and column $1 \leq \bar{j} \leq n_S^{[\nu]}$ are deleted from the Schur complement $S^{[\nu]}$; the remainder forms $S^{[\nu+1]}$. We compute $\det S^{[\nu]} = \det R^{[\nu+1]}$ by updating the diagonal elements of $R^{[\nu]}$. If $\operatorname{sgn}(\det S^{[\nu+1]}) = \operatorname{sgn}(\det S^{[\nu]})$, the Schur complement lost a positive eigenvalue. This contradicts (8.19), we set $S^{[\nu+1]} = S^{[\nu]}$ and the flipping bounds strategy is invoked.

- **Case 3: Vacuous Schur complement after a fresh factorization.**

  The Schur complement has reached its maximum size, but a bound to be removed was present in the initial working set $\mathcal{W}^{[0]}$ and the Schur complement needed to grow. In this case, the KKT matrix $K^{[\nu]}$ is refactorized, and $S^{[\nu+1]}$ is now vacuous. We check the inertia of the new factorization relying on `MA57` to provide this information. If there are more negative eigenvalues than active linear constraints, we start the new Schur complement $S^{[\nu+1]}$ by adding the removed bound at the opposite bound, and invoke the flipping bounds strategy.

In case 1 and 2, the flipping decisions based on the determinant of $S^{[\nu+1]}$ are made before the QR factorization of $S^{[\nu]}$ is updated. In case 1, the computation of $\det(S^{[\nu+1]})$ requires a solve with the known factorization of $S^{[\nu]}$. In case 2, $n_S^{[\nu+1]}$ Givens rotations, applied only to one column of $Q^{[\nu]}$ and the diagonal elements of $R^{[\nu]}$, are required as well as $(n_S^{[\nu+1]} - \bar{j})$ rotations applied to one column of $Q^{[\nu]}$ and the $3 \cdot (n_S^{[\nu+1]} - \bar{j})$ tridiagonal elements of $R^{[\nu+1]}$. Case 3 occurs only infrequently, depending on the maximum allowed size of the Schur complement.

Observing Theorems 8.2 and 8.3, the procedure described above can also be used to detect zero curvature. In this case, a zero eigenvalue appears in the Schur complement. It is thus suitable to replace Step 7 of Algorithm 9 within a Schur complement approach and the additional factorization therein can be avoided. A generalization of Step 7 that is independent of the linear algebra is described next.

8.3.3. Detecting negative curvature by a solve with $K^{[v]}$

Another possibility to detect negative curvature in Step 7 of Algorithm 9 is given by the following Lemma.

Lemma 8.4. *Let B be an $n \times n$ symmetric matrix and A be an $m \times n$ matrix of full row rank with $m \leq n$. Let furthermore B be positive definite on the nullspace of A. Denote by A^- the $(m-1) \times n$ matrix that is formed from A by removal of the i-th row and let $s \in \mathbb{R}^n$ and $\zeta \in \mathbb{R}^m$ be the solution of*

$$\begin{bmatrix} B & A^T \\ A & \end{bmatrix} \begin{bmatrix} s \\ \zeta \end{bmatrix} = \begin{bmatrix} 0 \\ -e_i \end{bmatrix}. \tag{8.20}$$

Then B is $\left\{ \begin{array}{l} \text{positive definite} \\ \text{indefinite} \\ \text{singular} \end{array} \right\}$ *on the nullspace of A^- if and only if* $\left\{ \begin{array}{l} \zeta_i > 0 \\ \zeta_i < 0 \\ \zeta_i = 0 \end{array} \right\}$.

Proof. By assumption, B is positive definite on the nullspace of A. Every vector that is in the nullspace of A is also in the nullspace of A^-. Thus we only need to consider the one-dimensional subspace of vectors v for which $A^-v = 0$ but $Av \neq 0$.

The second row of Eq. (8.20) implies that $As = -e_i$ and $A^-s = 0$. In particular, $s \neq 0$. Multiplying the first row with s^T from the left yields

$$s^T Bs + s^T A^T \zeta = s^T Bs - e_i^T \zeta = 0.$$

This is equivalent to

$$s^T Bs = \zeta_i,$$

and thus ζ_i indicates positive or negative definiteness, or singularity of B on the subspace spanned by s. □

Lemma 8.4 means that the sign of ζ_i indicates if the newly exposed curvature by removal of a bound is positive, negative, or zero. If it is negative, the flipping bounds strategy is invoked.

8.3.4. Inertia correction

Whenever the initial KKT matrix $K^{[0]}$ does not have the desired inertia directly after a factorization, we proceed as follows to obtain a working set that gives rise to a KKT matrix with correct inertia. We add one by one bounds of free variables to the working set $\mathcal{W}^{[0]}$ and monitor the inertia during every update of the Schur complement as described above. In our implementation, we add variables with ascending index to their nearest bound. This procedure iteratively reduces the dimension of the null space. It is terminated as soon as the inertia is as desired, or when the null space has eventually become vacuous. A similar procedure is invoked every time the KKT matrix needs to be refactorized and its inertia is found to be incorrect due to accumulation of round-off errors from earlier working set iterations.

8.4. Practical issues

The new algorithm using the Schur complement approach and the inertia monitoring strategy outlined above have been implemented in a variant of the QP solver `qpOASES`. In this section we first discuss some practical issues that must be considered in an efficient implementation. Then we present an algorithmic technique that allows us to efficiently verify positive definiteness of the Hessian in the null space of $A_{\mathcal{S}}$ which is required to make our SQP Algorithm 6 efficient.

8.4.1. Refactorization and linear independence

From time to time, a new LBL^T-factorization of the KKT matrix $K^{[\nu]}$ is computed and replaces the factorizations of $K^{[0]}$ and $S^{[\nu]}$. In our implementation, this occurs in two situations.

Because the cost for maintaining a dense QR factorization of the Schur complement $S^{[\nu]}$ grows with its size, a new LBL^T-factorization is computed when the dimension of $S^{[\nu]}$ exceeds a given threshold `nSmax`. A refactorization is also triggered when an estimate of the condition number of $S^{[\nu]}$ is larger than a given threshold `condMax`. In our implementation, the estimate is computed by the `LAPACK` routine `DTRCON` which is applied to the triangular factor R of the QR decomposition. We found that the values `nSmax`=100 and `condMax`=10^{14} work well in practice.

It is possible that the LBL^T-factorization cannot be performed because the matrix $K^{[\nu]}$ is singular. Recall that the final working set from the previous SQP iteration is used as the initial guess for $\mathcal{W}^{[0]}$ at the beginning of the solution of QP($B^{[k,l]}$). However, $K^{[0]}$ may be singular because the constraint Jacobian $A^{[k]}$ and Hessian approximation $B^{[k,l]}$ are different at the new iterate. Similarly, during a later QP iteration ν, a variable may have been added to the working set even though the resulting matrix $A^{[\nu]}$ no longer has full rank. This can occur because of numerical errors in the test for linear-independence due to finite precision. In this situation, if a refactorization is triggered at a later point, the corresponding KKT matrix is singular and the LBL^T-factorization cannot be performed.

In both cases, we make use of a feature of the symmetric indefinite solver `MA57` and obtain the list of zero pivot indices. This enables us to remove linearly dependent rows and columns from $K^{[\nu]}$ by manipulating the working set $\mathcal{W}^{[\nu]}$ appropriately. According to this list, we either (i) add a variable from the working set $\mathcal{W}^{[\nu]}$ if the zero pivot is found in the first block diagonal of (8.15), or (ii) add a slack variable $0 \leq s \leq 0$ to an equality constraint but do not add it to the working set $\mathcal{W}^{[\nu]}$ if the zero pivot corresponds to the second, all-zero block diagonal in (8.15).

8.4.2. Verifying positive definiteness of the Hessian on the null space of $A_{\mathcal{S}}$

We now consider the subproblems QP($B^{[k,l]}$) in Step 2 of the SQP Algorithm 6. For a trial Hessian $B^{[k,l]}$, we need to verify positive definiteness on the null space of $A_{\mathcal{S}}^{[k,l]}$ to decide if a direction produced by the solution of QP($B^{[k,l]}$) is admissible for the line search. Recall the

definitions of the set $\mathcal{S}^{[k,l]}$ given in (6.4):

$$\mathcal{S}^{[k,l]} := \left\{ 1 \leq i \leq n \mid (x_i^{[k]} = b_{\ell,i} \text{ or } x_i^{[k]} = b_{u,i}) \text{ and } d_i^{[k,l]} = 0 \right\}.$$

In particular, $\mathcal{S}^{[k,l]}$ can only be formed after QP($B^{[k,l]}$) has been solved. The associated Jacobian $A_{\mathcal{S}}^{[k,l]}$ is defined as $A^{[k]}$ augmented by unit row vectors corresponding to the elements in $\mathcal{S}^{[k,l]}$.

For each subproblem, `qpOASES` is warm-started using the primal-dual solution and working set of the previous SQP iteration, i.e. the one of the QP($B^{[k-1]}$) that yielded the final accepted direction $d^{[k-1]}$. For this initial working set $\mathcal{W}^{[0]}$, three cases can occur:

- **Case 1: The resulting KKT matrix is singular.**

  Then the working set is augmented according to the procedure described in Section 8.4.1, which gives us a KKT matrix with correct inertia.

- **Case 2: The resulting KKT matrix has correct inertia.**

  Then the parametric active-set Algorithm 9 is carried out to find a critical point of QP($B^{[k,l]}$). Afterwards, we determine the set $\mathcal{S}^{[k,l]}$. To find out the curvature of $B^{[k,l]}$ in the null space of $A_{\mathcal{S}}^{[k,l]}$ we have two options:

  1. We factorize the KKT matrix corresponding to $\mathcal{S}^{[k,l]}$ and obtain its inertia from the linear solver. If there are more negative eigenvalues than rows in A, we proceed with the next iteration $l+1$ in Step 2 of Algorithm 6.
  2. We successively remove the bounds that are in the final working set $\mathcal{W}^{[\nu]}$ but not in $\mathcal{S}^{[k,l]}$ and modify the Schur complement accordingly. By monitoring the inertia as described in Section 8.3.2, we can detect whether the constraint null space picks up a negative eigenvalue of the Hessian matrix. In this case, $B^{[k,l]}$ is also not positive definite on the larger null space of $A_{\mathcal{S}}^{[k,l]}$ and we proceed with the next iteration $l+1$ in Step 2 of Algorithm 6.

  In practice, we found the first option to be faster on average.

- **Case 3: The resulting KKT matrix does not have correct inertia.**

  In this case, $B^{[k,l]}$ cannot be positive definite in the null space of $A_{\mathcal{S}}^{[k,l]}$. To see this, note that the set $\mathcal{S}^{[k,l]}$ is a subset of $\mathcal{W}^{[0]}$ and hence yields a null space of $A_{\mathcal{S}}^{[k,l]}$ that contains the null space of $A_{\mathcal{W}^{[0]}}$ on which the negative curvature is detected. The QP solution is then terminated immediately and the loop in Step 2 of Algorithm 6 proceeds to the next iteration $l+1$. In our numerical experiments, this was often the case and saved many active set iterations.

Part IV.

Software and numerical results

Chapter 9.

Implementations

An important part of this thesis is the implementation of the presented numerical methods as efficient software. We designed two software packages: `blockSQP` is an implementation of the SR1-BFGS SQP method described in Chapters 6–8 that is especially suited for problems with block-diagonal Hessian, such as problems arising in DMS. `muse` is an implementation of the multiple shooting parameterizations for OED problems described in Chapter 4 and comes as extension for the software package VPLAN [134]. Both software packages are designed to be used together—`muse` provides structured and efficient evaluation of the NLP objective and constraints and `blockSQP` solves the block-structured NLP—but we stress that both can be used independently of each other and the interface between them is rather generic. This allows for easy maintenance, enhancement and substitution of individual components. In this chapter, we first describe the software package `blockSQP`. We give some implementation details and show how an NLP is specified using a generic problem specification class. Then, we give a description of our second software package `muse`, that implements the problem specification class to represent OED or OC problems parameterized by DMS.

9.1. `blockSQP`: An SR1-BFGS SQP method for NLPs with block-diagonal Hessian matrix

The filter line search SQP method described in the Chapters 6–8 is implemented in the C++-software package `blockSQP`. It solves NLPs of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \varphi(x) \\ \text{s.t.} \quad & b_\ell \leq \begin{bmatrix} x \\ c(x) \end{bmatrix} \leq b_u. \end{aligned}$$

It is especially suited for NLPs with a block-diagonal Hessian matrix. Quadratic subproblems are solved using a Schur complement variant of the QP solver `qpOASES`. First derivatives of the problem functions must be provided. The block-diagonal Hessian of second derivatives is approximated blockwise by SR1 and BFGS update schemes, however, exact second derivatives may also be included for individual blocks. In this case, update schemes are only maintained for the remaining blocks. The default option is to give the SR1 update preference and employ a BFGS update with selective sizing if the SR1 update yields negative

curvature in the nullspace of A_S. That means, at most two QPs per major iteration are solved. Other Hessian sequences are also possible, such as convex combinations between SR1 and BFGS with a given maximum number of steps $l_{\max}$. The QPs can be solved in parallel as described in Sec. 7.1. Furthermore, `blockSQP` allows to include problem-specific feasibility restoration heuristics.

9.1.1. Problem specification

An NLP is specified using the abstract class `Problemspec` that holds all the information that the SQP method needs to have about the problem:

```
class Problemspec {
  /* DATA MEMBERS */
  public:
    int     nVar;       ///< number of variables
    int     nCon;       ///< number of constraints

    double  objLo;      ///< lower bound for objective
    double  objUp;      ///< upper bound for objective
    Matrix  bl;         ///< lower bounds of variables and constraints
    Matrix  bu;         ///< upper bounds of variables and constraints

    int     nBlocks;    ///< number of diagonal blocks in the Hessian
    int*    blockIdx;   ///< indices corresponding to block structure

  /* METHODS */
  public:
    /** Set initial values for xi and lambda,
     *  and linear constraint matrix */
    virtual void initialize( Matrix &xi, Matrix &lambda,
                             Matrix &constrJac, int *info ) = 0;

    /** Evaluate problem functions and their derivatives at point xi */
    virtual void evaluate( const Matrix &xi,
                           double *objval, Matrix &constr,
                           Matrix &gradObj, Matrix &constrJac,
                           int dmode, int *info ) = 0;

    /** Problem specific heuristic to reduce constraint violation */
    virtual void reduceConstrVio( Matrix &xi, int *info ){};
};
```

An instance of this class describes an NLP of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \varphi(x) \\ \text{s.t.} \quad & b_\ell \leq \begin{bmatrix} x \\ c(x) \end{bmatrix} \leq b_u \end{aligned}$$

with additional information about partial separability of the Lagrangian given by the variables `nBlocks` and `blockIdx`. The following methods must be implemented for a problem:

- `initialize`: Initial values $x^{[0]}$ must be set. Additionally, the part of the Jacobian may be set that corresponds to purely linear constraints and does not change during the iterations.
- `evaluate`: The central evaluation method for objective and constraint functions. When called with `dmode=0`, `objval` and `constr` must contain the objective and constraints evaluated at `xi` on exit. If `dmode=1`, additionally the constraint Jacobian and objective gradient must be written to `constrJac` and `gradObj`, respectively.

Optionally, the following method can be implemented:

- `reduceConstrVio`: Sometimes, a problem offers possibilities to reduce its infeasibility in a computationally cheap way but the SQP method usually does not have enough information about the problem structure to exploit this. In this case, a feasibility heuristic may be implemented for a problem that is always called before the expensive feasibility restoration phase described in Sec. 6.3.2. A prominent example for this is the feasibility restoration heuristic for DMS problems described in Algorithm 8. On entry, `xi` contains the values of the last accepted iterate. On exit, `xi` is set by the restoration heuristic and it is checked if it is acceptable for the current filter.

9.1.2. Running the algorithm

The SQP method itself is modelled as a class `SQPmethod` which contains as class attribute a pointer to an object of the class `Problemspec`; the problem which is to be solved. Another class attribute is an object of a class `SQPiterate`, which holds all variables that are updated during every SQP iteration.

`SQPmethod` has a class method `run`, which must be called from a suitable driver program. In our case, the driver is implemented in VPLAN, that instantiates and runs the method. When the algorithm is run, it typically produces one line of output for every iteration. The columns of the output are described in Table 9.1.

Remark 9.1. The object-oriented paradigm allows for an elegant implementation of the feasibility restoration phase of Sec. 6.3, where a minimum norm NLP is solved. We implemented a class `RestorationProblem` derived from `ProblemSpec`. When instantiated, it is passed a pointer to the original problem's `Problemspec` object. Using the original problem's `evaluate` method, the implementation of the minimum norm constraints and objective in the `evaluate` method of the `RestorationProblem` is straightforward. Then, we simply instantiate another SQP method within the restoration phase to solve the minimum norm problem.

9.1.3. Default values of algorithmic parameters

In the specification of Algorithm 6, several parameters must be specified. While some of them, such as ε_{opt} and $\varepsilon_{\text{feas}}$ can be easily interpreted and should also be adapted to the problem at hand, the meaning of others is less obvious to the user and they have smaller

Column	Description
`it`	Number of iteration
`qpIt`	Number of QP iterations for the QP that yielded the accepted step
`qpIt2`	Number of QP iterations for the QPs whose solution was rejected
`obj`	Value of objective
`feas`	Infeasibility as in (6.3)
`opt`	Optimality as in (6.3)
`|lgrd|`	Maximum norm of Lagrangian gradient
`|stp|`	Maximum norm of step in primal variables
`|lstp|`	Maximum norm of step in dual variables
`alpha`	Steplength
`nSOCS`	Number of second-order correction steps
`sk`	Number of Hessian blocks where the update has been skipped
`da`	Number of Hessian blocks where the update has been damped
`sca`	Value of sizing factor, averaged over all blocks
`QPr`	Number of QPs whose solution was rejected

Table 9.1.: Columns of the textual output of `blockSQP`.

influence on the overall performance. For these parameters, Table 9.2 reports the default values that have worked well in our numerical tests for a wide range of problems.

9.2. `muse`: A multiple shooting method for optimum experimental design

The methods presented in Chapter 4 for the solution of the OED problem (3.28) are implemented in the software package `muse`. Additionally, optimal control problems of the form (2.4) can also be treated using direct single shooting or direct multiple shooting as described in Chapter 2. `muse` comes as extension module for the C++ software package VPLAN [134] and relies on its data structures and interfaces.

More specifically, the role of `muse` is the following: It takes a user-defined dynamic model and a parameterization of this model as input. From these information, it generates a function that corresponds to the objective and the constraint function of the structured NLP (4.9). If a standard optimal control problem is desired instead of the OED problem, the function corresponds to the NLP (2.13). The resulting function provides structure-exploiting evaluation of the objective and constraints as well as their derivatives at any given point $\xi = (\bar{s}_0^x, \hat{q}_0, \hat{w}_0, s_0^{\mathrm{pe}}, \ldots, \bar{s}_{N^{\mathrm{s}}}^x, \hat{q}_{N^{\mathrm{s}}}, \hat{w}_{N^{\mathrm{s}}}, s_{N^{\mathrm{s}}}^{\mathrm{pe}}, H_1, H_2)$. In particular, it comprises calls to the numerical integration routines to evaluate, e.g., the continuity conditions and their derivatives. The resulting data structure is derived from `blockSQP`'s generic `ProblemSpec` class, but it can also be easily passed to many other nonlinear programming solvers.

In this section, we first give a brief overview of the software package VPLAN and describe

General	
ε_{opt}	10^{-5}
$\varepsilon_{\text{feas}}$	10^{-5}
`zero`	10^{-16}
`inf`	10^{20}
Hessian approximations	
τ_{COL}	10^{4}
ε_{COL}	0.1
ε_{QN}	10^{-14}
ε_{SR1}	10^{-8}
qpOASES	
ε_{LI}	10^{-8}
ε_{NZC}	10^{-8}
`condMax`	10^{14}
`nsMax`	100

Line search	
$j_{\max}$	25
$\beta_{\eta}^{\mathcal{F}}$	10^{-5}
$\beta_{\varphi}^{\mathcal{F}}$	10^{-5}
$\eta_{\min}$	$\max\{\varepsilon_{\text{feas}}, 10^{-5}\}$
$\eta_{\max}$	10^{7}
δ_{η}	1.0
s_{η}	1.1
s_{φ}	2.3
η_{φ}	10^{-4}
κ_{soc}	0.99
κ_{kkt}	0.999

Table 9.2.: Default values of algorithmic parameters for `blockSQP`, grouped according to the different aspects of the SQP algorithm.

how dynamic optimization problems are formulated. Then we present our multiple shooting extension `muse` that sets up a structured NLP from the model formulation and a user-defined parameterization. We highlight the object-oriented structure and important features of the implementation. Finally, two examples from optimum experimental design and optimal control are given to illustrate how `muse` generates NLPs from a VPLAN model.

9.2.1. The software package VPLAN

The software package VPLAN [134] has been developed since the late 1990s as a tool for simulation, parameter estimation, and optimum experimental design of dynamic systems—a virtual laboratory for nonlinear processes. It has been used in numerous applications, e.g., [121, 139, 180, 202], and is also successfully applied in industrial practice.

For the solution of the dynamic systems, there exist interfaces to the variable-order and variable-stepsize BDF methods DAESOL [17] and DAESOL-II [4]. Both methods provide evaluation of first- and second-order sensitivities by means of IND. All required derivatives are evaluated using the software ADIFOR [25, 26] that employs the forward mode of algorithmic differentiation. First results with partial differential equations have been obtained using the finite-element library deal.II [12] in connection with VPLAN [130].

For parameter estimation as outlined in Sec. 3.1 VPLAN provides interfaces to the software modules PARFIT [31, 134] and PAREMERA [130] that implement DMS and a reduced approach for DMS, respectively. So far, optimum experimental design problems are treated

by a single shooting approach. The resulting small-scale NLP is solved by the general purpose SQP method SNOPT [93]. Similarly, optimal control problems for DAE may be formulated.

With the software package `muse`, we provide a new module of VPLAN to treat optimum experimental design and optimal control problems by a novel variant of the DMS method.

9.2.2. Formulating a dynamic model and specifying a parameterization

A model in VPLAN consists of two parts: Files with Fortran code, in which the model equations are coded, and ini-files that contain the parameterizations and further details about the model. VPLAN is called with a master file of a problem, `vplan.ini`, as argument. Then the specified action—simulation, parameter estimation, optimum experimental design, or process optimization—is performed.

Problem functions: Fortran-files
The formulation of a dynamic model of the form (3.28) requires the definition of the following functions:

- The DAE system right-hand side $f(t, y(t), z(t), p, u(t))$ and $g(t, y(t), z(t), p, u(t))$,
- model response functions $h(t, y(t), z(t), p, u_0)$ and functions for prediction of the standard deviation $\sigma(t, x, p, u_0)$,
- path constraints $c^{\mathrm{d}}(t, y(t), z(t), p, u(t))$ and nonlinear constraints involving the time-independent controls u_0, $c^{\mathrm{q}}(u_0)$,
- and functions for multi-point boundary constraints $c_i^{\mathrm{pe}}(t_i, y(t_i), z(t_i), p, u_0, s^{\mathrm{pe}})$.

VPLAN requires that these functions are provided by the user as FORTRAN77 subroutines that can be handled by the algorithmic differentiation tool ADIFOR 2.0 to provide all required derivatives. Each function must be coded in a separate file containing exactly the corresponding subroutine. Other general purpose subroutines can be coded in extra files and may be called by any of the other subroutines.

The tool DOIT (Design Of experiments Initialization Tool) calls ADIFOR 2.0 several times to create the required first- and second-order derivatives of the model equations. Furthermore, it compiles all files and creates a shared library that can be dynamically loaded by VPLAN.

Problem parameterization: ini-files
The second part of the problem formulation in VPLAN is specified within several ASCII text-files that we call *ini-files*. VPLAN follows a multi-experiment paradigm: Every experiment has its own dynamic system, time horizon, and parameterization. However, they share a single parameter vector p. Consequently, for every experiment, there exists an `exp.ini` file that specifies the Fortran files with the model functions and the parameterization according to the DMS method. Within this file, the following data must be provided:

- Time interval $[t_0, t_{\mathrm{f}}]$,

- initial values of the dynamic states,
- grid for parameterization of controls τ^{c},
- multiple shooting grid τ^{s},
- grid of potential measurement points τ^{m},
- grid for evaluation of path constraints τ^{d},
- points t_i for the specification of the multi-point boundary constraints c_i^{pe}.

Furthermore, (initial) values for all design variables q, w, and s^{pe} that result from the parameterization must be given. Note that the definition of every grid may be guided by considerations of numerical efficiency but also by requirements and constraints of the underlying process, e.g. a control may only switch at certain times during the process.

A master file, `vplan.ini`, links all experiments together and also specifies the common parameter vector p. Options for the integrator and the optimization algorithms are also specified in the ini-files. Figure 9.1 summarizes the problem formulation with VPLAN. Details on the functionality of `muse` are provided in the next section.

9.2.3. Program structure of `muse`

`muse` is implemented in the C++ programming language and follows an object-oriented programming paradigm. The aim of `muse` is to provide NLPs of the form (4.9) or (2.13) in a generic form that can be passed to most NLP solvers. To achieve this, we use the `Problemspec` class from `blockSQP` as base class that holds only the information that are necessary for the optimization algorithm. In particular, no reference to the dynamic system or the specific constraint structure is needed in the base class. However, the Hessian block structure that is due to multiple shooting must be communicated to the SQP method to facilitate blockwise Hessian updates. The abstract methods `initialize` and `evaluate` must be implemented in derived classes. We will now sketch how the core method `evaluate` is implemented for OC and OED problems parameterized by DMS.

The class `VplProblem`

To model OC and OED problems with VPLAN, an abstract class `VplProblem` is derived from `Problemspec` that provides access to the VPLAN data structures. Furthermore, it contains (pointers to) objects of an abstract class `ExpEval` that is responsible for the evaluation of all constraints corresponding to one experiment. An excerpt of the class definition reads as:

```
class VplProblem : public Problemspec
{
  /* DATA MEMBERS */
  public:
    Mexperiment* V;        ///< Pointer to central VPLAN data structure
    Matrix       p;        ///< Vector of model parameters
```

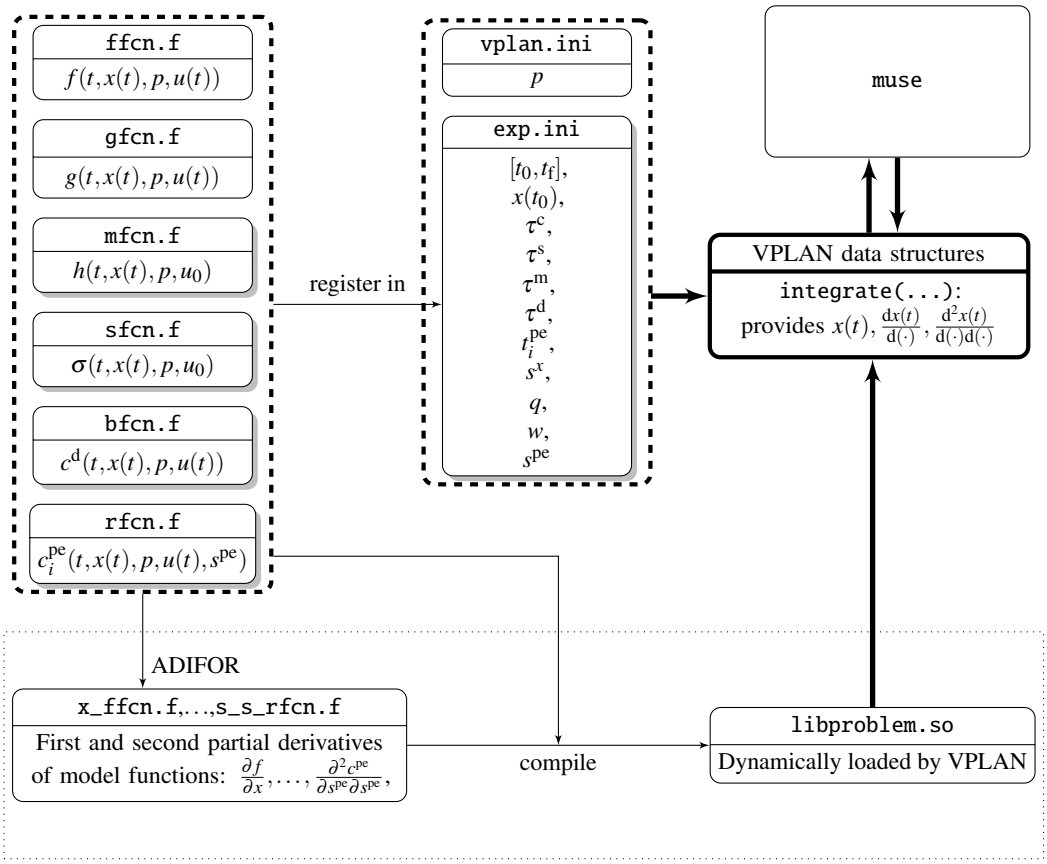


Figure 9.1.: Problem formulation with VPLAN. The dashed boxes correspond to the user interface. Some boxes are denoted by prototypical filenames and have below the mathematical quantities that are defined within these files using the notation from the OED problem (3.28). The dotted box at the bottom corresponds to the scope of the tool DOIT. `muse` relies on the VPLAN data structures to interact with the model. In particular, VPLAN handles calls to the integrator to evaluate states and sensitivities at various times τ.

```
    int           nEx;     ///< Total number of experiments
    int           nExOpt;  ///< Number of experiments to be optimized
    ExpEval**     expEval; ///< Pointers to evaluation objects

  /* METHODS */
  public:
    /** Evaluate problem functions and their derivatives at point xi */
    virtual void evaluate( const Matrix &xi,
                           double *objval, Matrix &constr,
                           Matrix &gradObj, Matrix &constrJac,
                           int dmode, int *info );

  protected:
    /** Evaluate objective, its gradient, and its Hessian */
    virtual void evalObjective( const Matrix &xi, double *objval,
                                Matrix &gradObj, SymMatrix *&hess,
                                int dmode, int *info ) = 0;
};
```

The abstract method `evaluate()` from the parent class `Problemspec` is implemented here in a generic way. Its body reads as

```
void VplProblem::evaluate( const Matrix &xi,
                           double *objval, Matrix &constr,
                           Matrix &gradObj, Matrix &constrJac,
                           int dmode, int *info ) {
  [...]
  #pragma omp parallel for default(shared) private(iShootTotal)
  for( iShootTotal=0; iShootTotal<nShootTotal; iShootTotal++ )
  {
      int i, currExp, iShoot;

      /* Find out the current experiment */
      for( i=0; i<nExOpt; i++ )
          if( iShootTotal < nShootArray[i] ) {
              currExp = i;
              break;
          }

      /* iShoot is the experiment-local shooting node index */
      iShoot = iShootTotal;
      if( currExp > 0 )
          iShoot -= nShootArray[currExp-1];

      /* Evaluate all constraints assigned to one shooting node */
      infos[currExp][iShoot] = 0;
      expEval[currExp]->evalOneShootingNode( iShoot, xi, lambda,
                                             objval, constr,
                                             gradObj, constrJac, hess,
                                             dmode, conflag,
                                            &infos[currExp][iShoot] );
  }
```

```
    [...]

    /* Evaluate objective function */
    evalObjective( xi, objval, gradObj, hess, dmode, info );

    [...]
}
```

The evaluation is now distributed to the individual shooting nodes through the method `evalOneShootingNode` that is a member of the class `ExpEval`. This class is described below. Note that the loop over all shooting nodes is parallelized using OpenMP [53]. Actually, there would be two loops: An outer loop for all experiments and an inner loop for the shooting nodes of this experiment. To facilitate parallelization on both multi-experiment and shooting-node level, we combined both into a single for-loop running over the set of all shooting nodes from all experiments combined. Furthermore, there exists a method `evalObjective` to evaluate the objective function φ. This is abstract in `VplProblem` and must be implemented for all problem classes. We have implemented classes for the following problems:

- single shooting OED problem,
- multiple shooting OED problem,
- single shooting OC problem, and
- multiple shooting OC problem.

For the multiple shooting OED case, there are also the pseudo states formulations available that are described in Sec. 4.5.4 and 4.5.3. Figure 9.2 summarizes the problem specification classes in the form of an inheritance diagram. In each class, the array `ExpEval** expEval` contains pointers to objects of a corresponding derived class of the base class `ExpEval` that implements a sutiable variant of `evalOneShootingNode`.

The class `ExpEval`

The `ExpEval` class and its derived classes have a similar structure as the `VplProblem` class, see Figure 9.3. The parent class `ExpEval` has methods to evaluate the different types of constraints of a problem parameterized by DMS, such as (4.9) or (2.13), for example

- `calcContinuityConstraints` – continuity constraints for nominal states
- `calcVarContinuityConstraints` – continuity constraints for variational states
- `calcConsistencyConstraints` – consistency constraints for nominal algebraic states
- `calcVarConsistencyConstraints` – consistency constraints for variational algebraic states
- `calcPathConstraints` – path constraints

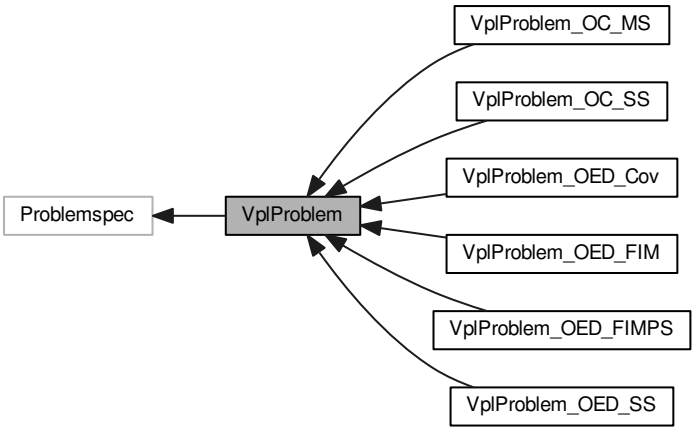


Figure 9.2.: Inheritance diagram for problem specification classes.

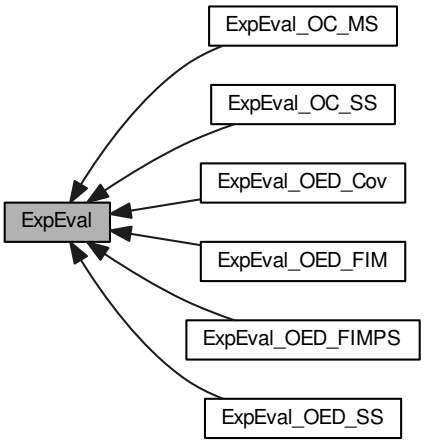


Figure 9.3.: Inheritance diagram for experiment evaluation classes.

- `calcMeasContrib` – contributions to the Fisher information matrix
- etc.

These methods take as input a shooting node index and results from the integrator. Then they write the result of the computation to the proper position of the constraint vector and the constraint Jacobian that are passed through by the NLP solver. This is facilitated by a set of index structures, `**varIdx` and `**conIdx`. They are pointers to integer pointers and are defined as follows:

- `varIdx[j][k]` = position of variable type k from shooting node j in the overall variable vector, where k represents a type, such as s^y, s^z, q, w, etc.
- `conIdx[j][k]` = position of constraint type k from shooting node j in the overall constraint vector, where k represents a constraint type, such as continuity constraints, path constraints, etc.

The class `ExpEval` has two abstract methods, that must be implemented for every derived class, see Fig. 9.3:

- `countVarsAndCons`: This determines the index structures `**varIdx` and `**conIdx`. The challenging part is that all grids are totally independent, so every shooting node may be assigned a different number of variables and constraints. Furthermore, it may be necessary to introduce artificial constraints if, e.g. a shooting node lies in the interior of a control interval. Every problem class (OC and OED, single and multiple shooting) has a different constraint and variable structure.
- `evalOneShootingNode`: This method forms the core of the evaluation. VPLAN's `integrate` function is called here and all constraints assigned to one shooting node are processed by calling a suitable set of the evaluation methods defined in the parent class `ExpEval`. For example, the single shooting problems do not have continuity constraints; OC problems do not need to take into account contributions to the Fisher matrix, etc.

9.2.4. Examples

We present two examples to illustrate the structure of the NLP generated by `muse` from a VPLAN model, one OED and one OC problem.

Optimum experimental design
As an example, let us consider a simple Lotka–Volterra type system as described in [170]. Here, we want to design an experiment to estimate the variables α and β of the following system:

$$\dot{y}_1(t) = \quad y_1(t) - \alpha y_1(t) y_2(t) - 0.4 u(t) y_1(t), \tag{9.1a}$$
$$\dot{y}_2(t) = -y_2(t) + \beta y_1(t) y_2(t) - 0.2 u(t) y_2(t), \tag{9.1b}$$
$$0 \le u(t) \le 1.$$

We need to code only the two ODE right-hand side equations in a Fortran subroutine. In the files `vplan.ini` we define the parameters and set the current estimate to $\alpha = \beta = 1$. In the file `exp.ini`, we set the parameterization grids. We assume that we can observe both states at $\tau^{\mathrm{m}} = (3,6,9,12)^T$ but we have to choose the two best observations for every observable. The shooting grid is $\tau^{\mathrm{s}} = (0,6,12)^T$ and the control grid is $\tau^{\mathrm{c}} = (0,4,6,8,12)$. We refer to the VPLAN documentation for details on how to set up the Fortran- and ini-files.

From these information, `muse` generates an NLP with 21 variables and 11 constraints. Using `muse`'s `printVariables` and `printConstraints` methods, we obtain the following output:

```
<|----- Variables -----|>
  1: exp1 [0] q_u1 [0, 4]        0 <=    0.3 <=        1
  2: exp1 [0] q_u1 [4, 6]        0 <=    0.3 <=        1
  3: exp1 [0] w1 (t=3)           0 <=      1 <=        1
  4: exp1 [0] w2 (t=3)           0 <=      1 <=        1
  5: exp1 [1] s_y(1)        -1e+20 <=  0.503 <=    1e+20
  6: exp1 [1] s_y(2)        -1e+20 <=   1.11 <=    1e+20
  7: exp1 [1] s_yp(1,1)     -1e+20 <=  0.317 <=    1e+20
  8: exp1 [1] s_yp(2,1)     -1e+20 <= -0.266 <=    1e+20
  9: exp1 [1] s_yp(1,2)     -1e+20 <= -0.146 <=    1e+20
 10: exp1 [1] s_yp(2,2)     -1e+20 <= -0.809 <=    1e+20
 11: exp1 [1] q_u1 [6, 8]        0 <=    0.3 <=        1
 12: exp1 [1] q_u1 [8, 12]       0 <=    0.3 <=        1
 13: exp1 [1] w1 (t=6)           0 <=      1 <=        1
 14: exp1 [1] w2 (t=6)           0 <=      1 <=        1
 15: exp1 [1] w1 (t=9)           0 <=      1 <=        1
 16: exp1 [1] w2 (t=9)           0 <=      1 <=        1
 17: exp1 [1] w1 (t=12)          0 <=      1 <=        1
 18: exp1 [1] w2 (t=12)          0 <=      1 <=        1
 19: [linC] H(p1,p1)             0 <=   5.33 <=    1e+20
 20: [linC] H(p2,p1)        -1e+20 <=   2.62 <=    1e+20
 21: [linC] H(p2,p2)             0 <=   9.96 <=    1e+20

<|----- Constraints -----|>
  1: exp1 [0] cont y(1)            0 <=    0 <=     0
  2: exp1 [0] cont y(2)            0 <=    0 <=     0
  3: exp1 [0] cont yp(1,1)         0 <=    0 <=     0
  4: exp1 [0] cont yp(2,1)         0 <=    0 <=     0
  5: exp1 [0] cont yp(1,2)         0 <=    0 <=     0
  6: exp1 [0] cont yp(2,2)         0 <=    0 <=     0
  7: exp1 [lin] meas/fun (mfcn1)   2 <=    4 <=     2
  8: exp1 [lin] meas/fun (mfcn2)   2 <=    4 <=     2
  9: [linC] cc H(p1,p1)            0 <=    0 <=     0
 10: [linC] cc H(p2,p1)            0 <=    0 <=     0
 11: [linC] cc H(p2,p2)            0 <=    0 <=     0
```

The variables and constraints are shown with their initial values and their lower and upper bounds. Note that shooting variables and continuity constraints for the two variational states are created automatically, i.e. we do not have to code the VDAE. Note also that we do not have shooting variables for the first and the last node, as they would enter the whole problem

only linearly and can thus be omitted to reduce costs for the linear algebra. Consequently, there are only continuity constraints for the node $\tau_1^s = 6$. The values for all shooting variables (5-10) are set such that the continuity constraints are satisfied. Constraints 7 and 8 are linear constraints for the maximum number of measurements. Constraints 9-11 are the $n_p(n_p+1)/2$ linearly coupled constraints for the Fisher information matrix.

Optimal control

As an optimal control example, we consider a variant of problem (9.1) described in [172]. The goal is to choose an optimal control to bring both the predator and prey states to a prescribed steady state. We augment Eqs. (9.1) by a Lagrange term that describes the distance to the steady state and obtain:

$$\dot{y}_1(t) = \quad y_1(t) - \alpha y_1(t) y_2(t) - 0.4 u(t) y_1(t), \tag{9.2a}$$

$$\dot{y}_2(t) = -\ y_2(t) + \beta y_1(t) y_2(t) - 0.2 u(t) y_2(t), \tag{9.2b}$$

$$\dot{y}_3(t) = \quad (y_1(t)-1)^2 + (y_2(t)-1)^2, \tag{9.2c}$$

$$0 \le u(t) \le 1.$$

Now we choose the shooting grid as $\tau^s = (0,3,6,9,12)^T$ and the control grid as $\tau^c = (0,4,6,8,12)$. The generated NLP has 19 variables and 15 equality constraints:

```
<|----- Variables -----|>
  1: exp1 [0] +q_u1 [0, 3]    -1e+20 <=    0.3 <=   1e+20
  2: exp1 [1] s_y(1)          -1e+20 <=   1.79 <=   1e+20
  3: exp1 [1] s_y(2)          -1e+20 <=  0.553 <=   1e+20
  4: exp1 [1] s_y(3)               0 <=   1.38 <=    1000
  5: exp1 [1] q_u1 [3, 4]          0 <=    0.3 <=       1
  6: exp1 [1] q_u1 [4, 6]          0 <=    0.3 <=       1
  7: exp1 [2] s_y(1)          -1e+20 <=  0.503 <=   1e+20
  8: exp1 [2] s_y(2)          -1e+20 <=   1.11 <=   1e+20
  9: exp1 [2] s_y(3)               0 <=   3.22 <=    1000
 10: exp1 [2] q_u1 [6, 8]          0 <=    0.3 <=       1
 11: exp1 [2] +q_u1 [8, 9]   -1e+20 <=    0.3 <=   1e+20
 12: exp1 [3] s_y(1)          -1e+20 <=   1.27 <=   1e+20
 13: exp1 [3] s_y(2)          -1e+20 <=  0.379 <=   1e+20
 14: exp1 [3] s_y(3)               0 <=   4.33 <=    1000
 15: exp1 [3] q_u1 [9, 12]         0 <=    0.3 <=       1
 16: exp1 [4] s_y(1)          -1e+20 <=  0.739 <=   1e+20
 17: exp1 [4] s_y(2)          -1e+20 <=   1.61 <=   1e+20
 18: exp1 [4] s_y(3)               0 <=   6.41 <=    1000
 19: exp1 [4] +q_u1 [12, 12] -1e+20 <=    0.3 <=   1e+20

<|----- Constraints -----|>
  1: exp1 [0] cont y(1)             0 <=      0 <=       0
  2: exp1 [0] cont y(2)             0 <=      0 <=       0
  3: exp1 [0] cont y(3)             0 <=      0 <=       0
  4: exp1 [1] cont y(1)             0 <=      0 <=       0
  5: exp1 [1] cont y(2)             0 <=      0 <=       0
  6: exp1 [1] cont y(3)             0 <=      0 <=       0
  7: exp1 [2] cont y(1)             0 <=      0 <=       0
```

```
  8: exp1 [2] cont y(2)              0 <=       0 <=        0
  9: exp1 [2] cont y(3)              0 <=       0 <=        0
 10: exp1 [3] cont y(1)              0 <=       0 <=        0
 11: exp1 [3] cont y(2)              0 <=       0 <=        0
 12: exp1 [3] cont y(3)              0 <=       0 <=        0
 13: exp1 [lin] cont+ u1 (node 1)    0 <=       0 <=        0
 14: exp1 [lin] cont+ u1 (node 3)    0 <=       0 <=        0
 15: exp1 [lin] cont+ u1 (node 4)    0 <=       0 <=        0
```

Note that the two shooting nodes at $t = 3$ and $t = 9$ are not part of the control grid τ^c. There, muse introduces the additional variables 1 and 11 (+q_u1), and linear constraints 13 and 14 to maintain separability but also ensure equivalence to the originally selected parameterization. Furthermore, another artificial variable was introduced at t_f. This is to facilitate the computation of exact Hessians of the objective. By explicitly maintaining variables $s^x_{N^s}$ and $\hat{q}_{N^s}$ we avoid computing second-order sensitivities of the states when the objective Hessian is evaluated.

Chapter 10.

Performance of blockSQP on benchmark collection

In this chapter, we evaluate the performance of different variants of our SQP implementation blockSQP on a range of NLPs arising from DMS for optimum experimental design and optimal control. We are interested in how fast and reliable blockSQP can find a local solution of a given NLP. In particular, we evaluate different Hessian approximation sequences and scaling strategies discussed in Chapter 7. We show that a strategy based on switching between an SR1 update and a BFGS update with selective sizing yields the best overall performance in terms of reliability, number of SQP iterations, and CPU time. This confirms our results from Section 5.3 where we showed for a model problem that indefinite Hessian approximations are necessary for fast local convergence. A comparison with the popular SQP solver SNOPT7 [93] shows that blockSQP is superior for problems arising in DMS in terms of SQP iterations and CPU time.

10.1. Test problems and algorithmic parameters

We first introduce our test set of OED and OC problems from the literature. Then we describe the multiple shooting parameterization settings and give an overview over algorithmic settings used. Finally, we explain *performance profiles*, a tool introduced in [61] for evaluating the performance of competing algorithms on a benchmark set.

10.1.1. Test problems

In total, we test blockSQP on 6 OED examples and 10 OC examples. Most of them are taken from the literature. The equations for the ocean and fermenter examples are given in the appendix.

OC problems
From the COPS 3.0 test set [60], we consider the optimal control problems 4 (hanging chain), 9 (particle steering), 10 (goddard rocket), 11 (hang glider), and 14 (catalyst mixing). The Lotka–Volterra optimal control model is described in [172], the Williams-Otto semi-batch reactor in [82], and the batch distillation process in [57]. Both the fermenter and ocean problems are found in the MUSCOD-II collection of optimal control problems [34, 142].

Their equations are given in Appendix A. Table 10.1 summarizes the number of states, controls, and path constraints of the models.

Problem name	n_x	n_u	n_d
oc-batchdist	13	2	22
oc-catalyst-mixing	4	1	0
oc-goddard	3	2	4
oc-hangglider	4	2	2
oc-hanging-chain	3	1	0
oc-particle-steering	5	2	0
oc-fermenter	10	3	19
oc-lotka	3	2	0
oc-ocean	3	2	4
oc-williams-otto	9	2	3

Table 10.1.: Characteristics of OC test problems. Shown are the number of states, control functions, and nonlinear dynamic constraints.

OED problems

We use the catalyst mixing model from the COPS test set and formulate it as an OED problem with the frequency factor of x_2 as uncertain parameter and both states as observables. The continuous stirred-tank reactor (CSTR) model is from [65] and is discussed in the context of OED in [123]. The Lotka–Volterra OED problem is taken from [170]. The polymerization example is described in [205]. The Urethane reaction was first described as OED problem in [134]. The baker's yeast problem appears in a number of publications on OED, e.g., [85]. Table 10.2 lists the number of states, parameters, controls, observables, and path constraints for the models. All OED problems are optimized with regard to the A-criterion.

Problem name	n_x	n_p	n_u	n_h	n_d
oed-catalyst-mixing	2	1	1	2	0
oed-cstr	2	2	2	2	4
oed-lotka	2	2	1	2	0
oed-polymerization	9	4	3	4	0
oed-urethane	6	6	10	4	0
oed-yeast	2	4	3	2	0

Table 10.2.: Characteristics of OED test problems. Shown are the number of states, parameters, control functions, observables, and nonlinear dynamic constraints.

10.1.2. Problem discretization

We discretize the problems using equdistant grids $\tau^c = \tau^s = \tau^d = \tau^m$. Here, we choose the same grid for the controls, multiple shooting, measurements, and path constraints. The case with different grids will be investigated in Chapter 12. The controls are approximated by piecewise constant functions. The number of intervals and the size of the resulting NLPs are given in Table 10.3. For the first iterate $x^{[0]}$, we chose all shooting variables such that the continuity conditions are satisfied. For the OED problems, we set all $w_i = 1$. The starting values of the control variables and the parameter values are given in Appendix B.

Problem name	N^s	variables	constraints
`oed-catalyst-mixing`	64	447	255
`oed-cstr`	16	159	95
`oed-lotka`	64	575	383
`oed-polymerization`	4	167	147
`oed-urethane`	16	862	781
`oed-yeast`	16	225	176
`oc-batchdist`	64	1092	1027
`oc-cops-catalyst-mixing`	64	257	193
`oc-cops-goddard`	64	322	258
`oc-cops-hangglider`	64	386	324
`oc-cops-hanging-chain`	64	257	195
`oc-cops-particle-steering`	64	450	388
`oc-fermenter`	64	900	708
`oc-lotka`	64	322	257
`oc-ocean`	64	322	194
`oc-williams-otto`	64	706	578

Table 10.3.: Number of variables and constraints for benchmark NLPs.

10.1.3. Algorithmic settings

All problems are implemented in VPLAN using `muse` to generate the NLPs. The integration tolerances of DAESOL are set between 10^{-7} and 10^{-9}.

In `blockSQP`, we use limited memory Hessian updates with a memory size of $\tilde{M} = 20$. This proved to be enough for the asymptotic properties of the updates to kick in while still allowing old and hence irrelevant information to be forgotten. For the OED problems, we use the exact second derivative of the A-criterion in the lower right block of the Hessian as described in Sec. 4.3.2.

We declare optimality if the KKT tolerance (6.3) is satisfied with $\varepsilon_{\text{opt}} = \varepsilon_{\text{feas}} = 10^{-5}$. The SQP iteration limit was set to 500 and the maximum number of QP iterations (active set changes) per SQP iteration was set to 2500.

10.1.4. Performance profiles

Performance profiles were introduced by [61] as a tool for benchmarking and comparing optimization software. First, a performance index is selected, which is often the CPU time needed to solve a problem. However, as we are particularly interested in the local convergence properties of the algorithms, the number of SQP iterations provides a more meaningful message. Furthermore, in OED for large dynamical systems, as they appear in industrial applications, the overall computational cost is dominated by the cost of problem linearization, which is usually done once per iteration. This means that it is reasonable to concentrate on the number of SQP iterations. We define:

$$\text{perf}_{\wp,s} := \text{number of SQP iterations of solver } s \text{ for problem } \wp.$$

To compare different solvers from a set S, we compare the performance of a solver s on a problem $\wp$ with the best performance of all the solvers on problem $\wp$. We define the performance ratio $r_{\wp,s}$:

$$r_{\wp,s} := \frac{\text{perf}_{\wp,s}}{\min\left\{\text{perf}_{\wp,s} \mid s \in S\right\}}$$

The performance profile for a solver is the cumulative distribution function for a performance metric. Suppose $\mathcal{P}$ is a set of problems. If we define

$$\rho_s(T) := \frac{\left|\left\{\wp \in \mathcal{P} \mid r_{\wp,s} \leq T\right\}\right|}{|\mathcal{P}|}.$$

then $\rho_s(T)$ is the probability for solver $s \in S$ that a performance ratio $r_{\wp,s}$ is within a factor $T \in \mathbb{R}$ of the best possible performance ratio. Usually, solvers with large probabilities $\rho_s(T)$ are to be preferred. In particular, the value $\rho_s(1)$ denotes the fraction of problems for which solver s yields the best performance.

10.1.5. Computing environment

All results were obtained on a workstation with two Intel® Xeon® E5645 hexacore CPUs (2.4 GHz) and 32 GB memory running Ubuntu 14.04. VPLAN, `muse`, `blockSQP`, and the problem functions were compiled using the GCC 4.8.2 compiler collection with the `-O3` compiler flag set. The maximum number of parallel threads was set to 16.

10.2. Comparison of Hessian scaling strategies

First, we evaluate the different scaling strategies introduced in Sections 7.2.4 and 7.3 for BFGS and SR1 updates:

1. Scaling $B^{[0]}$ by $\sigma_{\text{SP}}^{[k]}$ (Shanno–Phua),

2. scaling $B^{[0]}$ by $\sigma_{\mathrm{OL}}^{[k]}$ (Oren–Luenberger),

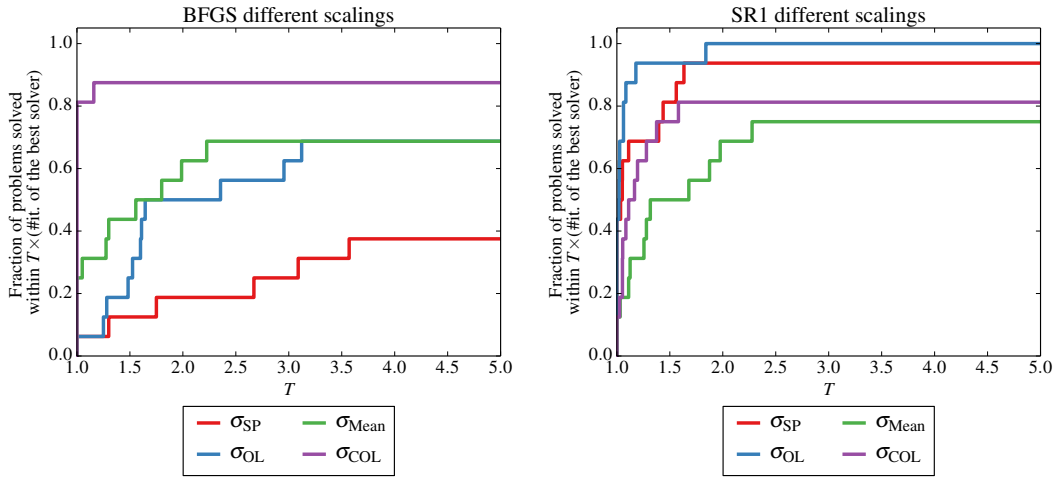
3. scaling $B^{[0]}$ by $\sigma_{\mathrm{Mean}}^{[k]}$ (geometric mean of σ_{SP} and σ_{OL}),

4. scaling $B^{[k]}$ for $k \geq 0$ by $\sigma_{\mathrm{COL}}^{[k]}$ if $0 < \varepsilon_{\mathrm{COL}} < \sigma_{\mathrm{COL}}^{[k]} < 1$ (selective sizing strategy).

Recall that the strategies 1–3 scale only the initial approximation $B^{[0]}$, while the selective sizing strategy attempts to scale every approximation $B^{[k]}$ before it is updated. Figure 10.1a compares the options in form of a performance profile for the block-BFGS update.

We see that the selective sizing strategy clearly dominates the other strategies that scale only the initial Hessian. In the remaining computations, we use it as the fallback approximation $B^{[k,l_{\max}]}$ in Step 2 of Algorithm 6.

To evaluate scaling options for SR1 updates, we use SR1 whenever possible and use BFGS with selective sizing as fallback strategy, i.e. $B^{[k,0]} = B_{\mathrm{SR1}}^{[k]}$ and $B^{[k,1]} = B_{\mathrm{BFGS}}^{[k]}$. Figure 10.1b shows that the selective sizing strategy is not the best choice for the SR1 updates. In the subsequent numerical experiments, we chose the Oren-Luenberger scaling factor σ_{OL} for the initial SR1 approximations.



(a) Performance profile comparing the number of SQP iterations for different scaling strategies for block-BFGS.

(b) Performance profile comparing the number of SQP iterations for different scaling strategies for block-SR1. The fallback strategy uses BFGS updates with selective sizing.

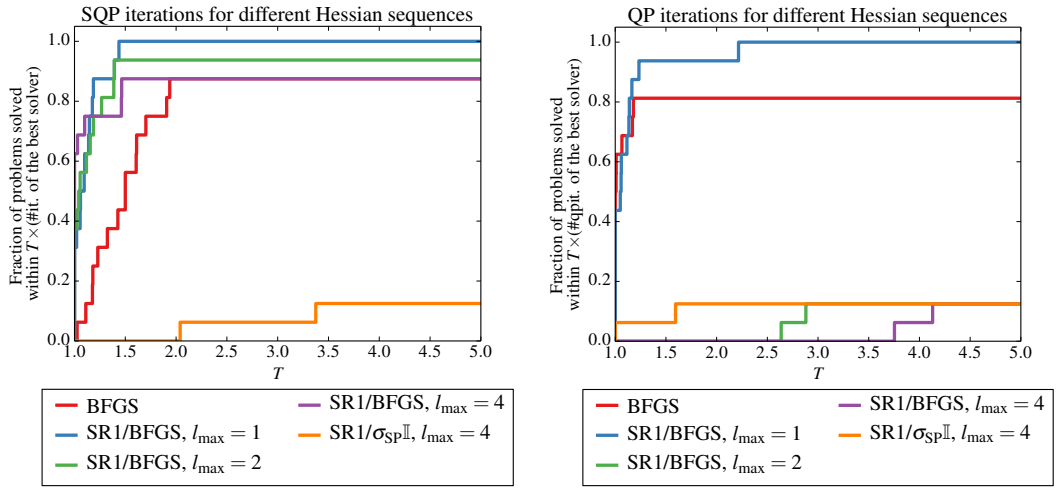
Figure 10.1.: Performance profiles comparing different scaling strategies

10.3. Comparison of Hessian approximation sequences

Next, we compare several Hessian approximation sequences. Motivated by the observations in Section 5.3, we want to include meaningful negative curvature as often as possible. On the

other hand, we need to ensure that Assumption 6.1 is eventually satisfied and we want to avoid solving too many quadratic subproblems in every iteration. We compare the performance for five different strategies:

1. BFGS only, $l_{\max} = 0$,
2. convex combination SR1 and BFGS, $l_{\max} = 1$,
3. convex combination SR1 and BFGS, $l_{\max} = 2$,
4. convex combination SR1 and BFGS, $l_{\max} = 4$,
5. convex combination of SR1 and $\sigma_{\text{OL}}\mathbb{I}$, $l_{\max} = 4$.



(a) Performance profile comparing the number of SQP iterations for different Hessian approximation sequences.

(b) Performance profile comparing the number of QP iterations for different Hessian approximation sequences.

Figure 10.2.: Performance profiles comparing different Hessian approximation sequences

Figure 10.2a shows the performance profile for the number of SQP iterations for all options. We see that strategies 2–4 are superior to the variant where only BFGS updates are used, meaning more problems are solved in fewer iterations. Option 5 is not competitive at all and we made similar experience with other strategies based on a convex combination of SR1 and a scaled identity. We also notice that strategy 4 scores the most wins, being the best method in terms of SQP iterations for more than 60% of the problems. However, strategies 3 and 4 require far too many QP iterations which makes them too slow in practice. Figure 10.2b shows the performance profile for the number of QP iterations. As we expect, the pure BFGS version of `blockSQP` requires comparably few QP iterations for most problems because only one QP is solved per iteration. Strategies 3 and 4, where sometimes three or more QPs must

be solved per iteration, are not competitive at all. Strategy 2 is also very competitive with the pure BFGS version in terms of QP iterations, making it the best method overall.

Table 10.4 shows the results for SQP iterations, QP iterations, and CPU time. We note that the SR1-BFGS method requires fewer SQP iterations on all 16 problems and is faster in terms of CPU time on 14 problems. Interestingly, SR1 updates are accepted relatively rarely by the globalization strategy. Figure 10.3 shows in which particular iterations the SR1 updates are accepted. This is mostly the case during the last iterations, when the active set has settled and the exact Hessian is positive definite in the null space of active constraints. However, if they are accepted, SR1 updates provide rapid local convergence and repair the undesirable behavior of the block-BFGS updates pointed out in Section 5.3.

Instance	SR1-BFGS			BFGS		
	SQP	QP	CPU[s]	SQP	QP	CPU[s]
oed-catalyst-mixing	**39** (3)	557	**1.77**	45	**484**	2.26
oed-cstr	**39** (3)	517	1.28	40	**453**	**1.22**
oed-lotka	**111** (7)	2167	**9.93**	131	**1896**	9.99
oed-polymerization	**30**(11)	466	**4.41**	45	**427**	7.52
oed-urethane	**49** (4)	**676**	**22.38**	59	787	25.79
oed-yeast	**92** (7)	**2026**	**7.94**	143	2312	11.53
oc-batchdist	**87** (5)	**700**	**15.37**	27†	1417†	15.5†
oc-catalyst-mixing	**46** (7)	**1318**	**2.73**	62	3582	4.89
oc-goddard	**41** (6)	371	**1.49**	58	**300**	1.81
oc-hangglider	**44** (6)	**600**	**2.53**	136†	1257†	15.41†
oc-hanging-chain	**86** (7)	700	**2.65**	107	**548**	2.94
oc-particle-steering	**19** (2)	338	1.27	20	**321**	**1.26**
oc-fermenter	**51** (3)	**700**	**5.65**	82	744	7.72
oc-lotka	**17** (3)	444	**0.82**	20	**355**	0.82
oc-ocean	**34** (5)	3764	**3.51**	51	**3359**	3.96
oc-williams-otto	**36** (6)	**1690**	**6.57**	45	1692	6.86

Table 10.4.: SQP iterations, QP iterations, and run times for blockSQP with SR1-BFGS and for pure BFGS updates. Listed in parentheses is the number of SQP iterations in which the SR1 update was accepted by the inertia controlling strategy. Boldface numbers indicate the algorithm with better performance in the respective category. Failures are marked by †.

The reason why strategy 2 needs only little more QP iteration than the pure BFGS strategy is that we can terminate the QP solver early whenever the initial active set has the wrong inertia, cf. Sec. 8.4.2. Table 10.5 shows that this technique saves many QP iterations. For convex combinations of SR1 and BFGS with $0 < \mu < 1$, this was less often the case, causing the strategies with $l_{\max} > 1$ to be less efficient overall.

Instance	with QP early term.		w/o QP early term.	
	QP	CPU[s]	QP	CPU[s]
oed-catalyst-mixing	557	1.77	5562	8.02
oed-cstr	517	1.28	9454	5.53
oed-lotka	2167	9.93	35981	89.85
oed-polymerization	466	4.41	1602	7.27
oed-urethane	676	22.38	67201	605.74
oed-yeast	2026	7.94	125150	123.54
oc-batchdist	700	15.37	26826	276.99
oc-catalyst-mixing	1318	2.73	4761	6.53
oc-goddard	371	1.49	7314	10.17
oc-hangglider	600	2.53	27034	44.09
oc-hanging-chain	700	2.65	22414	20.95
oc-particle-steering	338	1.27	4641	13.11
oc-fermenter	700	5.65	18414	115.19
oc-lotka	444	0.82	1962	2.76
oc-ocean	3764	3.51	16861	13.23
oc-williams-otto	1690	6.57	17583	57.25

Table 10.5.: Total number of QP iterations and overall computing times of SR1-BFGS blockSQP ($l_{\max} = 1$) with and without early termination of the QP solution. The number of SQP iterations is not affected.

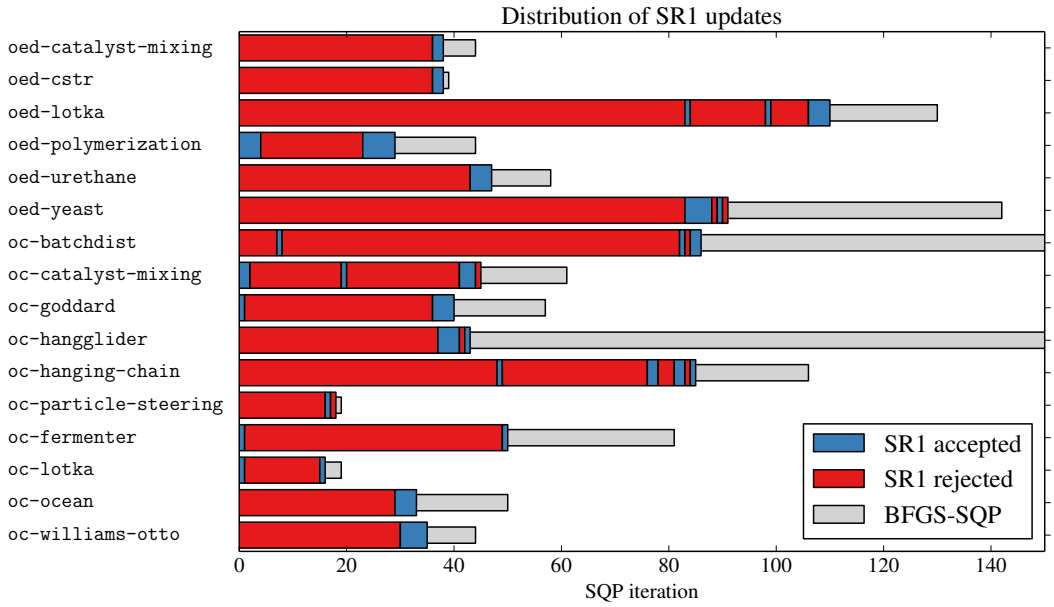


Figure 10.3.: Distribution of accepted and rejected SR1 updates for `blockSQP` with SR1-BFGS. The number of iterations for `blockSQP` using only BFGS are given for comparison.

We conclude that `blockSQP` with a combination of SR1 and selectively sized BFGS updates is a fast and reliable algorithm for problems arising in DMS and we will use this algorithm variant for the computations in the remainder of this thesis. In practical situations, where dynamic systems are larger and more complicated than those considered here, the linearization is expected to dominate the computation time compared to the solution of the QPs. There, we expect `blockSQP` to be particularly efficient and clearly superior in terms of CPU time.

10.4. Comparison with SNOPT

Finally, we compare `blockSQP` to the state-of-the-art SQP method SNOPT7 [93] for our problem test set. It uses a line search based on an augmented Lagrangian merit function to promote global convergence and the primal active set method SQOPT to solve the quadratic subproblems. A limited memory BFGS update is used that gives rise to convex QPs. Although in general sparsity is exploited, SNOPT does not take into account the block-diagonal structure of the Hessian arising in DMS. Table 10.6 compares the number of SQP iterations to those of `blockSQP`.

We notice that SNOPT fails to find a solution to four out of six OED problems. These four examples describe chemical reactions and involve highly nonlinear ODE systems, whereas the two other problems, `oed-lotka` and `oed-catalyst-mixing`, contain only

mild nonlinearities in the right-hand side of the ODE. We conclude that a general purpose SQP method is insufficient for practical OED problems parameterized by DMS and that a dedicated, structure-exploiting SQP method is required to solve this class of problems.

Instance	blockSQP	SNOPT7
oed-catalyst-mixing	39	**36**
oed-cstr	**39**	159†
oed-lotka	**111**	371
oed-polymerization	**30**	6†
oed-urethane	**49**	842†
oed-yeast	**92**	10†
oc-batchdist	**87**	10†
oc-catalyst-mixing	46	**43**
oc-goddard	41	**34**
oc-hangglider	**44**	151
oc-hanging-chain	**86**	151
oc-particle-steering	**19**	169
oc-fermenter	51	**34**
oc-lotka	**17**	24
oc-ocean	**34**	340†
oc-williams-otto	**36**	78

Table 10.6.: Number of SQP iterations for blockSQP and SNOPT7. Boldface numbers indicate better performance of the respective method. Failures are marked by †.

Chapter 11.

Optimum experimental design case studies

In this chapter we illustrate the capabilities of the newly developed methods by two example applications from chemical engineering. First, we study a continuous stirred tank reactor that is operated under restrictive process constraints. We investigate optimal solutions with respect to the A-criterion defined on the covariance matrix and the exp-criterion defined on the Fisher matrix. The second example is the Urethane reaction from [134] that has a highly nonlinear dynamic. For both examples, our newly implemented methods are significantly better in terms of SQP iterations and computing time compared to the existing single shooting implementation of VPLAN.

11.1. A continuous stirred-tank reactor

11.1.1. Model

The first example is a continuous stirred-tank reactor (CSTR) introduced in [65] and formulated as OED problem in [123]. An exothermic reaction of $c(\cdot)$ takes place in a liquid with feed $u_1(\cdot)$, and is controlled by external regulation $u_2(\cdot)$ of the temperature $T(\cdot)$. The state variables are formulated as follows:

$$\dot{c}(t) = \frac{F_{\text{in}}u_1(t) - F_{\text{out}}c(t)}{A_{\text{r}}L} - k_{\text{r},0}\exp\left(-\frac{E}{RT(t)}\right)c(t) \tag{11.1a}$$

$$\dot{T}(t) = \frac{F_{\text{in}}T_{\text{in}} - F_{\text{out}}T(t)}{A_{\text{r}}L} - \frac{\Delta H_{\text{r}}}{\rho C_{\text{p}}}k_{\text{r},0}\exp\left(-\frac{E}{RT(t)}\right)c(t) + \frac{2U}{r\rho C_{\text{p}}}\left(u_2(t) - T(t)\right) \tag{11.1b}$$

The OED task is to minimize the uncertainty of the frequency factor $k_{\text{r},0}$ and the heat transfer coefficient U using the controls u_1 and u_2 and 4 observations of each $c(\cdot)$ and $T(\cdot)$ during an experiment time of 20 minutes. We assume that the level of the liquid is perfectly controlled, hence $F_{\text{in}} = F_{\text{out}}$. The process constraints are

$$0.8\text{mol/l} \leq c(t) \leq 1\text{mol/l}, \tag{11.2a}$$

$$298\text{K} \leq T(t) \leq 333\text{K}, \tag{11.2b}$$

and the control constraints are

$$0.8\text{mol/l} \leq u_1(t) \leq 1\text{mol/l}, \tag{11.3a}$$

$$288\text{K} \leq u_2 \leq 353\text{K}. \tag{11.3b}$$

Table 11.1 lists all units and values for the CSTR model.

Sym.	Value	Unit	Sym.	Value	Unit
$c(t)$		mol/l	r	2.19	dm
$T(t)$		K	E	72740	J/mol
$u_1(t)$		mol/l	ρ	1	kg/l
$u_2(t)$		K	C_p	239	J/(kg K)
L	6.6	dm	ΔH_r	$-5 \cdot 10^4$	J/mol
F_in	100	l/min	U	549.36	J/(min dm^2 K)
T_in	350	K	$k_{\mathrm{r},0}$	$7.2 \cdot 10^{10}$	1/min

Table 11.1.: State and control units and parameter values and units for the CSTR model.

Parameterization and starting values
For the control, measurement and path constraint grids, we divide the time horizon into 64 equidistant intervals, hence

$$\tau_j^\mathrm{c} = \tau_j^\mathrm{d} = \tau_j^\mathrm{m} = 20 \cdot \frac{j}{64}, \quad j = 0, \ldots, 64.$$

The controls are approximated on the grid τ^c by piecewise constant functions, which means that the control constraints (11.3) are linear. The multiple shooting grid consists of 33 nodes which are chosen such that

$$\tau_j^\mathrm{s} = \tau_{2j}^\mathrm{d}, \quad j = 0, \ldots, 32.$$

Note that half of the path constraint (11.2) are now simple bounds for the shooting variables, while the other half are added as nonlinear inequality constraints to the NLP. The resulting NLP has a total of 444 variables and 257 constraints, 255 of which are nonlinear.

The initial values of the states are fixed at $x_1(0) = 0.877$ and $x_2(0) = 323$. As starting point for the optimization we set the controls to $u_1(t) = 0.9$ and $u_2(t) = 300$. All 130 measurement weights are set to 1. The values for the shooting variables are obtained by forward integration. The parameters $k_{\mathrm{r},0}$ and U are both scaled to 1. The absolute and relative integrator tolerances are set to 10^{-9}.

11.1.2. *A*-optimal design

First, we compute an OED with respect to the A-criterion, that means we minimize the average variance of the parameters. With the configuration given above, the standard deviation of the parameters are 33.56% for $k_{\mathrm{r},0}$ and 12.97% for U, which corresponds to an A-criterion objective value of $\Phi_A = 0.06473$. Note, however, that this design is infeasible because all 130 potential measurements are selected, but only eight are allowed.

When we run the optimization, `blockSQP` converges in 46 iterations taking 6.7 seconds. Figure 11.1 depicts the optimal controls and the corresponding states. The standard deviation

of the parameters at the optimum is 3.86% for $k_{\mathrm{r},0}$ and 2.37% for U, yielding an objective value of $\Phi_A = 0.001027$.

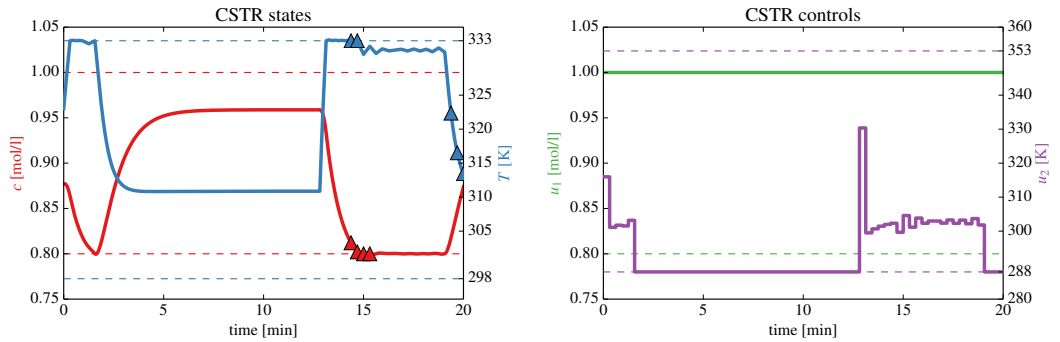


Figure 11.1.: Optimal states and controls with respect to the A-criterion for the CSTR example. Dashed lines mark state and control constraints. Triangles mark optimal measurement placements. The four measurements for $c(t)$ have integral weights equal to one, for $T(t)$, three weights are integral. The remaining measurement is split between two weights at $t = 14.6875$ ($w^2_{49} = 0.51$) and $t = 20$ ($w^2_{64} = 0.49$).

11.1.3. Fisher matrix optimization criterion

Next, we investigate a new OED criterion based on the Fisher matrix that was introduced in [120]. Here, the idea is to avoid the computation of the covariance matrix, which can be a problem if the model is poorly identified, and optimize a criterion on the Fisher matrix instead. The criterion reads as follows:

$$\Phi_{\mathrm{exp}}(H) = \frac{1}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} \exp(-a_i \cdot H_{ii}), \tag{11.4}$$

where H_{ii} are the diagonal elements of the Fisher matrix and a_i are suitable scaling factors. Note that in one dimension, $\Phi_{\mathrm{exp}}(H) = \exp(-a \cdot H)$ has a similar shape as the classical criteria on the covariance matrix, $\Phi(C) = 1/H$, except that it is defined for $H = 0$.

In our experiment, we chose $a_i = 1/H^{[0]}_{ii}$, i.e. the diagonal elements of the Fisher matrix for the initial design. With the same starting configuration as above, `blockSQP` needed 63 iterations and 12.7 seconds to converge. Figure 11.2 shows the optimal solution for this choice. However, we noted in our experiments that the solution strongly depends on the choice of the scaling factors. In the plot, we see that the maximum number of measurements is already exhausted after about 6 minutes, before the end of the experiment. Then the objective is independent of the choice of the controls after the last measurement. Although in general this may cause numerical problems, we found in our experiments that our algorithm coped quite well with these situations. The optimal solution depicted in Fig. 11.2 yields standard

deviations of only 25.52% for $k_{r,0}$ and 15.88% for U. When we start the optimization with respect to the A-criterion in the solution point, the standard deviations can be reduced to 4.46% and 2.75% which is close to the optimum computed above. Hence, we recommend using the A criterion if the covariance matrix is available because then the new criterion does not provide any numerical benefits.

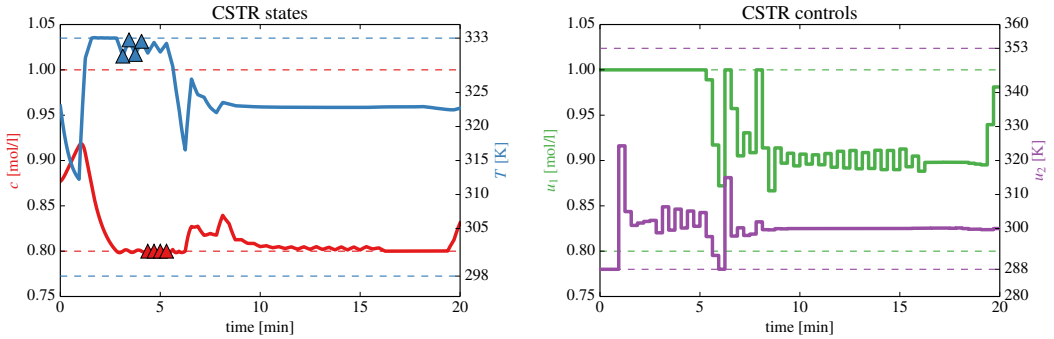


Figure 11.2.: Optimal states and controls with respect to the exp-criterion for the CSTR example. Dashed lines mark state and control constraints. Triangles mark optimal measurement placements. Note that all measurements weights are integral.

11.1.4. Comparison with existing implementation

We now compare our new method with the existing OED implementation in VPLAN. We optimize with respect to the A-criterion using a single shooting approach and SNOPT to solve the NLP. The single shooting NLP has 258 variables and 131 constraints (path and measurement constraints). We note that the method finds a different local minimum, yielding standard deviations of 4.32% for $k_{r,0}$ and 3.69% for U, which is slightly higher than those found by `blockSQP` and `muse`.

The existing implementation requires 66 SQP major iterations but takes 156.1 seconds. This is a factor of more than 23 compared to the 6.7 seconds computing time of the new method! The reasons for this are the following:

- **Sensitivity load:** We compute all sensitivities using the forward mode of AD. That means for the single shooting objective gradient we need to solve a forward variational ODE over the whole time interval for *every* control discretization variable q. Although the reverse mode of AD allows a considerably cheaper computation of the objective gradient, the presence of many path constraints limits its efficiency. This is because the constraint Jacobian for the path constraints of dimension 128×128 needs to be evaluated. In contrast, the DMS method only requires the solution of variational systems for the local control variables on every shooting interval. Here, one multiple shooting interval comprises two control intervals, i.e., a total of four control variables.

- **Parallelization:** The sensitivities are evaluated in parallel on the multiple shooting intervals. However, evaluating the sensitivities sequentially yields a computing time of 14.8 seconds which is still more than ten times faster than the existing implementation. Here, the parallelization on 16 threads improves the runtime only by a factor of about 2.2 as considerable time is spent for the solution of the individual QPs, which is not parallelized.

11.2. The Urethane reaction

11.2.1. Model

Our second OED case study is the Urethane reaction, a challenging OED benchmark example introduced in [134], and discussed in several other publications, e.g., [15, 126, 135]. The reaction scheme is the following:

$$\begin{aligned} A + B &\to C \\ A + C &\rightleftharpoons D \\ 3A &\to E \end{aligned}$$

Educts are phenylisocyanate A and butanol B in the solvent dimethylsulfoxide L. During the reaction the product urethane C, the byproduct allophanate D and the byproduct isocyanate E are formed.

The products C, D, and E are modelled as differential states while A, B, and L can be computed from C, D, and E using molar number balance. In total, six parameters have to be identified, namely the frequency factors and activation energies for the Arrhenius kinetics. To achieve this, one experiment is to be designed for the time horizon $[t_0, t_\mathrm{f}] = [0h, 80h]$. Every 30 minutes, a measurement may be taken using one out of three potential measurement procedures. The reactor is run in a stirrer tank with two feeds: Feed 1 contains phenylisocyanate and the solvent, feed 2 contains dimethylsulfoxide and the solvent. Both can be fed into the reactor during the process. Furthermore, the temperature can be controlled. The full model including process constraints and measurement methods is summarized in Table 11.3.

Remark 11.1. Note the different interpretation of the measurement grids τ^m in the CSTR model and in the Urethane model: In the CSTR model, we are interested in the optimal sampling *times*. We approximate this by choosing a grid of potential measurements. There, the choice of the measurement grid is arbitrary. In the Urethane model, the measurement grid is considered to be part of the model formulation. The sampling times are fixed but the *choice* of measurement function at each time is optimized.

Parameterization and starting values

For the numerical computation, we parameterize $u_1(t) = \dot{feed}_1(t)$, $u_2(t) = \dot{feed}_2(t)$, and $u_3 = \dot{T}(t)$ by piecewise constant functions on 16 equidistant intervals. The actual process controls $T(t)$, $feed_1(t)$, and $feed_2(t)$ are set up as additional differential states, so we

States

$$\dot{n}_C = V \cdot (r_1 - r_2 + r_3), \qquad n_C(0) = 0$$
$$\dot{n}_D = V \cdot (r_2 - r_3), \qquad n_D(0) = 0$$
$$\dot{n}_E = V \cdot r_4, \qquad n_E(0) = 0$$
$$\dot{feed}_1 = u_1, \qquad feed_1(0) = 0$$
$$\dot{feed}_2 = u_2, \qquad feed_2(0) = 0$$
$$\dot{T} = u_3, \qquad T(0) = 293.15$$

Educts

$$n_{A,e} = n_{A,e1,0} \cdot feed_1$$
$$n_{B,e} = n_{B,e2,0} \cdot feed_2$$
$$n_{L,e} = n_{L,e1,0} \cdot feed_1 + n_{L,e2,0} \cdot feed_2$$
$$n_A = n_{A,0} + n_{A,e} - n_C - 2 \cdot n_D - 3 \cdot n_E$$
$$n_B = n_{B,0} + n_{B,e} - n_C - n_D$$
$$n_L = n_{L,0} + n_{L,e}$$

Reaction Rates

$$r_1 = k_{ref1} \cdot \exp\left(-\frac{E_{a,1}}{R} \cdot \left(\frac{1}{T} - \frac{1}{363.16}\right)\right) \cdot \frac{n_A}{V} \cdot \frac{n_B}{V}$$
$$r_2 = k_{ref2} \cdot \exp\left(-\frac{E_{a,2}}{R} \cdot \left(\frac{1}{T} - \frac{1}{363.16}\right)\right) \cdot \frac{n_A}{V} \cdot \frac{n_C}{V}$$
$$r_3 = k_{ref2} \cdot \exp\left(-\frac{E_{a,2}}{R} \cdot \left(\frac{1}{T} - \frac{1}{363.16}\right)\right) \cdot \left(K_{C2} \cdot e^{-\frac{\Delta H_2}{R} \cdot \left(\frac{1}{T} - \frac{1}{T_{C2}}\right)}\right)^{-1} \cdot \frac{n_D}{V}$$
$$r_4 = k_{ref4} \cdot \exp\left(-\frac{E_{a,4}}{R} \cdot \left(\frac{1}{T} - \frac{1}{363.16}\right)\right) \cdot \left(\frac{n_A}{V}\right)^2$$
$$V = \frac{n_A \cdot M_A}{\rho_A} + \frac{n_B \cdot M_B}{\rho_B} + \frac{n_C \cdot M_C}{\rho_C} + \frac{n_D \cdot M_D}{\rho_D} + \frac{n_E \cdot M_E}{\rho_E} + \frac{n_L \cdot M_L}{\rho_L}$$

Time Dependent Controls

$$0 \le feed_1, feed_2 \le 1$$
$$0 \le u_1, u_2 \le 0.0125$$
$$273.15 \le T \le 473.15$$
$$-40 \le u_3 \le 40$$

Time Independent Controls

$$0.1 \le n_{A,0} \le 1, \qquad 0 \le n_{B,0}, n_{L,0} \le 1$$
$$\frac{n_{A,0}M_A + n_{B,0}M_B}{n_{A,0}M_A + n_{B,0}M_B + n_{L,0}M_L} \le 0.8$$
$$n_{A,0}M_A\rho_A^{-1} + n_{B,0}M_B\rho_B^{-1} + n_{L,0}M_L\rho_L^{-1} \le 7.5 \cdot 10^{-4}$$

Measurements

$$h_1(t) = 100 \cdot \frac{n_A \cdot M_A}{n_A \cdot M_A + n_B \cdot M_B + n_C \cdot M_C + n_D \cdot M_D + n_E \cdot M_E + n_L \cdot M_L}$$
$$h_2(t) = 100 \cdot \frac{n_C \cdot M_C}{n_A \cdot M_A + n_B \cdot M_B + n_C \cdot M_C + n_D \cdot M_D + n_E \cdot M_E + n_L \cdot M_L}$$
$$h_3(t) = 100 \cdot \frac{n_D \cdot M_D}{n_A \cdot M_A + n_B \cdot M_B + n_C \cdot M_C + n_D \cdot M_D + n_E \cdot M_E + n_L \cdot M_L}$$
$$h_4(t) = 100 \cdot \frac{n_E \cdot M_E}{n_A \cdot M_A + n_B \cdot M_B + n_C \cdot M_C + n_D \cdot M_D + n_E \cdot M_E + n_L \cdot M_L}$$
$$w_i^2 = w_i^3, \quad w_i^1 + w_i^2 + w_i^4 \le 1 \quad i = 0, \ldots, 16$$

Figure 11.3.: Urethane Reaction Model.

Parameter	Value
k_{ref1}	$5.0 \cdot 10^{-4}$
$E_{a,1}$	35240
k_{ref2}	$8.0 \cdot 10^{-8}$
$E_{a,2}$	85000
k_{ref4}	$1.0 \cdot 10^{-8}$
$E_{a,4}$	35000

Constant	Value	Constant	Value
M_A	0.11911	ρ_A	1095
M_B	0.07412	ρ_B	809
M_C	0.19323	ρ_C	1415
M_D	0.31234	ρ_D	1528
M_E	0.35733	ρ_E	1451
M_L	0.07806	ρ_L	1101
ΔH_2	-17031.0	K_{C2}	0.17

Table 11.2.: Parameters and constants for the Urethane model.

end up with a total of six state variables. The constraints on the controls are formulated as path constraints. The multiple shooting grid comprises 8 equidistant intervals. The resulting NLP has a total of 470 variables and 415 constraints, 349 of which are nonlinear. The linear constraints are the measurement restrictions and localization constraints for the time-independent controls to achieve separability of the NLP.

As starting values for the optimization we set $u_1(t) = u_2(t) = 0.0125$ and $u_3(t) = 2$ for $t \in [0, 80]$. The starting values for the time independent controls are $n_{A,0} = 0.1$, $n_{B,0} = 0$, and $n_{L,0} = 0.2$. All 51 measurement weights are set to 0.33. Values for the shooting variables are obtained by forward integration and all parameters are scaled to 1. The absolute and relative integrator tolerances are set to 10^{-8}.

11.2.2. *A*-optimal design

We compute an OED with respect to the *A*-criterion. With the configuration given above, the *A*-criterion objective value of the starting point is $\Phi_A = 1265.52$.

When we run the optimization, `blockSQP` converges in 40 iterations taking 12.4 seconds. The objective value is reduced to $\Phi_A = 0.1763$. Table 11.3 lists the standard deviations of the parameters before and after the optimization and Figure 11.1 depicts the optimal controls and the corresponding states. For 12 out of 16 measurement times, the weights are integral, i.e., exactly one of the measurement procedures is selected. If the fractional weights are rounded to the next integer, the objective increases only slightly to $\Phi_A = 0.1765$.

11.2.3. Comparison with existing implementation

Again, we compare the new method to the existing implementation in VPLAN. We optimize with respect to the *A*-criterion using a single shooting approach and SNOPT to solve the NLP. Here, the default settings of SNOPT are used, as they provided the best performance in our tests. The single shooting NLP has 106 variables and 76 constraints (path and measurement constraints). The method finds a different local minimum, yielding an objective value of $\Phi_A = 0.2324$. Figure 11.5 depicts the optimal controls and corresponding states.

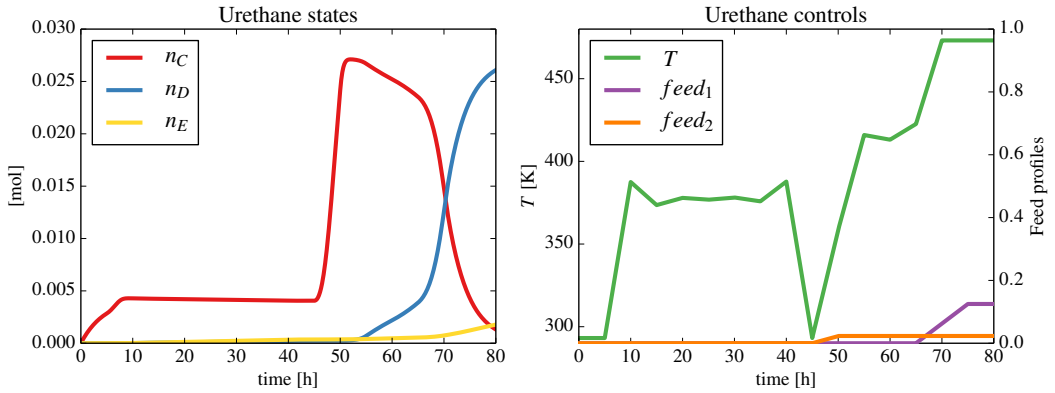


Figure 11.4.: Optimal states and controls with respect to the A-criterion for the Urethane example computed with `blockSQP`.

Parameter	Std. dev. before	Std. dev. after
k_{ref1}	20.40%	34.17%
$E_{a,1}$	11.97%	24.70%
k_{ref2}	220.09%	37.60%
$E_{a,2}$	52.21%	7.92%
k_{ref4}	5209.56%	67.30%
$E_{a,4}$	6981.39%	52.87%

Table 11.3.: Standard deviation of parameters at an A-optimal design computed with `blockSQP`.

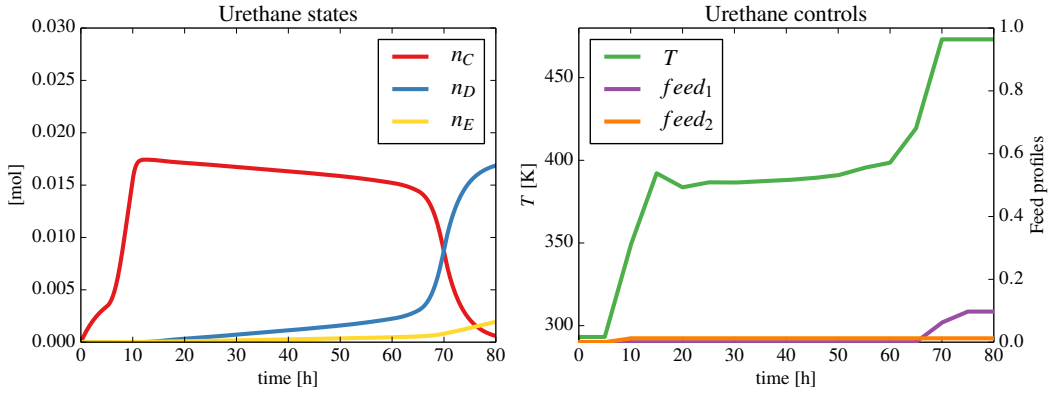


Figure 11.5.: Optimal states and controls with respect to the A-criterion for the Urethane example computed with SNOPT.

The existing VPLAN implementation requires 234 SQP major iterations and takes 283.2 seconds. We note that the number of SQP iterations is larger by almost a factor of 6. Here, the single shooting approach struggles with the highly nonlinear dynamics and takes only small steps. We also note that the cost of one iteration in single shooting is more than 3 times higher than for DMS because of the reasons provided in Section 11.1.4.

Chapter 12.

Numerical study of lifting

In Chapter 11, we already used the capability of `muse` to employ shooting grids that are coarser than the control grids. In many cases, coarser shooting grids yield the best overall performance as a trade-off between the number of SQP iterations, QP size, and sensitivity load. In this chapter, we study the effect of lifting in OC and OED problems on the number of SQP iterations. To this end, we consider several problems from the test set of Chapter 10 with fixed control discretization grids, but vary the multiple shooting grid. This gives rise to families of equivalent NLPs in the sense of Def. 5.1 which we solve with `blockSQP`.

Then, we study one of these problems, a Lotka–Volterra OED problem, in detail. Here, an increasing number of shooting intervals has a negative impact on the number of SQP iterations. Further experiments to investigate the local convergence properties reveal that the effect is due to the higher number of iterations *before* a domain of local convergence is reached. Once that has happened, the SR1 updates provide fast, grid-independent convergence. In the local setting, the nonlinear transformation of the OED objective introduced in [150] can further reduce the number of SQP iterations.

Finally, we study OC problems where the objective is to track an optimal solution. Their Hessian is positive definite. Here, numerical results for five problems show that lifting has a positive impact on the number of SQP iterations.

12.1. Experimental setup and results

We consider several problems from the test set of Chapter 10:

- The OC problems `oc-goddard`, `oc-lotka`, and `oc-particle-steering`, and
- the OED problems `oed-catalyst-mixing`, `oed-lotka`, and `oed-urethane`.

For `oed-urethane`, the control functions are discretized on a grid of 16 intervals. For the five remaining problems, we use 64 equidistant intervals. For the OED problems, we ignore the measurement restrictions and eliminate the measurement weights from the problems by internally setting all $w_i = 1$. This eliminates a number of local minima in the OED problems that make a comparison difficult. For all problems, we use the starting values as given in Appendix B

We now consider different, equidistant shooting grids to obtain a families of NLPs that are equivalent in the sense of Def. 5.1. We choose grids with 2^i, $i = 0, \ldots, 6$ (`oed-urethane`:

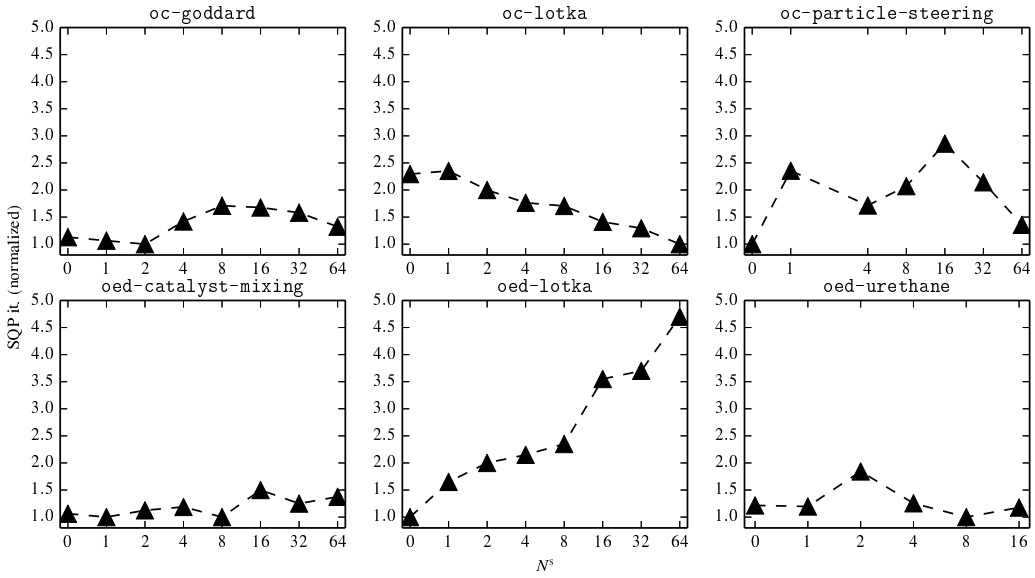


Figure 12.1.: Number of shooting intervals plotted versus number of SQP iterations for three OC and OED problems. The control discretization grid is fixed. The markers at 0 correspond to single shooting. The number of iterations are normalized to the lowest number.

$i = 0, \ldots, 4$) shooting intervals, where 0 intervals represents a single shootin parameterization. This way, every shooting node coincides with a point in the control grid. Note the difference between one shooting interval and the single shooting parameterization: One shooting interval corresponds to a decoupling of the objective and yields a Hessian with two diagonal blocks. That means the dynamic system is evaluated as part of the constraints while with direct single shooting the solution of the dynamic system is evaluated as part of the objective.

Results

Figure 12.1 shows the relative number of SQP iterations normalized to the best method. For all problems except `oed-urethane`, all formulations converged to the same minimum. For `oed-urethane`, the six formulations yielded six different local minima.

The results for four of the problems, `oc-goddard`, `oc-particle-steering`, `oed-catalyst-mixing`, and `oed-urethane`, do not provide a clear message in terms of convergence speed. The number of iterations do not differ much, or do not follow a clear pattern as in the case of `oc-particle-steering`. But the results for the two remaining problems, `oed-lotka` and `oc-lotka`, exhibit a certain monotonicity: When we increase the number of shooting intervals, the number of SQP iterations increases for `oed-lotka` and decreases for `oc-lotka`. In the following sections, we study this behavior in more detail.

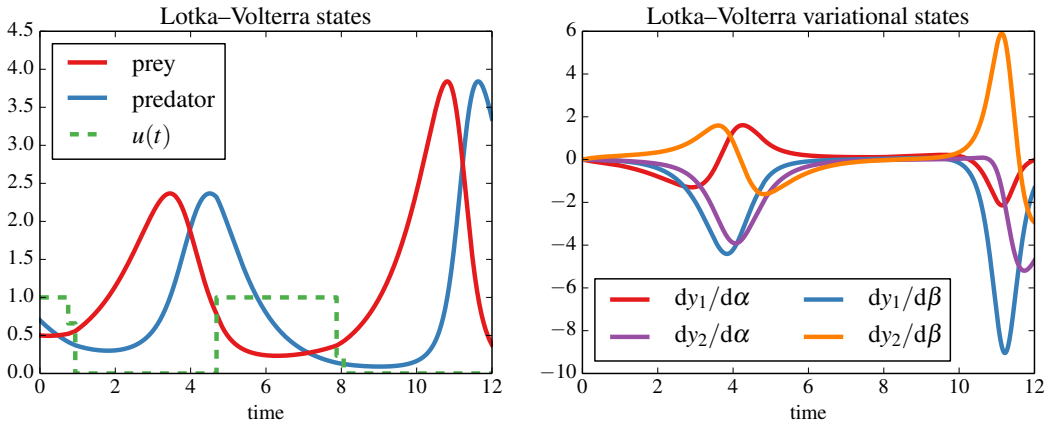


Figure 12.2.: Optimal controls and states with respect to the A-criterion for the Lotka–Volterra example.

12.2. A Lotka–Volterra OED problem

We consider the Lotka–Volterra example from [170]. The system equations are given by

$$\dot{y}_1(t) = \quad y_1(t) - \alpha y_1(t)y_2(t) - 0.4u(t)y_1(t), \qquad y_1(0) = 0.5, \tag{12.1a}$$
$$\dot{y}_2(t) = -y_2(t) + \beta y_1(t)y_2(t) - 0.2u(t)y_2(t), \qquad y_2(0) = 0.7, \tag{12.1b}$$
$$t \in [0,12] \qquad \leq u(t) \leq 1.$$

We optimize the A-criterion on the covariance matrix of α and β evaluated at $\alpha = \beta = 1$. Measurements of y_1 and y_2 are taken at $\tau_j^{\mathrm{m}} = j \cdot \frac{12}{64}$, $j = 0,\ldots,64$. The control is discretized by piecewise constant functions and the starting values for the control variables are set to $q_i = 0.3$, $i = 0,\ldots,N^{\mathrm{c}}$. The starting values for the shooting variables are obtained by forward integration. We set the absolute and relative integration tolerances to 10^{-10} and the tolerances for `blockSQP`, $\varepsilon_{\mathrm{tol}}$ and $\varepsilon_{\mathrm{feas}}$, to 10^{-6}. All formulations produce the same minimum with objective value $\Phi_A = 0.0032653$. Figure 12.2 shows the optimal control and the corresponding nominal and variational states at the solution.

12.2.1. Results for remote starting values

Figure 12.3 shows the distribution of SR1 and BFGS updates for all formulations. Especially for finer shooting grids, SR1 updates are accepted mostly in the final iterations, when a region of local convergence is reached. We attribute the slow convergence to the fact that for the block-BFGS update many blocks are damped. The *minimum* number of damped blocks for all SQP iterations where the BFGS update is used is as follows:

N^{s}	S	1	2	4	8	16	32	64
min. # damped blocks	0	0	1	2	3	5	11	23

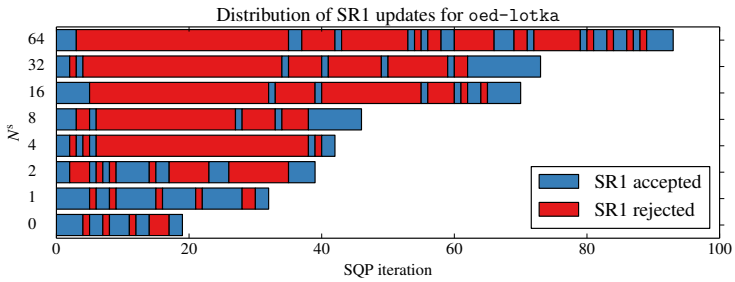


Figure 12.3.: SQP iterations plotted against the distribution of BFGS and SR1 updates for different shooting formulations of the `oed-lotka` example. The starting values for the control variables are $q_i = 0.3$, $i = 0, \ldots, N^c$.

We see that for $N^s > 1$, an unmodified BFGS step is never taken. We suspect that this causes the algorithm to proceed only slowly towards a region where SR1 updates are possible. This confirms our theoretical analysis from Sec. 5.3.

12.2.2. Results for starting values close to the solution

To get a better idea of the local convergence properties, we now choose a starting value close to the optimal solution. For this, we perturb the optimal values of the control variables as shown in Fig. 12.2 by 0.01. Again, starting values for the shooting variables are obtained by forward integration. Fig. 12.4 shows the distribution of SR1 and BFGS updates when started close to the solution. First, we note that for formulations with a higher number of shooting intervals, the SR1 update is still rejected more often. However, the overall number of iterations in the local setting is independent of the number of shooting intervals. If few shooting intervals are used, the total number of iterations in the local setting is similar or even higher than with remote starting values.This is due to the fact that low-rank quasi-Newton updates need a certain number of iterations to collect enough curvature information about the problem.

12.2.3. Results: Nonlinear transformation of OED objective

We now test the nonlinear transformation proposed in [150] on the `oed-lotka` problem. Instead of the A-criterion $\mathrm{tr}(C)$ we minimize the function $-\mathrm{tr}(C)^{-2}$. This preserves all local minima but helps again ill-conditioning as shown in [150]. The implementation of the transformation is straightforward and does not introduce any noticeable costs.

Table 12.1 lists the number of SQP iterations for the formulation with the nonlinear transformed objective. The results for the A-criterion objective are given for comparison. We remark that all formulations reproduced the optimal objective value with the same accuracy, that means "early termination" of the SQP method due to scaling issues was not the case. In the remote case, the new formulation suffers from similar problems as the A-criterion. Relatively few SR1 updates are accepted and the number of iterations for fine shooting grids

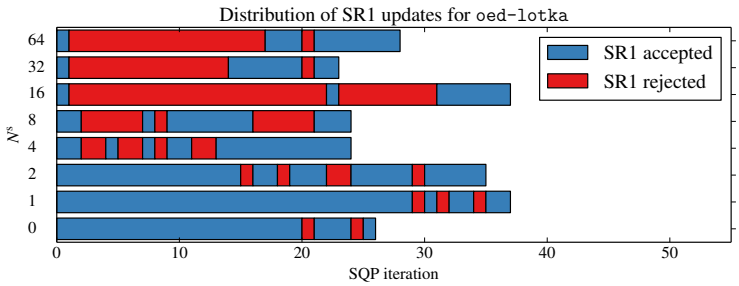


Figure 12.4.: SQP iterations plotted against the distribution of BFGS and SR1 updates for different shooting formulations of the `oed-lotka` example. The starting values for the control variables are obtained by perturbing the optimal solution depicted in Fig. 12.2 by 0.01.

is higher than for coarse ones. However, when we consider the local setting, we see that the new formulation converges two to six times faster than the standard formulation. In particular, the convergence is independent of the shooting grid. This makes the nonlinear transformed objective an interesting alternative to the standard OED objectives.

N^s	$\varphi = \mathrm{tr}(C)$	$\varphi = -\mathrm{tr}(C)^{-2}$
S	20 (14)	12(9)
1	33 (27)	39(18)
2	40 (19)	30(12)
4	43 (8)	38(9)
8	47 (15)	43(10)
16	71 (17)	113(14)
32	74 (19)	71(13)
64	94 (25)	68(12)

(a) Results for `blockSQP` with starting values $q_i = 0.3$, $i = 0, \ldots, N^c$.

N^s	$\varphi = \mathrm{tr}(C)$	$\varphi = -\mathrm{tr}(C)^{-2}$
S	27 (25)	8(8)
1	38 (35)	6(6)
2	36 (31)	6(6)
4	25 (18)	11(8)
8	25 (14)	11(10)
16	38 (9)	8 (8)
32	24 (10)	8 (7)
64	29 (12)	9 (7)

(b) Results for `blockSQP` with starting values close to the solution.

Table 12.1.: Number of SQP iterations for different shooting grids and different formulations of the objective. Listed are iteration numbers for the A-criterion objective and the nonlinearly transformed A-criterion. In parentheses, the number of accepted SR1 updates is given.

12.3. Problems with a tracking objective

For the `oc-lotka` test example we observe a decreasing number of SQP iterations for finer shooting grids. The objective of this problem is to minimize the Euclidean distance to a

prescribed steady state:

$$\min_{y,u} \int_{t_0}^{t_\mathrm{f}} (y_1(\tau)-1)^2 + (y_2(\tau)-1)^2 \,\mathrm{d}\tau,$$

where y_1 and y_2 are given by the ODEs (12.1). This is a special case of a *tracking* problem, where the distance to a reference trajectory y^ref is to be minimized:

$$\begin{aligned}
\min_{y,y_0,u} \quad & \tfrac{1}{2}\int_{t_0}^{t_\mathrm{f}} \left\| y(\tau) - y^\mathrm{ref}(\tau) \right\|_2^2 \,\mathrm{d}\tau && && \text{(12.2a)}\\
\text{s.t.} \quad & \dot{y}(t) = f(t, y(t), z(t), p, u(t)), && t \in [t_0, t_\mathrm{f}], && \text{(12.2b)}\\
& y(t_0) = y_0, && && \text{(12.2c)}\\
& b_L \le u(t) \le b_U, && t \in [t_0, t_\mathrm{f}], && \text{(12.2d)}
\end{aligned}$$

Tracking problems appear, e.g., in the context of nonlinear model-predictive control, where DMS has been applied successfully [58]. In the following, we study the problem of tracking an optimal solution for the OED and OC problems from Sec. 12.1.

12.3.1. Problem discretization and properties of the NLP

We apply direct multiple shooting with a piecewise constant control discretization to problem (12.2) and replace the integral with a sum over all points in the control discretization grid τ^c. We assume that y^ref is the optimal solution of the original problem obtained with the same discretization τ^c. For the following analysis, we assume that the state and control are scalar. Furthermore, we drop inequality constraints. Then the resulting NLP reads as:

$$\begin{aligned}
\min_{s,q} \quad & \tfrac{1}{2}\sum_{j=0}^{N^\mathrm{s}} \sum_{\substack{i:\\ \tau_j^\mathrm{s} \le \tau_i^\mathrm{c} < \tau_{j+1}^\mathrm{s}}}^{N^\mathrm{s}} \left(y(\tau_i^\mathrm{c}; s_j, \hat{q}_j) - y^\mathrm{ref}(\tau_i^\mathrm{c}) \right)^2 && && \text{(12.3a)}\\
\text{s.t.} \quad & 0 = y(\tau_j^\mathrm{s}; s_{j-1}, \hat{q}_{j-1}) - s_j, && j = 1, \dots, N^\mathrm{s}, && \text{(12.3b)}
\end{aligned}$$

Note that for different multiple shooting grids τ^s we again obtain a family of equivalent NLPs. This is because we sum over all points of the control grid τ^c, which is the same for all NLPs. We define

$$F : \mathbb{R}^{n_\mathrm{y}} \times \mathbb{R}^{n_\mathrm{q}} \to \mathbb{R}^{N^\mathrm{c}+1}, \qquad F_i(s,q) := y(\tau_i^\mathrm{c}; s_{j(i)}, \hat{q}_{j(i)}) - y^\mathrm{ref}(\tau_i^\mathrm{c}), \quad i = 0, \dots, N^\mathrm{c}.$$

Here $j(i)$ denotes the index j of the shooting node corresponding to index $i \in \tau^\mathrm{c}$. Furthermore, we denote the Jacobian of F by $J := J(s,q) \in \mathbb{R}^{(N^\mathrm{c}+1)\times(n_\mathrm{y}+n_\mathrm{q})}$ and by J_{s_j} the column of J corresponding to s_j. For the NLP (12.3), the following holds:

Lemma 12.1. *Assume that the first and second derivatives of F are bounded in a neighborhood of the global minimum. Then the Hessian of the Lagrangian*

$$\mathcal{L}(s,q,\lambda) = \tfrac{1}{2}\left\|F(s,q)\right\|_2^2 - \sum_{i=j}^{N^\mathrm{s}} \lambda_j \left(y(\tau_j^\mathrm{s}; s_{j-1}, \hat{q}_{j-1}) - s_j \right) \tag{12.4}$$

of Problem (12.3) *is positive semi-definite in a neighborhood of the global minimum.*

Proof. At the global minimum, we know that $F(s^*, q^*) = 0$. Therefore, in a neighborhood of (s^*, q^*) it holds that $\|F(s,q)\| < \varepsilon$. Furthermore, the gradient of the Lagrangian must vanish at the solution. The gradient with respect to s_j is given by

$$\begin{aligned}
\nabla_{s_j} \mathcal{L} &= J_{s_j}^T F + \lambda_j - \lambda_{j+1} \frac{\mathrm{d}y(\tau_{j+1}^{\mathrm{s}})}{\mathrm{d}s_j}, && j = 0, \dots, N^{\mathrm{s}} - 1, \\
\nabla_{s_{N^{\mathrm{s}}}} \mathcal{L} &= J_{s_{N^{\mathrm{s}}}}^T F + \lambda_{N^{\mathrm{s}}},
\end{aligned}$$

therefore at the solution we have that

$$\begin{aligned}
\lambda_{N^{\mathrm{s}}}^* &= -J_{s_{N^{\mathrm{s}}}}^T F, \\
\lambda_j^* &= -J_{s_j}^T F + \lambda_{j+1}^* \frac{\mathrm{d}y(\tau_{j+1}^{\mathrm{s}})}{\mathrm{d}s_j}, && j = 0, \dots, N^{\mathrm{s}} - 1.
\end{aligned}$$

Because we assume that the first derivatives are bounded, we conclude that $\lambda_j = O(\|F(s,q)\|)$. The Hessian of the Lagrangian is given by

$$\nabla^2 \mathcal{L}(s, q, \lambda) = J^T J + \sum_{i=0}^{N^{\mathrm{c}}} F_i \nabla^2 F_i - \sum_{i=j}^{N^{\mathrm{s}}} \lambda_j \nabla^2 \left(y(\tau_j^{\mathrm{s}}; s_{j-1}, \hat{q}_{j-1}) - s_j \right).$$

Because we assume that the second derivatives are bounded, the positive definite term $J^T J$ dominates the right-hand side if $\|F(s,q)\|$ is sufficiently small. □

12.3.2. Results

For the problems from Sec. 12.1, we now consider the problem of tracking a reference solution. Note that the resulting NLPs have exactly the same set of variables and constraints and differ only in the objective function. The reference solutions are the optimal trajectories computed in Sec. 12.1. For OED problems, this comprises the nominal and variational states. The problem `oed-urethane` exhibits many local minima and is excluded here. The global minimum has objective value 0 by construction, so that the Hessian is positive semi-definite near the solution.

The starting values for all optimization variables and the algorithmic options are chosen as in Sec. 12.1. We remark that in practice, one would probably chose the starting values $s^{[0]} = y^{\mathrm{ref}}$ to exploit prior knowledge about the optimal solution. Also, an algorithm based on a Gauss–Newton approximation of the Hessian may be more appropriate for tracking problems. However, here the goal is to study the effect of lifting when applied to a problem with positive definite Hessian.

Fig. 12.5 depicts the relative number of SQP iterations for different shooting grids. If the number of shooting nodes is chosen to be smaller than eight, the algorithm did not converge. We see that the number of SQP iterations decreases as the shooting grid is refined. We conclude that lifting has a positive effect for tracking problems with regard to convergence speed.

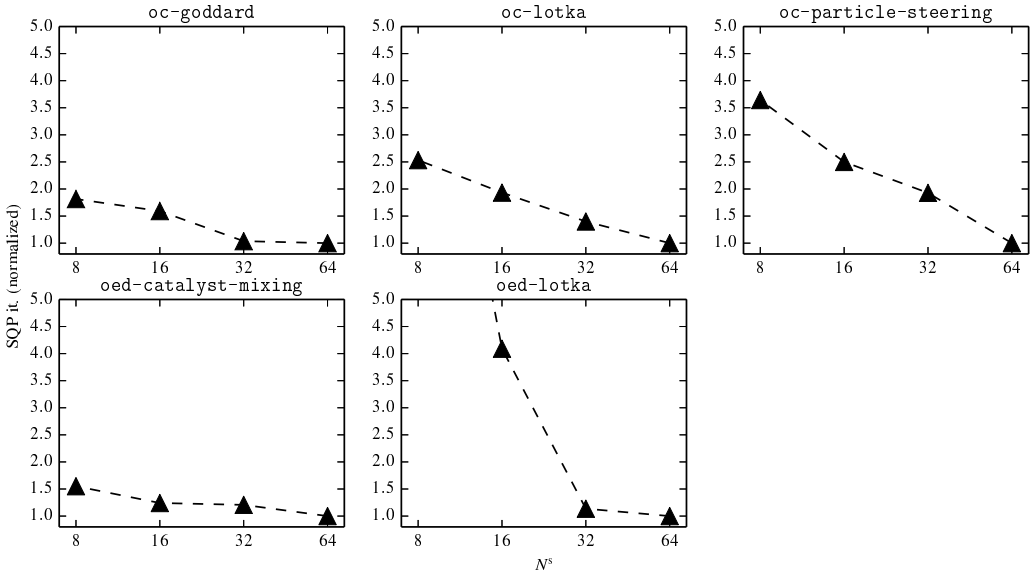


Figure 12.5.: Number of SQP iterations plotted versus number of shooting intervals for tracking problems.

Conclusions and future work

In this work, we have developed theory, algorithms and software for the numerical solution of OED and OC problems constrained by DAEs. Based on Bock's direct multiple shooting method, we have proposed new ways to transcribe infinite-dimensional OED problems to finite-dimensional NLPs. A special feature of our method in comparison to previously proposed DMS implementations is the ability to cope with different grids for the multiple shooting parameterization, the discretization of controls, the discretization of path constraints, and—in the case of OED—the discretization of measurement functions. After all, in problems from industrial practice often very specific restrictions apply concerning process and measurement constraints. On the other hand, it must be guaranteed that the underlying dynamic system can be solved reliably and efficiently during the optimization. Different grids provide the freedom to choose a parameterization that suits best the problem at hand.

From a theoretical point of view, the new formulation allows to investigate DMS in the light of the lifted Newton method: Different multiple shooting grids correspond to different liftings of the constraint function. The same SQP method applied to these problems iterates in different spaces and approaches the solution differently.

We have developed a new SQP method tailored to the specially structured NLPs arising in DMS that uses blockwise quasi-Newton updates to approximate the Hessian. Numerical tests with a newly developed implementation of the method have shown how fast local convergence is impeded if only positive definite blockwise Hessian approximations are used, which confirms our theoretical findings on model problems. Another important result is that if the block-diagonal structure in DMS is ignored and full-space updates are employed, an off-the-shelf SQP solver fails on non-trivial OED problems. This supports our claim that DMS must be used with a dedicated SQP method. Furthermore, we have performed numerical experiments to investigate the effect of lifting on the number of SQP iterations by using different multiple shooting grids. Closer examination of a Lotka–Volterra OED problem has shown that locally, fast convergence can be achieved independent of the multiple shooting grid, provided that indefinite Hessians are used. Often, however, other, non-local effects dominate the overall number of SQP iterations. An exception are tracking problems, where refining the multiple shooting grid considerably improves the speed of convergence.

We want to point out that in practice, the overall performance of the optimization method is not only dependent on the (expected) number of SQP iterations. It rather is a complex combination of several factors, including the size of the control discretization, the size and the characteristic of the dynamic system, the size and condition of the arising quadratic subproblems, and the parallelization architecture available. Here, `muse` and `blockSQP` are a powerful toolkit that can be easily adjusted to constitute a fast and reliable solution method for the specific OED or OC problem instance at hand.

Directions of further research

During the work on this thesis, several questions arose which we think deserve further investigation. We conclude this thesis with the most promising ones.

Communication between integrator and SQP method
Occasionally, we observed in our numerical tests certain "outlier" steps. That means, after a number of SQP iterations, in which only few active set iterations are needed and good progress towards a solution is made, a step is computed that takes the iterate relatively far from the current region. We conjecture that this is due to internal changes in the integrator, which is, strictly speaking, a change of the NLP constraint function. Then the secant information used to produce the new Hessian may be highly unreliable and thus results in a poor step. A remedy for this undesirable behavior could be to employ global error estimators such as those proposed in [18]. Then a fixed integration grid should be used during several SQP iterations while the global error is monitored along the way. Adjustments to the grid based on the estimated error should only be made in few iterations and within this iterations, existing secant information should probably be discarded to avoid poor Hessian approximations.

Development of a second-order SQP method
The SQP framework presented in Ch. 6 is independent of the specific Hessian approximations and the QP method presented in Ch. 8 can exploit arbitrary sparsity patterns. In many applications, (sparse) Hessians are available by employing algorithmic differentiation. This motivates the development of a second-order SQP method, where exact Hessians are used within the inner loop of the line search SQP Algorithm 6. A challenge is how to proceed if the exact Hessian is rejected. If the Lagrangian is still partially separable, sparsity-exploiting quasi-Newton updates such as [111] can be employed as fallback. Another possibility that should be investigated in this context are strategies based on recently proposed convexification schemes [96].

Danksagungen

Mein Dank geht an meine Lehrer und Mentoren der *Fakultät für Mathematik und Informatik*, von denen ich in den letzten Jahren so viel lernen durfte und die wesentlich zum Gelingen dieser Arbeit beigetragen haben: *Stefan Körkel*, *Georg Bock*, *Sebastian Sager (Uni Magdeburg)*, *Christian Kirches* und *Johannes Schlöder*. Jede Diskussion mit Ihnen über neue (und alte) Ideen hat diese Arbeit ein Stück weiter voran gebracht.

Des weiteren gebührt *Andreas Wächter* großer Dank, der mir den Aufenthalt an der Northwestern University ermöglicht hat und mich dort so hervorragend betreut hat. Diese Arbeit hat wesentlich von seiner Expertise profitiert und die drei Monate in Evanston waren insgesamt eine tolle Erfahrung für mich.

Meinen Kollegen der Arbeitsgruppen *Optimum Experimental Design*, *Simulation & Optimierung*, *Optimization of Uncertain Systems* und *Model-Based Optimizing Control* danke ich für die freundliche und kooperative Atmosphäre, die vielen fachlichen und nicht-fachlichen Gespräche und dafür, dass in den vergangenen Jahren das Arbeiten ein solches Vergnügen war. Für die schöne und erfolgreiche Zusammenarbeit im SIAM Chapter danke ich *Dörte Beigel*, *Holger Diedam*, *Kathrin Hatz* und *Andreas Sommer*. Für Diskussionen im Kontext meiner Arbeit danke ich besonders *Chris Hoffmann*, *Robert Kircheis*, *Mario Mommer*, *Andreas Potschka*, *Andreas Schmidt*, *Sebastian Walter* und *Christoph Weiler*.

Der *BASF SE*, der *Heidelberg Graduate School MathComp* sowie der *Northwestern University* danke ich für finanzielle Unterstützung.

Von Herzen danke ich meinen Eltern *Günter* und *Gerlinde* sowie meinen Schwestern *Katrin* und *Amelie* für ihre fortwährende Unterstützung. Schließlich gehen Liebe und Dank an meine Frau *Teresa* und meinen Sohn *Ferenc*, die mich immer wieder daran erinnern, was wirklich wichtig ist.

Appendix

A. Model equations for the oc-ocean and oc-fermenter problems

The oc-ocean model

The model describes fossil fuel consumption and sequestration into the ocean [169]. It is a two box model where S describes the carbon stock in the atmosphere and upper layer ocean, R describes the carbon stock in fossil reserve and D_L the carbon stock in the deeper layer. Together with a utility function y we have the following three differential states that are integrated on $[0,400]$:

$$\begin{aligned} \dot{y}(t) &= \exp(-\rho t)(U(t) - A(t) - u_1(t)C(t) - D(t), && y(0) = 0 \\ \dot{S}(t) &= u_1(t) - u_2(t) - \gamma \cdot (S(t) - \omega D_L(t)), && S(0) = 2000 \\ \dot{R}(t) &= -u_1(t), && R(0) = 10^4 \end{aligned}$$

where

$$\begin{aligned} U(t) &= bu_1(t) - \mu u_1(t)^2, & D(t) &= v(0.3S(t) - S_{\text{preind}})^2, \\ A(t) &= a_1 u_2(t) + a_2 u_2(t)^2, & D_L(t) &= D_{L,0} + R_0 + S_0 - R(t) - S(t), \\ C(t) &= c_1 - c_2 R(t). \end{aligned}$$

The objective is to maximize the utility function y at $t_f = 400$. The states S and R, and the controls u_1 and u_2 are subject to the following bounds:

$$0 \le S(t), R(t) \le 10^5, \qquad 0 \le u_1(t), u_2(t) \le 40.$$

Values for all remaining constants are listed in Table A.1.

Sym.	Value	Sym.	Value	Sym.	Value
ρ	0.03	a_1	2	S_{preind}	600
γ	0.001	a_2	2	S_0	2000
ω	0.1	v	1	R_0	10^4
b	50	c_1	50	$D_{L,0}$	$2.3 \cdot 10^4$
μ	0.5	c_2	0.004		

Table A.1.: Constants for the oc-ocean model.

The `oc-fermenter` model

The model describes a fermentation process with two substrates S_1 and S_2, and two products P and G. Enzyme biomass concentration is modeled by a state E. Further states are the fermentation volume V and the accumulated product P_{acc} and substrates $S_{1,\text{acc}}$ and $S_{2,\text{acc}}$. S_1 and S_2 can be fed into the reactor. This is described by two controls u_{S_1} and u_{S_2}. Furthermore, P can be harvested with rate u_P. The full system reads as

$$
\begin{aligned}
\dot{P}(t) =& \mu_p E(t) S_1(t) S_2(t) - P(t) \frac{u_{S_1}(t) + u_{S_2}(t)}{25V(t)}, && P(0) = 0,\\
\dot{S_1}(t) =& -\gamma_{x,1} E(t) S_1(t) S_2(t) G(t) - \gamma_{p,1} E(t) S_1(t) S_2(t)\\
&+ 0.42 \frac{u_{S_1}}{25V(t)} - S_1(t) \frac{u_{S_1}(t) + u_{S_2}(t)}{25V(t)}, && S_1(0) = 0.03,\\
\dot{S_2}(t) =& -\gamma_{x,2} E(t) S_1(t) S_2(t) G(t) - \gamma_{p,2} E(t) S_1(t) S_2(t)\\
&+ 0.333 \frac{u_{S_2}}{25V(t)} - S_2(t) \frac{u_{S_1}(t) + u_{S_2}(t)}{25V(t)}, && S_2(0) = 0.03,\\
\dot{E}(t) =& \mu_x E(t) S_1(t) S_2(t) G(t) - E(t) \frac{u_{S_1}(t) + u_{S_2}(t)}{25V(t)}, && S_3(0) = 0.01,\\
\dot{V}(t) =& u_{S_1}(t) + u_{S_2}(t) - u_P(t), && V(0) = 0.3,\\
\dot{G}(t) =& -\gamma_{x,g} E(t) S_1(t) S_2(t) G(t) - G(t) \frac{u_{S_1}(t) + u_{S_2}(t)}{25V(t)}, && G(0) = 0.1,\\
\dot{P}_{\text{acc}}(t) =& u_P(t) P(t) + \frac{u_{S_1}(t) + u_{S_2}(t) - u_P(t)}{25} P(t) + V(t) \dot{P}(t), && P_{\text{acc}}(0) = 0.0,\\
\dot{S}_{1,\text{acc}}(t) =& 0.0168 u_{S_1}, && S_{1,\text{acc}}(0) = 0.009,\\
\dot{S}_{2,\text{acc}}(t) =& 0.01332 u_{S_2}, && S_{2,\text{acc}}(0) = 0.009.
\end{aligned}
$$

The objective is to minimize the following cost function:

$$\phi(t_\text{f}) = \frac{2 S_{1,\text{acc}}(t_\text{f}) S_{2,\text{acc}}(t_\text{f})}{P_{\text{acc}}(t_\text{f})}$$

at $t_\text{f} = 1$. The control functions are subject to the bounds

$$0 \leq u_{S_1} \leq 15, \qquad 0 \leq u_{S_2} \leq 1, \qquad 0 \leq u_P \leq 30,$$

and the states are constrained by

$$
\begin{aligned}
&0 \leq P(t) \leq 0.1, && 0 \leq S_1(t) \leq 0.04, && 0 \leq S_2(t) \leq 0.03\\
&0 \leq E(t) \leq 0.1, && 0.3 \leq V(t) \leq 0.45, && 0 \leq G(t) \leq 0.1\\
&0 \leq P_{\text{acc}}(t) \leq 0.05, && 0 \leq S_{1,\text{acc}}(t) \leq 0.2, && 0 \leq S_{2,\text{acc}}(t) \leq 0.025.
\end{aligned}
$$

Values for all remaining constants are listed in Table A.2.

Sym.	Value
μ_x	$2 \cdot 10^5$
μ_p	5000
$\gamma_{x,g}$	$5 \cdot 10^4$

Sym.	Value
$\gamma_{x,1}$	10^5
$\gamma_{p,1}$	$2 \cdot 10^4$

Sym.	Value
$\gamma_{x,2}$	1500
$\gamma_{p,2}$	$5 \cdot 10^4$

Table A.2.: Constants for the `oc-fermenter` model.

B. Control and parameter values for the optimization benchmark collection

Table B.1 lists the starting values for the controls that were used in the numerical tests in Chapter 10. Table B.2 lists the values of the parameters that were used for the OED problems.

Problem name	u_1	u_2	u_3
`oed-catalyst-mixing`	0.5		
`oed-cstr`	0.9	300.0	
`oed-lotka`	0.3		
`oed-polymerization`	$[2.73 : 3.13]$	1.0	1.0
`oed-urethane`	0.0125	0.0125	2.0
`oed-yeast`	0.2	35	
`oc-batchdist`	8.0		
`oc-cops-catalyst-mixing`	0.5		
`oc-cops-goddard`	1.75		
`oc-cops-hangglider`	0.7		
`oc-cops-hanging-chain`	$[-2.0 : 6.0]$		
`oc-cops-particle-steering`	0.0		
`oc-fermenter`	$[14.0 : 0.0]$	0.7	0.5
`oc-lotka`	0.3		
`oc-ocean`	30.0	10.0	
`oc-williams-otto`	0.5	2.9	

Table B.1.: Starting values for the control variables for the optimization benchmark collection. A single value indicates that all q are set to the same value. Squared brackets indicate that q is chosen as a linear interpolation between $[q_0 : q_{N^c}]$.

Problem name	p_1	p_2	p_3	p_4	p_5	p_6
`oed-catalyst-mixing`	10					
`oed-cstr`	549.36	$7.2 \cdot 10^{10}$				
`oed-lotka`	1	1				
`oed-polymerization`	$8.96 \cdot 10^6$	-3557	$4.16 \cdot 10^{16}$	-16573		
`oed-urethane`	$5 \cdot 10^{-4}$	35240	$8 \cdot 10^{-8}$	$8.5 \cdot 10^4$	10^{-8}	$3.5 \cdot 10^4$
`oed-yeast`	0.31	0.18	0.55	0.05		

Table B.2.: Parameter values for the OED problem instances. Note that for the computation of the covariance matrix, all parameters are scaled to 1.

Bibliography

[1] W. Achtziger and C. Kanzow. "Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications". In: *Mathematical Programming* 114.1 (2008), pp. 69–99 (cit. on p. 28).

[2] M. Al-Baali. "Global and superlinear convergence of a restricted class of self-scaling methods with inexact line searches, for convex functions". In: *Computational Optimization and Applications* 9.2 (1998), pp. 191–203 (cit. on p. 132).

[3] M. Al-Baali. "Numerical experience with a class of self-scaling quasi-Newton algorithms". In: *Journal of optimization theory and applications* 96.3 (1998), pp. 533–553 (cit. on p. 132).

[4] J. Albersmeyer. "Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems". PhD thesis. Universität Heidelberg, 2010 (cit. on pp. 52, 86, 159).

[5] J. Albersmeyer and M. Diehl. "The Lifted Newton Method and its Application in Optimization". In: *SIAM Journal on Optimization* 20.3 (2010), pp. 1655–1684 (cit. on pp. 21, 22, 106–108).

[6] P. Amestoy, H. S. Dollar, J. K. Reid, and J. A. Scott. *An approximate minimum degree algorithm for matrices with dense rows*. Tech. rep. RAL-TR-2007-020. Rutherford Appleton Laboratory, 2007 (cit. on p. 144).

[7] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. "A fully asynchronous multifrontal solver using distributed dynamic scheduling". In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 15–41 (cit. on p. 144).

[8] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. Philadelphia, PA: SIAM, 1995 (cit. on p. 51).

[9] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations*. Philadelphia: SIAM, 1998 (cit. on p. 51).

[10] A. C. Atkinson and A. Donev. *Optimum Experimental Designs*. Ed. by A. C. Atkinson, J. B. Copas, D. A. Pierce, M. J. Schervish, and D. M. Titterington. Oxford Statistical Sciences Series 8. Oxford: Oxford University Press, 1992 (cit. on pp. 18, 60, 68).

[11] J. Banga and E. Balsa-Canto. "Parameter estimation and optimal experimental design". In: *Essays Biochem* 45 (2008), pp. 195–210 (cit. on p. 60).

[12] W. Bangerth, R. Hartmann, and G. Kanschat. "deal.II—a general-purpose object-oriented finite element library". In: *ACM Transactions on Mathematical Software (TOMS)* 33.4 (2007), p. 24 (cit. on p. 159).

[13] Y. Bard. *Nonlinear Parameter Estimation*. Academic Press, 1974 (cit. on p. 61).

[14] I. Bauer, H. G. Bock, S. Körkel, and J. P. Schlöder. "Numerical Methods for Initial Value Problems and Derivative Generation for DAE Models with Application to Optimum Experimental Design of Chemical Processes". In: *Scientific Computing in Chemical Engineering II*. Ed. by F. Keil, W. Mackens, H. Voß, and J. Werther. Springer, 1999, pp. 282–289 (cit. on p. 51).

[15] I. Bauer, H. G. Bock, S. Körkel, and J. P. Schlöder. "Numerical methods for optimum experimental design in DAE systems". In: *J. Comput. Appl. Math.* 120.1-2 (2000), pp. 1–15 (cit. on pp. 18, 60, 79, 86, 185).

[16] I. Bauer, H. G. Bock, D. Leineweber, and J. P. Schlöder. "Direct Multiple Shooting Methods for Control and Optimization of DAE in Chemical Engineering". In: *Scientific Computing in Chemical Engineering II*. Ed. by F. Keil, W. Mackens, H. Voß, and J. Werther. Springer, 1999, pp. 2–18 (cit. on p. 45).

[17] I. Bauer, H. G. Bock, and J. P. Schlöder. *DAESOL – a BDF-code for the numerical solution of differential algebraic equations*. Internal report, IWR, SFB 359, Universität Heidelberg. 1999 (cit. on pp. 81, 86, 159).

[18] D. Beigel. "Efficient goal-oriented global error estimation for BDF-type methods using discrete adjoints". PhD thesis. Universität Heidelberg, 2012 (cit. on p. 200).

[19] M. Benzi, G. H. Golub, and J. Liesen. "Numerical solution of saddle–point problems". In: *Acta Numerica* 14 (2005), pp. 1–137 (cit. on p. 66).

[20] M. J. Best. "Equivalence of some Quadratic Programming Algorithms". In: *Mathematical Programming* 30 (1984), pp. 71–87 (cit. on p. 33).

[21] M. J. Best. "An Algorithm for the Solution of the Parametric Quadratic Programming Problem". In: *Applied Mathematics and Parallel Computing – Festschrift for Klaus Ritter*. Ed. by H. Fischer, B. Riedmüller, and S. Schäffler. Heidelberg: Physica-Verlag, 1996. Chap. 3, pp. 57–76 (cit. on pp. 33, 137, 138, 142).

[22] J. T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. Philadelphia: SIAM, 2001 (cit. on p. 43).

[23] L. T. Biegler. "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation". In: *Computers & Chemical Engineering* 8 (1984), pp. 243–248 (cit. on p. 45).

[24] L. T. Biegler. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Series on Optimization. SIAM, 2010 (cit. on p. 43).

[25] C. H. Bischof, A. Carle, P. D. Hovland, P. Khademi, and A. Mauer. *ADIFOR 2.0 User's Guide (Revision D)*. Tech. rep. Mathematics, Computer Science Division Technical Memorandum no. 192, and Center for Research on Parallel Computation, 1998 (cit. on p. 159).

[26] C. H. Bischof, A. Carle, P. Khademi, and A. Mauer. "ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs". In: *IEEE Computational Science & Engineering* 3 (1996), pp. 18–32 (cit. on p. 159).

[27] J. Bisschop and A. Meeraus. "Matrix augmentation and partitioning in the updating of the basis inverse". In: *Mathematical Programming* 13.1 (1977), pp. 241–254 (cit. on pp. 19, 21, 34, 144).

[28] H. G. Bock. "Numerical Solution of Nonlinear Multipoint Boundary Value Problems with Applications to Optimal Control". In: *Zeitschrift für Angewandte Mathematik und Mechanik* 58 (1978), p. 407 (cit. on p. 45).

[29] H. G. Bock. "Numerical treatment of inverse problems in chemical reaction kinetics". In: *Modelling of Chemical Reaction Systems*. Ed. by K. Ebert, P. Deuflhard, and W. Jäger. Vol. 18. Springer Series in Chemical Physics. Heidelberg: Springer, 1981, pp. 102–125 (cit. on pp. 41, 51, 62).

[30] H. G. Bock. "Recent advances in parameter identification techniques for ODE". In: *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Ed. by P. Deuflhard and E. Hairer. Boston: Birkhäuser, 1983, pp. 95–121 (cit. on p. 51).

[31] H. G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*. Vol. 183. Bonner Mathematische Schriften. Bonn: Universität Bonn, 1987 (cit. on pp. 41, 56, 60–62, 66, 159).

[32] H. G. Bock, E. Eich, and J. P. Schlöder. "Numerical Solution of Constrained Least Squares Boundary Value Problems in Differential-Algebraic Equations". In: *Numerical Treatment of Differential Equations. Proceedings of the NUMDIFF-4 Conference, Halle-Wittenberg, 1987*. Ed. by K. Strehmel. Vol. 104. Texte zur Mathematik. Teubner, 1988, pp. 269–280 (cit. on p. 47).

[33] H. G. Bock, E. A. Kostina, and O. I. Kostyukova. "Conjugate Gradient Methods for Computing Covariance Matrices for Constrained Parameter Estimation Problems". In: *SIAM Journal on Matrix Analysis and Application* 29 (2007), p. 626 (cit. on p. 63).

[34] H. G. Bock and K. J. Plitt. "A Multiple Shooting algorithm for direct solution of optimal control problems". In: *Proceedings of the 9th IFAC World Congress*. Budapest: Pergamon Press, 1984, pp. 242–247 (cit. on pp. 18, 43, 45, 55, 56, 102, 127, 171).

[35] P. T. Boggs and J. W. Tolle. "Sequential Quadratic Programming". In: *Acta Numerica* 4 (1995), pp. 1–51 (cit. on pp. 19, 27).

[36] G. E. P. Box and N. R. Draper. *Empirical model-building and response surfaces*. John Wiley & Sons, 1987 (cit. on pp. 18, 60).

[37] C. G. Broyden. "The convergence of a class of double–rank minimization algorithms". In: *Journal of the Institute of Mathematics and its Applications* 6 (1970), pp. 76–90 (cit. on p. 34).

[38] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. New York: Wiley, 1975 (cit. on p. 43).

[39] R. Bulirsch. *Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung*. Tech. rep. Oberpfaffenhofen: Carl-Cranz-Gesellschaft, 1971 (cit. on p. 45).

[40] R. H. Byrd, H. F. Khalfan, and R. B. Schnabel. "Analysis of a symmetric rank-one trust region method". In: *SIAM Journal on Optimization* 6.4 (1996), pp. 1025–1039 (cit. on pp. 35, 128).

[41] R. H. Byrd, J. Nocedal, and R. B. Schnabel. "Representations of quasi-Newton matrices and their use in limited memory methods". In: *Mathematical Programming* 63 (1994), pp. 129–156 (cit. on pp. 35, 131).

[42] R. H. Byrd, F. E. Curtis, and J. Nocedal. "Infeasibility detection and SQP methods for nonlinear optimization". In: *SIAM Journal on Optimization* 20.5 (2010), pp. 2281–2299 (cit. on p. 32).

[43] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. "An algorithm for nonlinear optimization using linear programming and equality constrained subproblems". In: *Mathematical Programming* 100.1 (2003), pp. 27–48 (cit. on p. 39).

[44] R. H. Byrd, J. Nocedal, and R. A. Waltz. "KNITRO: An integrated package for nonlinear optimization". In: *Large-scale nonlinear optimization*. Springer, 2006, pp. 35–59 (cit. on pp. 39, 41).

[45] R. H. Byrd, J. Nocedal, and R. A. Waltz. "Steering exact penalty methods for nonlinear programming". In: *Optimization Methods and Software* 23.2 (2008), pp. 197–213 (cit. on p. 36).

[46] A. R. Conn, N. I. M. Gould, and P. L. Toint. "Convergence of quasi-Newton matrices generated by the symmetric rank one update". In: *Mathematical Programming* 50.1-3 (1991), pp. 177–195 (cit. on pp. 35, 128, 129).

[47] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Lancelot: A FORTRAN Package for Large-Scale Nonlinear Optimization (Release A)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1992 (cit. on p. 40).

[48] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust region methods*. Vol. 1. Siam, 2000 (cit. on p. 37).

[49] B. L. Contesse. "Une caractérisation complète des minima locaux en programmation quadratique." In: *Numerische Mathematik* 34 (1980), pp. 315–332 (cit. on pp. 33, 106).

[50] M. Contreras and R. A. Tapia. "Sizing the BFGS and DFP updates: numerical study". In: *Journal of Optimization Theory and Applications* 78.1 (1993), pp. 93–108 (cit. on pp. 21, 132, 134, 135).

[51] F. E. Curtis, T. C. Johnson, D. P. Robinson, and A. Wächter. "An Inexact Sequential Quadratic Optimization Algorithm for Nonlinear Optimization". In: *SIAM Journal on Optimization* 24.3 (2014), pp. 1041–1074 (cit. on p. 34).

[52] J. E. Cuthrell and L. T. Biegler. "On the optimization of differential-algebraic process systems". In: *AIChE Journal* 33.8 (1987), pp. 1257–1270 (cit. on p. 45).

[53] L. Dagum and R. Menon. "OpenMP: an industry standard API for shared-memory programming". In: *Computational Science & Engineering, IEEE* 5.1 (1998), pp. 46–55 (cit. on p. 164).

[54] W. C. Davidon. "Variable metric method for minimization". In: *SIAM Journal on Optimization* 1.1 (1991), pp. 1–17 (cit. on pp. 34, 128).

[55] J. E. Dennis Jr and J. J. Moré. "Quasi-Newton methods, motivation and theory". In: *SIAM review* 19.1 (1977), pp. 46–89 (cit. on pp. 35, 130).

[56] J. E. Dennis Jr, D. M. Gay, and R. E. Welsch. "Algorithm 573: NL2SOL—an adaptive nonlinear least-squares algorithm [E4]". In: *ACM Transactions on Mathematical Software (TOMS)* 7.3 (1981), pp. 369–383 (cit. on p. 133).

[57] M. Diehl, H. G. Bock, and E. Kostina. "An approximation technique for robust nonlinear optimization". In: *Mathematical Programming* 107 (2006), pp. 213–230 (cit. on p. 171).

[58] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. "Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations". In: *J. Proc. Contr.* 12.4 (2002), pp. 577–585 (cit. on pp. 45, 196).

[59] M. Diehl, D. B. Leineweber, and A. A. S. Schäfer. *MUSCOD-II Users' Manual*. IWR-Preprint 2001-25. Universität Heidelberg, 2001 (cit. on p. 19).

[60] E. D. Dolan, J. J. Moré, and T. S. Munson. *Benchmarking Optimization Software with COPS 3.0*. Tech. rep. ANL/MCS-TM-273. 9700 South Cass Avenue, Argonne, IL 60439, U.S.A.: Mathematics and Computer Science Division, Argonne National Laboratory, 2004 (cit. on p. 171).

[61] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical programming* 91.2 (2002), pp. 201–213 (cit. on pp. 171, 174).

[62] E. D. Dolan, J. J. Moré, and T. S. Munson. "Optimality measures for performance profiles". In: *SIAM Journal on Optimization* 16.3 (2006), pp. 891–909 (cit. on p. 32).

[63] I. S. Duff. "MA57 — a code for the solution of sparse symmetric definite and indefinite systems". In: *ACM Transactions on Mathematical Software* 30.2 (2004), pp. 118–144 (cit. on pp. 21, 144).

[64] P. S. Dwyer and M. S. MacPhail. “Symbolic matrix derivatives”. In: *The annals of mathematical statistics* (1948), pp. 517–534 (cit. on p. 92).

[65] S Engell and K.-U. Klatt. “Nonlinear control of a non-minimum-phase CSTR”. In: *American Control Conference, 1993*. IEEE. 1993, pp. 2941–2945 (cit. on pp. 172, 181).

[66] D. Espie and S. Macchietto. “The optimal design of dynamic experiments”. In: *AIChE Journal* 35.2 (1989), pp. 223–229 (cit. on pp. 18, 60, 79).

[67] V. V. Fedorov. *Theory of optimal experiments*. Elsevier, 1972 (cit. on pp. 18, 60).

[68] H. J. Ferreau, H. G. Bock, and M. Diehl. “An online active set strategy to overcome the limitations of explicit MPC”. In: *International Journal of Robust and Nonlinear Control* 18.8 (2008), pp. 816–830 (cit. on p. 137).

[69] H. J. Ferreau, A. Potschka, and C. Kirches. *qpOASES webpage*. `http://www.qpOASES.org/`. 2007–2015 (cit. on p. 137).

[70] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. “qpOASES: A parametric active-set algorithm for quadratic programming”. In: *Mathematical Programming Computation* (2014), pp. 1–37 (cit. on pp. 19, 21, 33, 34, 137, 147).

[71] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. New York: John Wiley and Sons, Inc., 1968 (cit. on p. 128).

[72] R. Fletcher. “A new approach to variable metric algorithms”. In: *Computer Journal* 13 (1970), pp. 317–322 (cit. on p. 34).

[73] R. Fletcher. “A general quadratic programming algorithm”. In: *J. Inst. Math. Appl.* 7 (1971), pp. 76–91 (cit. on p. 33).

[74] R. Fletcher. *Practical Methods of Optimization*. Second. ISBN 0-471-49463-1 (paperback). Chichester: Wiley, 1987 (cit. on p. 27).

[75] R. Fletcher, N. I. M. Gould, S. Leyffer, P. L. Toint, and A. Wächter. “Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming”. In: *SIAM Journal on Optimization* 13.3 (2002), pp. 635–659 (cit. on p. 36).

[76] R. Fletcher and S. Leyffer. “User manual for filterSQP”. In: *University of Dundee Numerical Analysis Report NA-181* (1998) (cit. on pp. 19, 39).

[77] R. Fletcher and S. Leyffer. “Nonlinear programming without a penalty function”. In: *Mathematical Programming* 91.2 (2002), pp. 239–269 (cit. on pp. 36, 37).

[78] R. Fletcher, S. Leyffer, and P. L. Toint. “A brief history of filter methods”. In: *Preprint ANL/MCS-P1372-0906, Argonne National Laboratory, Mathematics and Computer Science Division* (2006) (cit. on p. 36).

[79] R. Fletcher, S. Leyffer, and P. L. Toint. “On the Global Convergence of a Filter–SQP Algorithm”. In: *SIAM Journal on Optimization* 13.1 (2002), pp. 44–59 (cit. on p. 36).

[80] R. Fletcher and M. J. D. Powell. "A rapidly convergent descent method for minimization". In: *The Computer Journal* 6.2 (1963), pp. 163–168 (cit. on p. 34).

[81] C. A. Floudas. *Deterministic global optimization*. Vol. 37. Springer, 1999 (cit. on p. 28).

[82] J. Forbes. "Model structure and adjustable parameter selection for operations optimizations". PhD thesis. McMaster University, Hamilton, Canada, 1994 (cit. on p. 171).

[83] G. Franceschini and S. Macchietto. "Model-based design of experiments for parameter precision: State of the art". In: *Chemical Engineering Science* 63 (2008), pp. 4846–4872 (cit. on pp. 18, 59, 60).

[84] F. Galvanin, A. Boschiero, M. Barolo, and F. Bezzo. "Model-based design of experiments in the presence of continuous measurement systems". In: *Industrial & Engineering Chemistry Research* 50.4 (2011), pp. 2167–2175 (cit. on p. 79).

[85] F. Galvanin, S. Macchietto, and F. Bezzo. "Model-based design of parallel experiments". In: *Industrial & Engineering Chemistry Research* 46.3 (2007), pp. 871–882 (cit. on pp. 18, 172).

[86] R.-P. Ge and M. J. D. Powell. "The convergence of variable metric matrices in unconstrained optimization". In: *Mathematical programming* 27.2 (1983), pp. 123–143 (cit. on p. 131).

[87] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer, 2002 (cit. on p. 27).

[88] M. Gerdts. "Direct shooting method for the numerical solution of higher index DAE optimal control problems". In: *Journal of Optimization Theory and Applications* 117.2 (2003), pp. 267–294 (cit. on p. 45).

[89] M. Gerdts. *Optimal Control of ODEs and DAEs*. De Gruyter, 2012 (cit. on p. 43).

[90] M. B. Giles. "Collected matrix derivative results for forward and reverse mode algorithmic differentiation". In: *Advances in Automatic Differentiation*. Springer, 2008, pp. 35–44 (cit. on p. 92).

[91] P. E. Gill, L. O. Jay, M. W. Leonard, L. R. Petzold, and V. Sharma. "An SQP method for the optimal control of large-scale dynamical systems". In: *Journal of Computational and Applied Mathematics* 120.1-2 (2000), pp. 197–213 (cit. on p. 56).

[92] P. E. Gill and W. Murray. "Numerically Stable Methods for Quadratic Programming". In: *Mathematical Programming* 14 (1978), pp. 349–372 (cit. on p. 34).

[93] P. E. Gill, W. Murray, and M. A. Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". In: *SIAM Journal of Optimization* 12 (2002), pp. 979–1006 (cit. on pp. 19, 37, 160, 171, 179).

[94] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. *A Schur-complement method for sparse quadratic programming*. Tech. rep. Stanford Univ., CA (USA). Systems Optimization Lab., 1987 (cit. on pp. 19, 21, 34, 144).

[95] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. "Inertia-Controlling Methods for General Quadratic Programming". In: *SIAM Review* 33.1 (1991), pp. 1–36 (cit. on p. 33).

[96] P. E. Gill and E. Wong. *Convexification schemes for SQP methods*. Tech. rep. CCoM-14-06. Center for Computational Mathematics, University of California at San Diego, July 2014 (cit. on pp. 126, 200).

[97] P. E. Gill and E. Wong. "Methods for convex and general quadratic programming". In: *Mathematical Programming Computation* 7.1 (2015), pp. 71–112 (cit. on pp. 19, 21, 144).

[98] P. E. Gill, W. Murray, and M. A. Saunders. *User's Guide for SQOPT Version 7: Software for Large-Scale Linear and Quadratic Programming*. 2006 (cit. on p. 37).

[99] P. E. Gill, M. A. Saunders, and E. Wong. *On the Performance of SQP Methods for Nonlinear Optimization*. `http://www.optimization-online.org/DB_HTML/2015/01/4761.html`. 2015 (cit. on p. 19).

[100] P. E. Gill and E. Wong. "Sequential quadratic programming methods". In: *Mixed integer nonlinear programming*. Springer, 2012, pp. 147–224 (cit. on pp. 19, 27).

[101] D. Goldfarb. "A family of variable metric updates derived by variational means". In: *Mathematics of Computation* 24 (1970), pp. 23–26 (cit. on p. 34).

[102] D. Goldfarb and A. Idnani. "A numerically stable dual method for solving strictly convex quadratic programs". In: *Mathematical Programming* 27 (1983), pp. 1–33 (cit. on p. 33).

[103] G. H. Golub and C. F. Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012 (cit. on p. 145).

[104] N. I. M. Gould and P. L. Toint. "An iterative working-set method for large-scale nonconvex quadratic programming". In: *Applied Numerical Mathematics* 43.1 (2002), pp. 109–128 (cit. on pp. 19, 21, 144, 148).

[105] N. I. M. Gould. "On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem". In: *Mathematical Programming* 32.1 (1985), pp. 90–99 (cit. on p. 148).

[106] N. I. M. Gould, Y. Loh, and D. P. Robinson. "A filter method with unified step computation for nonlinear optimization". In: *SIAM Journal on Optimization* 24.1 (2014), pp. 175–209 (cit. on p. 36).

[107] N. I. M. Gould and P. L. Toint. "Nonlinear programming without a penalty function or a filter". In: *Mathematical Programming* 122.1 (2010), pp. 155–196 (cit. on p. 36).

[108] N. I. M. Gould and P. L. Toint. "A quadratic programming bibliography". In: *Numerical Analysis Group Internal Report* 1 (2000) (cit. on p. 32).

[109] N. I. M. Gould and P. L. Toint. "SQP Methods for Large-Scale Nonlinear Programming". English. In: *System Modelling and Optimization*. Ed. by M. Powell and S. Scholtes. Vol. 46. IFIP — The International Federation for Information Processing. Springer US, 2000, pp. 149–178 (cit. on pp. 19, 27).

[110] A. Griewank and P. L. Toint. "Local convergence analysis for partitioned quasi-Newton updates". In: *Numerische Mathematik* 39.3 (1982), pp. 429–448 (cit. on p. 127).

[111] A. Griewank and P. L. Toint. "Partitioned variable metric updates for large structured optimization problems". In: *Numerische Mathematik* 39.1 (1982), pp. 119–137 (cit. on pp. 127, 200).

[112] A. Griewank and A. Walther. *Evaluating Derivatives*. Second. SIAM, 2008 (cit. on pp. 51, 86).

[113] A. Griewank and P. L. Toint. "On the Unconstrained Optimization of Partially Separable Functions". In: *Nonlinear Optimization 1981*. Ed. by M. J. D. Powell. New York, NY: Academic Press, 1982, pp. 301–312 (cit. on p. 54).

[114] A. Griewank and A. Walther. "On constrained optimization by adjoint based quasi-Newton methods". In: *Optimization Methods and Software* 17.5 (2002), pp. 869–889 (cit. on p. 34).

[115] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*. Second. Vol. 8. Springer Series in Computational Mathematics. Berlin: Springer-Verlag, 1993 (cit. on p. 51).

[116] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. Second. Vol. 14. Springer Series in Computational Mathematics. Berlin: Springer, 1996 (cit. on p. 51).

[117] S. P. Han. "A globally convergent method for nonlinear programming". In: *JOTA* 22 (1977), pp. 297–310 (cit. on pp. 19, 30).

[118] K. Hatz, S. Leyffer, J. P. Schlöder, and H. G. Bock. *Regularizing Bilevel Nonlinear Programs by Lifting*. Tech. rep. Technical Report ANL/MCS-P4076-0613, Argonne National Laboratory, Mathematics and Computer Science Division, 2013 (cit. on p. 28).

[119] E. V. Haynsworth. "Determination of the inertia of a partitioned Hermitian matrix". In: *Linear algebra and its applications* 1.1 (1968), pp. 73–81 (cit. on p. 148).

[120] T. A. N. Heirung, B. E. Ydstie, and B. Foss. "An adaptive model predictive dual controller". In: *Adaptation and Learning in Control and Signal Processing*. Vol. 11. 1. 2013, pp. 62–67 (cit. on pp. 69, 183).

[121] J. Herold, S. F. Walter, S. Körkel, and M. Buchner. "Optimal experimental design for parameter estimation of the Fitness-Fatigue model". In: *Proceeding of the International Conference on Biomechanics and Sports Engineering, Riga, Latvia, 24-25 Oct 2014*. 2014 (cit. on p. 159).

[122] D. M. Hoang, T Barz, V. A. Merchan, L. T. Biegler, and H Arellano-Garcia. "Simultaneous solution approach to model-based experimental design". In: *AIChE Journal* (2013) (cit. on pp. 18, 79).

[123] D. M. Hoang. "A simultaneuos optimization approach to optimal experimental design". PhD thesis. Technische Universität Berlin, 2014 (cit. on pp. 172, 181).

[124] D. Janka. "Optimum Experimental Design and Multiple Shooting". Diploma Thesis. Universität Heidelberg, 2010 (cit. on pp. 18, 19, 79, 100, 101).

[125] D. Janka, C. Kirches, S. Sager, and A. Wächter. "An SR1-BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal Hessian matrix". In: *Mathematical Programming Computation* (2015 (submitted)) (cit. on pp. 108, 115, 137).

[126] D. Janka, S. Körkel, and H. G. Bock. "Direct multiple shooting for nonlinear optimum experimental design". In: *Multiple Shooting and Time Domain Decomposition Methods*. Ed. by T. Carraro, M. Geiger, S. Körkel, and R. Rannacher. Contributions in Mathematical and Computational Sciences. Springer, 2014 (submitted) (cit. on pp. 79, 185).

[127] S. J. Julier and J. K. Uhlmann. "Unscented filtering and nonlinear estimation". In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422 (cit. on p. 67).

[128] M. N. Jung, C. Kirches, and S. Sager. "On perspective functions and vanishing constraints in mixed-integer nonlinear optimal control". In: *Facets of Combinatorial Optimization*. Springer, 2013, pp. 387–417 (cit. on p. 28).

[129] H. F. Khalfan, R. H. Byrd, and R. B. Schnabel. "A theoretical and experimental study of the symmetric rank-one update". In: *SIAM Journal on Optimization* 3.1 (1993), pp. 1–24 (cit. on pp. 35, 128).

[130] R. Kircheis. "On efficient parameter estimation and optimum experimental design and applications in microbial-enhanced oil recovery". PhD thesis. Universität Heidelberg, 2015 (cit. on p. 159).

[131] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager. "A Factorization with Update Procedures for a KKT Matrix Arising in Direct Optimal Control". In: *Mathematical Programming Computation* 3.4 (2011), pp. 319–348 (cit. on p. 56).

[132] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager. "Block Structured Quadratic Programming for the Direct Multiple Shooting Method for Optimal Control". In: *Optimization Methods and Software* 26.2 (Apr. 2011), pp. 239–257 (cit. on p. 56).

[133] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager. "Complementary Condensing for the Direct Multiple Shooting Method". In: *Modeling, Simulation, and Optimization of Complex Processes. Proceedings of the Fourth International Conference on High Performance Scientific Computing, March 2-6, 2009, Hanoi, Vietnam*. Ed. by H. Bock, H. Phu, R. Rannacher, and J. Schlöder. Heidelberg Dordrecht London New York: Springer Verlag, 2012, pp. 195–206 (cit. on p. 56).

[134] S. Körkel. "Numerische Methoden für Optimale Versuchsplanungsprobleme bei nichtlinearen DAE-Modellen". PhD thesis. Heidelberg: Universität Heidelberg, 2002 (cit. on pp. 18, 60, 74, 79, 86, 92, 93, 155, 158, 159, 172, 181, 185).

[135] S. Körkel, E. Kostina, H. G. Bock, and J. P. Schlöder. "Numerical Methods for Optimal Control Problems in Design of Robust Optimal Experiments for Nonlinear Dynamic Processes". In: *Optimization Methods and Software* 19 (2004), pp. 327–338 (cit. on pp. 18, 73, 185).

[136] S. Körkel, A. Potschka, H. G. Bock, and S. Sager. "A Multiple Shooting Formulation for Optimum Experimental Design". In: *Mathematical Programming* (2012 (submitted revisions)) (cit. on pp. 18, 79, 83, 100).

[137] S. Körkel, H. Arellano-Garcia, J. Schöneberger, and G. Wozny. "Optimum Experimental Design for Key Performance Indicators". In: *Proceedings of 18th European Symposium on Computer Aided Process Engineering – ESCAPE 18*. Ed. by B. Braunschweig and X. Joulia. 2008 (cit. on p. 74).

[138] M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan. "The polynomial solvability of convex quadratic programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.5 (1980), pp. 223–228 (cit. on p. 106).

[139] A. Kud, S. Körkel, and S. Maixner. "A cubic equation of state based on saturated vapor modeling and the application of model-based design of experiments for its validation". In: *Chemical Engineering Science* 65.14 (2010), pp. 4194–4207 (cit. on p. 159).

[140] D. B. Leineweber. "Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen". Diploma thesis. Universität Heidelberg, 1995 (cit. on p. 56).

[141] D. B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*. Vol. 613. Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik. Düsseldorf: VDI Verlag, 1999 (cit. on p. 32).

[142] D. B. Leineweber, I. Bauer, A. A. S. Schäfer, H. G. Bock, and J. P. Schlöder. "An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II)". In: *Computers & Chemical Engineering* 27 (2003), pp. 157–174 (cit. on pp. 18, 19, 45, 56, 171).

[143] S. Leyffer and A. Mahajan. "Foundations of Constrained Optimization". In: *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley and Sons, Inc., 2011 (cit. on pp. 27, 35).

[144] S. Lindner. "Skalierung von BFGS Approximationen in einem SQP Verfahren". Bachelor's Thesis. Universität Heidelberg, 2013 (cit. on pp. 132, 135).

[145] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1-3 (1989), pp. 503–528 (cit. on pp. 35, 131).

[146] T. Lohmann, H. G. Bock, and J. P. Schlöder. "Numerical methods for parameter estimation and optimal experiment design in chemical reaction systems". In: *Industrial & Engineering Chemistry Research* 31.1 (1992), pp. 54–57 (cit. on pp. 18, 60).

[147] X. Lu. "A Study of the Limited Memory SR1 Method in Practice". PhD thesis. University of Colorado at Boulder, 1996 (cit. on pp. 128, 131).

[148] D. G. Luenberger. *Linear and nonlinear programming*. Reading, Massachusetts: Addison-Wesley, 1989 (cit. on p. 27).

[149] N. Maratos. "Exact penalty function algorithms for finite dimensional and control optimization problems". PhD thesis. Imperial College London, 1978 (cit. on p. 119).

[150] M. S. Mommer, A. Sommer, J. P. Schlöder, and H. G. Bock. "A nonlinear preconditioner for experimental design problems". In: *arXiv preprint arXiv:1108.1689* (2011) (cit. on pp. 100, 191, 194).

[151] B. A. Murtagh and M. A. Saunders. *MINOS 5.51 user's guide*. Tech. rep. SOL 83-20R. Stanford Univ., CA (USA). Systems Optimization Lab., 2003 (cit. on p. 40).

[152] M. Nattermann. "Numerical Methods for Optimum Experimental Design Based on Second-Order Approximation of Confidence Regions". PhD thesis. Universität Marburg, 2014 (cit. on p. 67).

[153] J. Nocedal. "Updating quasi-Newton matrices with limited storage". In: *Mathematics of Computation* 35 (1980), pp. 773–782 (cit. on p. 35).

[154] J. Nocedal and S. J. Wright. *Numerical Optimization*. Second. ISBN 0-387-30303-0 (hardcover). Berlin Heidelberg New York: Springer Verlag, 2006 (cit. on pp. 27, 31, 33, 35, 36, 40, 106, 108, 120, 127, 130).

[155] J. Nocedal and Y.-X. Yuan. "Analysis of a self-scaling quasi-Newton method". In: *Mathematical Programming* 61.1-3 (1993), pp. 19–37 (cit. on p. 132).

[156] S. S. Oren. "Perspectives on self-scaling variable metric algorithms". In: *Journal of Optimization Theory and Applications* 37.2 (1982), pp. 137–147 (cit. on p. 132).

[157] S. S. Oren and D. G. Luenberger. "Self-Scaling Variable Metric (SSVM) Algorithms Part I: Criteria and Sufficient Conditions for Scaling a Class of Algorithms". In: *Management Science* 20.5 (1974), pp. 845–862 (cit. on p. 132).

[158] S. S. Oren and E. Spedicato. "Optimal conditioning of self-scaling variable metric algorithms". In: *Mathematical programming* 10.1 (1976), pp. 70–90 (cit. on p. 132).

[159] M. R. Osborne. "On shooting methods for boundary value problems". In: *Journal of Mathematical Analysis and Applications* 27 (1969), pp. 417–433 (cit. on p. 56).

[160] P. M. Pardalos and C. A. Floudas. *State of the art in global optimization: computational methods and applications*. Kluwer Academic Publishers, 1996 (cit. on p. 28).

[161] P. M. Pardalos and G. Schnitger. “Checking local optimality in constrained quadratic programming is NP-hard”. In: *Operations Research Letters* 7.1 (1988), pp. 33–35 (cit. on p. 33).

[162] P. M. Pardalos and S. A. Vavasis. “Quadratic programming with one negative eigenvalue is NP-hard”. In: *Journal of Global Optimization* 1.1 (1991), pp. 15–22 (cit. on pp. 33, 106).

[163] K. J. Plitt. “Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen”. Diploma Thesis. Rheinische Friedrich–Wilhelms–Universität Bonn, 1981 (cit. on pp. 43, 45, 55, 56, 102, 127).

[164] A. Potschka, H. G. Bock, and J. P. Schlöder. “A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems”. In: *Optimization Methods and Software* 24.2 (2009), pp. 237–252 (cit. on p. 49).

[165] M. J. D. Powell. “A fast algorithm for nonlinearly constrained optimization calculations”. In: *Numerical Analysis*. Ed. by G. e. Watson. Vol. 3. Springer Verlag. Berlin, 1978, pp. 144–157 (cit. on pp. 19, 30, 108, 130).

[166] M. J. D. Powell. “The convergence of variable metric methods for non-linearly constrained optimization calculations”. In: *Nonlinear programming* 3 (1978) (cit. on pp. 34, 35).

[167] F. Pukelsheim. *Optimal Design of Experiments*. Classics in Applied Mathematics 50. ISBN 978-0-898716-04-7. SIAM, 2006 (cit. on pp. 18, 60, 68).

[168] A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. “Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood”. In: *Bioinformatics* 25.15 (2009), pp. 1923–1929 (cit. on p. 67).

[169] W. Rickels and S. Sager. Personal communication. 2015 (cit. on p. 203).

[170] S. Sager. *On the Integration of Optimization Approaches for Mixed-Integer Nonlinear Optimal Control*. Universität Heidelberg. Habilitationsschrift. Aug. 2011 (cit. on pp. 166, 172, 193).

[171] S. Sager. “Sampling Decisions in Optimum Experimental Design in the Light of Pontryagin’s Maximum Principle”. In: *SIAM Journal on Control and Optimization* 51.4 (2013), pp. 3181–3207 (cit. on pp. 81, 99, 112).

[172] S. Sager, H. G. Bock, M. Diehl, G. Reinelt, and J. P. Schlöder. “Numerical methods for optimal control with binary control functions applied to a Lotka-Volterra type fishing problem”. In: *Recent Advances in Optimization*. Ed. by A. Seeger. Vol. 563. Lectures Notes in Economics and Mathematical Systems. ISBN 978-3-5402-8257-0. Heidelberg: Springer, 2009, pp. 269–289 (cit. on pp. 168, 171).

[173] R. W. H. Sargent and G. R. Sullivan. “The development of an efficient optimal control package”. In: *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2*. Ed. by J. Stoer. Heidelberg: Springer, 1978 (cit. on p. 45).

[174] A. A. S. Schäfer. “Efficient reduced Newton-type methods for solution of large-scale structured optimization problems with application to biological and chemical processes”. PhD thesis. Universität Heidelberg, 2005 (cit. on p. 56).

[175] O. Schenk and K. Gärtner. “Solving unsymmetric sparse systems of linear equations with PARDISO”. In: *Future Generation Computer Systems* 20.3 (2004), pp. 475–487 (cit. on p. 144).

[176] R. Schenkendorf, A Kremling, and M Mangold. “Optimal experimental design with the sigma point method”. In: *IET systems biology* 3.1 (2009), pp. 10–23 (cit. on p. 67).

[177] K Schittkowski. “Experimental design tools for ordinary and algebraic differential equations”. In: *Industrial & Engineering Chemistry Research* 46.26 (2007), pp. 9137–9147 (cit. on pp. 18, 60, 79).

[178] M. Schlegel, K. Stockmann, T. Binder, and W. Marquardt. “Dynamic optimization using adaptive control vector parameterization”. In: *Computers & Chemical Engineering* 29.8 (2005), pp. 1731–1751 (cit. on p. 45).

[179] J. P. Schlöder. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*. Vol. 187. Bonner Mathematische Schriften. Bonn: Universität Bonn, 1988 (cit. on pp. 60, 61).

[180] J. C. Schoneberger, H. Arellano-Garcia, G. Wozny, S. Körkel, and H. Thielert. “Model-based experimental analysis of a fixed-bed reactor for catalytic SO2 oxidation”. In: *Industrial & Engineering Chemistry Research* 48.11 (2009), pp. 5165–5176 (cit. on p. 159).

[181] V. H. Schulz. “Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots”. PhD thesis. Universität Heidelberg, 1996 (cit. on pp. 45, 56).

[182] S. R. Searle. “Matrix algebra useful for statistics”. In: *New York* (1982) (cit. on p. 97).

[183] R. Serban and A. C. Hindmarsh. “CVODES: the sensitivity-enabled ODE solver in SUNDIALS”. In: *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers. 2005, pp. 257–269 (cit. on p. 86).

[184] D. F. Shanno. “Conditioning of Quasi–Newton methods for function minimization”. In: *Mathematics of Computation* 24.111 (July 1970), pp. 647–656 (cit. on p. 34).

[185] D. F. Shanno and K.-H. Phua. “Matrix conditioning and nonlinear optimization”. In: *Mathematical Programming* 14.1 (1978), pp. 149–160 (cit. on p. 132).

[186] M. C. Steinbach. “Fast recursive SQP methods for large-scale optimal control problems”. PhD thesis. Universität Heidelberg, 1995 (cit. on p. 56).

[187] O. Stryk. “Numerical solution of optimal control problems by direct collocation”. In: *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*. Vol. 111. Bulirsch et al., 1993, pp. 129–143 (cit. on p. 45).

[188] O. Stryk. “Numerische Lösung optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung der adjungierten Variablen”. PhD thesis. TU Munich, 1995 (cit. on p. 45).

[189] P. Tanartkit and L. T. Biegler. “Stable decomposition for dynamic optimization”. In: *Industrial and Engineering Chemistry Research* 34 (1995), pp. 1253–1266 (cit. on p. 56).

[190] R. Tapia. “On averaging and representation properties of the BFGS and related secant updates”. In: *Mathematical Programming* (2014), pp. 1–18 (cit. on pp. 127, 130).

[191] M. Ulbrich and S. Ulbrich. *Nichtlineare Optimierung*. Springer, 2012 (cit. on p. 27).

[192] M. Ulbrich, S. Ulbrich, and L. N. Vicente. “A globally convergent primal-dual interior-point filter method for nonlinear programming”. In: *Mathematical Programming* 100.2 (2004), pp. 379–410 (cit. on p. 36).

[193] S. Ulbrich. “On the superlinear local convergence of a filter-SQP method”. In: *Mathematical Programming* 100.1 (2004), pp. 217–245 (cit. on p. 36).

[194] R. J. Vanderbei and D. F. Shanno. “An interior-point algorithm for nonconvex nonlinear programming”. In: *Computational Optimization and Applications* 13 (1999), pp. 231–252 (cit. on p. 117).

[195] R. J. Vanderbei. “LOQO: An interior point code for quadratic programming”. In: *Optimization methods and software* 11.1-4 (1999), pp. 451–484 (cit. on pp. 41, 126).

[196] V. S. Vassiliadis, R. W. H. Sargent, and C. C. Pantelides. “Solution of a class of multistage dynamic optimization problems. Parts 1. & 2.” In: *Industrial and Engineering Chemistry Research* 10.33 (1994), pp. 2111–2133 (cit. on p. 45).

[197] A. Wächter and L. T. Biegler. “Line search filter methods for nonlinear programming: Local convergence”. In: *SIAM Journal on Optimization* 16.1 (2005), pp. 32–48 (cit. on pp. 19, 36, 117).

[198] A. Wächter and L. T. Biegler. “Line search filter methods for nonlinear programming: Motivation and global convergence”. In: *SIAM Journal on Optimization* 16.1 (2005), pp. 1–31 (cit. on pp. 19, 21, 36, 115–117, 122).

[199] A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical programming* 106.1 (2006), pp. 25–57 (cit. on pp. 41, 117, 126).

[200] S. F. Walter. “Structured higher-order algorithmic differentiation in the forward and reverse mode with application in optimum experimental design”. PhD thesis. Humboldt Universität zu Berlin, 2012 (cit. on p. 69).

[201] S. F. Walter, A. Schmidt, and S. Körkel. "Adjoint-based optimization of experimental designs with many control variables". In: *Journal of Process Control* 24.10 (2014), pp. 1504–1515 (cit. on pp. 79, 81).

[202] C. K. F. Weiler and S Körkel. "Optimum experimental design for extended Gaussian disorder modeled organic semiconductor devices". In: *Journal of Applied Physics* 113.9 (2013), p. 094903 (cit. on p. 159).

[203] R. B. Wilson. "A simplicial algorithm for concave programming". PhD thesis. Harvard University, 1963 (cit. on pp. 19, 30).

[204] P. Wolfe. "The simplex method for quadratic programming". In: *Econometrica* 27 (1959), pp. 382–398 (cit. on p. 33).

[205] M. Wulkow. "Computer Aided Modeling of Polymer Reaction Engineering—The Status of Predici, I-Simulation". In: *Macromolecular Reaction Engineering* 2.6 (2008), pp. 461–494 (cit. on p. 172).

[206] H. Yabe, H. J. Martinez, and R. A. Tapia. "On sizing and shifting the BFGS update within the sized-Broyden family of secant updates". In: *SIAM Journal on Optimization* 15.1 (2004), pp. 139–160 (cit. on p. 132).