Algorithms for Nonconvex Optimization Problems in Machine Learning and Statistics

Zur Erlangung des akademischen Grades
eines Doktors der Wirtschaftswissenschaften

Dr. rer. pol.

von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

M. Sc. Robert Mohr

Tag der mündlichen Prüfung:	11.03.2020
Referent:	Prof. Dr. Oliver Stein
Korreferent:	Prof. Dr. Oliver Grothe

Contents

List of Figures

List of Tables

Abbreviations

Part I

ARC	Adaptive regularization method with cubics
ASTR	Adaptive subset trust-region method
SGD	Stochastic gradient descent method
SVRG	Stochastic variance reduced gradient method
TR	Trust-region method

Part II

B&B	Branch-and-bound algorithm
CF	Combinatorial formulation of the continuous least-squares spline approximation problem with free knots
CF	Combinatorial formulation
CF+FR	Fletcher-Reeves nonlinear conjugate gradient method initialized with the solution to the combinatorial formulation
CG	Continuous genetic algorithm
FR	Fletcher-Reeves nonlinear conjugate gradient method initialized with the equidistant knot placement
LP	Linear optimization problem
MILP	Mixed-integer linear optimization problem
MIQCP	Mixed-integer quadratically constrained optimization problem

Notation

General Mathematical Notation

$\mathbb{N}$	Natural numbers $1, 2, 3 \dots$
$\mathbb{R}^n$	Real coordinate space of n dimensions
$[a, b]$	Line segment between the vectors $a, b \in \mathbb{R}^n$
$\mathbb{R}^{m \times n}$	Space of real matrices with m rows and n columns
$\mathbb{S}^n$	Space of real symmetric matrices of dimension n
$\langle x, y \rangle$	Canonical inner product $\langle x, y \rangle = x^\intercal y$ in $\mathbb{R}^n$
$\|x\|_2$	Euclidean norm of a vector $x \in \mathbb{R}^n$
$\|A\|_2$	Spectral norm of a matrix $A \in \mathbb{R}^{m \times n}$
$\|S\|$	Cardinality of the set S
$S \setminus T$	Set difference of S and T
$C^2(\mathbb{R}^n, \mathbb{R})$	Space of twice continuously differentiable functions with domain $\mathbb{R}^n$ and codomain $\mathbb{R}$
∇f	Gradient of the differentiable function $f : \mathbb{R}^n \to \mathbb{R}$
$D^2 f$	Hessian of the twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$

Part I

n	Number of data points
m	Dimension of the decision variable
F	Objective function of the finite-sum problem
f_i	Functions summed up in objective function F
S	Sample of data points
$S^{\nu,k}$	Sample of data points in iteration ν, k
$S_H^{\nu,k}$	Sample of data points used for the curvature approximation in iteration ν, k
s^ν	Sample size in iteration ν
F_S	Sampled objective function
$F^{\nu,k}$	Shorthand for $F_{S^{\nu,k}}$
x^ν	Outer iterate in iteration ν
$x^{\nu,k}$	Inner iterate in iteration ν, k
$q^{\nu,k}(\cdot)$	Quadratic model function of $F^{\nu,k}$ at x^ν
$g^{\nu,k}$	Shorthand for $\nabla F^{\nu,k}(x^{\nu,k})$

Part I (Continued)

$A^{\nu,k}$	Matrix containing curvature information of F
$\rho^{\nu,k}$	Ratio of actual to predicted improvement
δ^{ν}	Trust-region radius
R^{ν}	Number of inner iterations
$\widehat{R}$	Upper limit on the number of inner iterations
θ	Parameter controlling the sample size adjustment
a^{ν}	Progress on the objective function F in iteration
b^{ν}	Average progress during the inner iterations on F_S
$d^{\nu,k}$	Trust-region step in iteration ν, k

Part II

n	Number of data points
k	Number of free knots
β	Vector of coefficients of a spline function
ξ	Knots of a spline function
$s(\cdot, \beta, \xi)$	Spline function with parameter vector β and knot vector ξ
$p(\cdot, \beta)$	Cubic polynomial with parameter vector β
$p'(\cdot, \beta)$	First derivative of $p(\cdot, \beta)$
$p''(\cdot, \beta)$	Second derivative of $p(\cdot, \beta)$
I_j	Set of indices of data points assigned to the jth polynomial
I	Set of unassigned indices
X_j	Vandermonde matrix corresponding to the data points with indices in I_j
z_{ij}	Binary variable that is one if data point i is assigned to the jth polynomial
Λ_j	jth restriction set $\Lambda_j \subseteq \{1, ..., n\}$
$\max(\Lambda_j)$	Largest element in Λ_j
$\min(\Lambda_j)$	Smallest element in Λ_j

Chapter 1

Thesis Overview

Mathematical optimization is one of the pillars of machine learning and modern statistics. Optimization algorithms are used to compute parameters of model functions such that available data is approximated optimally with respect to some error criterion. The numerical efficiency of these optimization algorithms is of key importance concerning the overall effectiveness of many of the techniques used to tackle problems in data science.

The purpose of this thesis is the design of algorithms that can be used to determine optimal solutions to nonconvex data approximation problems. If such a problem is large-scale, i.e. the decision variable is high dimensional and/or the number of data points is large, the computation of globally optimal solutions is usually not a realistic goal. Instead, local optimization algorithms can be used to efficiently approximate locally optimal solutions. And despite the fact that in the nonconvex case locally optimal solutions can be far from globally optimal, there are relevant large-scale problem classes in machine learning, e.g., the training of neural networks, for which local optimization algorithms are very effective empirically and therefore very popular. Naturally, there also exist nonconvex data approximation problems where local optimization algorithms get stuck in suboptimal local solutions if they are not initialized in a sufficiently small neighborhood of the global solution. Here it is necessary to develop algorithms that compute solutions close to the globally optimal solution, such that theses solutions can be used as starting points in local optimization algorithms.

In Part I of this thesis, we consider a very general class of nonconvex and large-scale data approximation problems and devise an algorithm that efficiently computes locally optimal solutions to these problems. After the introduction and literature review in Chapter 2, we present our algorithm in Chapter 3. In contrast to most algorithms proposed in the machine learning literature, our algorithm is a second-order method, i.e., it incorporates curvature information of the objective function in the computation of the search direction. As a type of trust-region Newton-CG method, the algorithm can make use of directions of negative curvature to escape saddle points, which otherwise might slow down the optimization process when solving nonconvex problems. In Chapter 4 we present results of numerical experiments on convex and nonconvex problems which support our claim that our algorithm has significant advantages compared to current state-of-the-art methods. Part I of this thesis is based on the preprint Mohr and Stein (2019).

In Part II we consider a specific nonconvex data approximation problem which is known to possess a large number of locally minimal points far from the globally optimal solution, namely the univariate least-squares spline approximation problem with free knots. In Chapter 5 we introduce the optimization problem and review existing approaches for its solution. Moreover, we show that local optimization algorithms, like the one presented in the first part of this thesis, should not be expected to yield satisfactory solutions for this problem if the initial point is not chosen sufficiently close to a globally optimal solution. However, since in typical applications, neither the dimension of the decision variable nor the number of data points is particularly large, it is possible to make use of the specific problem structure in order to devise algorithmic approaches to approximate the globally optimal solution of problem instances of relevant sizes. In Chapter 6 we propose to approximate the continuous original problem with a combinatorial optimization problem and, as a first algorithmic approach for the solution of the latter, we present a convex mixed-integer formulation that can be solved with commercial optimization solvers. As an alternative algorithmic approach, we propose a branch-and-bound method in Chapter 7, which is tailored specifically to the combinatorial problem formulated in the previous chapter. In Chapter 8 we present numerical experiments on real and synthetic data which show that the combinatorial approach to the least-squares spline approximation problem with free knots makes it possible to compute high-quality solutions to problems of realistic sizes within reasonable computing times. Part II of this thesis is based on joint work with Dr. Maximilian Coblenz and Dr. Peter Kirst.

Parts I and II of this thesis are concerned with the same basic problem of data approximation. However, the optimization methods that are proposed range from randomized local methods for continuous problems to deterministic global methods for discrete problems. This shows that when dealing with the difficult problem of nonconvexity in data science applications, it is important to make use of diverse tools from mathematical optimization in order to obtain high-quality solutions.

Part I

A Trust-Region Method for the Local Solution of Large-Scale Finite-Sum Problems

Chapter 2

Introduction

In this chapter, we first describe how the finite-sum minimization problem arises naturally in machine learning applications. Then we review existing algorithmic approaches to the large-scale finite sum minimization problem, before giving an overview of our approach.

2.1 Machine Learning and Finite-Sum Minimization

There exists a variety of applications from statistics and machine learning that require the minimization of an objective function that is the sum of a large number of convex or nonconvex functions. Well known examples are logistic regression problems, the training of neural networks and nonlinear least-squares problems. In these applications, the number of functions summed up in the objective function typically corresponds to the number of data points considered.

A typical task in supervised machine learning is prediction. Suppose we are given a data set $(x^i, y^i) \in \mathbb{R}^r \times \mathbb{R}^z$, $i = 1, \dots, n$, and a family of prediction functions $h(\cdot, w) : \mathbb{R}^r \to \mathbb{R}^z$ parametrized with a vector $w \in \mathbb{R}^m$. The goal is to find a parameter $\bar{w}$ such that the probability is high that if $(\bar{x}, \bar{y})$ is a new and so far unseen data point, we have that $h(\bar{x}, \bar{w}) \approx \bar{y}$. This is referred to as *generalization* in the machine learning literature. However, since we have no information apart from the given data set, the most natural way to approach this task is to compute the global minimizer of the finite-sum problem

$$\min_{w \in \mathbb{R}^m} \sum_{i=1}^{n} \ell(h(x^i, w), y^i), \tag{2.1}$$

where $\ell : \mathbb{R}^z \times \mathbb{R}^z \to \mathbb{R}$ is an error function that quantifies the discrepancy between $h(x^i, w)$ and the actual value y^i corresponding to x^i. In the machine learning literature, ℓ is typically called a *loss function* and problem (2.1) is referred to as the *empirical risk minimization problem.*

2.2 Optimization Algorithms in Large-Scale Machine Learning

Successful algorithms from classical nonlinear optimization, such as quasi-Newton, nonlinear conjugate-gradient and trust-region methods, usually require the computation of the gradient and (approximate) Hessian of the objective function in every iteration. If the number of data points is very large, these computations are expensive and prohibit fast progress in the early stages of the optimization process.

Popular methods therefore use single data points or samples of data points (so-called mini-batches) in order to obtain approximate information about the objective function. Arguably the most well-known and successful algorithms in this area are the stochastic gradient descent method, which was first proposed by Robbins and Monro (1951), and its variance-reduced variants (e.g., Schmidt et al. 2017, Defazio et al. 2014, Johnson and Zhang 2013), in which single data points or mini-batches are used in order to approximate the gradient of the objective function. We refer the interested reader to the excellent surveys by Bottou et al. (2018) and Curtis and Scheinberg (2017) for details concerning these methods. However, these methods have two major drawbacks. Firstly, extensive experimentation is needed for every new problem and data set in order to find hyper-parameters (e.g., the step-size) that lead to a good performance of these methods. Secondly, since only first-order information is employed, their ability to make progress in the presence of saddle points or to deal with ill-conditioned problems is limited.

In the last few years there has been growing interest in algorithms that speed up the minimization of large-scale finite-sum problems by incorporating approximate second-order information via sampling. In particular, stochastic Newton, Gauss-Newton and (limited-memory) BFGS methods were developed (e.g., Byrd et al. 2011, Roosta-Khorasani and Mahoney 2019, Bollapragada et al. 2018b, Martens 2010, Martens and Sutskever 2011, Schraudolph et al. 2007, Bordes et al. 2009, Sohl-Dickstein et al. 2014, Mokhtari et al. 2015, Byrd et al. 2016, Berahas et al. 2016, Curtis 2016, Gower et al. 2016, Zhou et al. 2017, Bollapragada et al. 2018c, Berahas and Takáč 2019). However, despite promising theoretical and empirical results, most of the proposed methods still depend on extensive hyper-parameter tuning for each new problem and data set. Moreover, all of these methods work with positive definite curvature approximations and experience numerical instability when these matrices become close to singular. However, in the context of nonconvex optimization, it was demonstrated by Curtis and Robinson (2019) that incorporating directions of negative curvature can be beneficial and, according to Dauphin et al. (2014), they might help to escape saddle points more quickly.

In this thesis, we propose a trust-region method that can be applied to large-scale nonconvex finite sum minimization. The reason why we propose a trust-region method is that it is very flexible with respect to the type of approximate curvature information that can be used, and can exploit directions of negative curvature. In the next section we discuss the existing literature on trust-region methods for the finite-sum minimization problem.

2.3 Trust-Region Methods for Finite-Sum Minimization

Along with nonlinear conjugate gradient and quasi-Newton methods, trust-region algorithms belong to the most reliable and efficient algorithms for the local minimization of general nonlinear functions. Theoretical results concerning global convergence properties of classical trust-region methods, as well as practical considerations, can be found in the books by Conn et al. (2000) and Nocedal and Wright (2006) and the survey paper by Yuan (2015).

From a practical point of view, trust-region algorithms for the finite-sum minimization problem proposed so far can be broadly classified into three groups, depending on how sampling is used in order to obtain approximate information about the objective function. Members of the first group evaluate the objective function and its gradient exactly in each iteration, while using a sample of the data points to determine approximate curvature information (e.g., Xu et al. 2019, 2017). In addition to approximating curvature information, methods that belong to the second group also approximate the gradient based on a (possibly different) sample, while still evaluating the objective function exactly in every iteration (e.g., Gratton 2017, Yao et al. 2018, Erway et al. 2019). The last group contains methods that, at least in the early stages of the optimization process, only work with inexact information about the objective function based on samples, i.e., the objective function is evaluated inexactly as well (e.g., Chen et al. 2018, Bellavia et al. 2018, Blanchet et al. 2019).

The main idea underlying methods from the first group is that the most expensive step in each iteration of a trust-region method is the (approximate) solution of the trust-region subproblem, at least if nontrivial curvature approximations are employed. This cost can be greatly reduced if the curvature information is approximated based on a small sample of the data points. The global convergence to first order critical points is covered by results on standard trust-region methods. However, since the objective function and its gradient are evaluated exactly in each iteration, the behavior of these methods is more similar to deterministic than to randomized methods.

This drawback also applies to the methods of the second group. In typical finite-sum problems from machine learning and statistics, the evaluation of the objective function is about half as expensive as the computation of the gradient. Therefore, although methods that approximate the gradient can be more efficient than methods that use the exact gradient, the progress of these methods will be slow in the early stages of the optimization process as long as the objective function is evaluated in every iteration.

In contrast, methods from the last group can achieve very low per iteration costs if the samples used for the approximations are sufficiently small. However, in order to obtain a convergent method, the objective function and its gradient have to be approximated with increasing accuracy. In the context of the finite-sum minimization, this necessitates increasing the corresponding sample size during the optimization process, which is referred to as dynamic/adaptive sampling or progressive batching. This technique leads to hybrid deterministic-stochastic methods, i.e., methods that start off as randomized methods and eventually turn into deterministic methods.

In these methods, the sample size can either be increased at a preset rate or adaptively according to information obtained during the optimization process. Promising theoretical and empirical results were obtained for the stochastic gradient descent and the stochastic L-BFGS methods (e.g., Friedlander and Schmidt 2012, Byrd et al. 2012, De et al. 2017, Bollapragada et al. 2018a,b,c). In the context of trust-region methods, adaptive rules for adjusting the sample size so far either depend on unknown quantities or require experimentation by the user in order to obtain good performance.

2.4 Overview of our Approach

We propose a trust-region method for finite-sum minimization with an adaptive sample size adjustment technique, which is practical in the sense that it leads to a globally convergent method that shows strong performance empirically without the need for experimentation by the user. During the optimization process, the size of the samples is adaptively increased (or decreased) depending on the progress made on the objective function. We prove that after a finite number iterations the sample includes all points from the data set and the method becomes a full-batch trust-region method. Numerical experiments on convex and nonconvex problems support our claim that our algorithm has significant advantages compared to stochastic gradient descent and its variance-reduced versions.

We note that the technique we propose for sample size adjustment could also be used in conjunction with the adaptive regularization method with cubics (ARC) proposed by Cartis et al. (2011a) and Cartis et al. (2011b). The ARC method is an adaptive version of the cubic regularization method first introduced by Griewank (1981). It was shown by Nesterov and Polyak (2006) and Cartis et al. (2011b) that cubic regularization methods and their adaptive variants are, from a worst-case complexity point of view, superior to classical trust-region methods. This fact lead to increased research interest in stochastic variants of these methods (e.g., Kohler and Lucchi 2017, Xu et al. 2017, Cartis and Scheinberg 2018). However, we chose to propose a trust-region method since it was observed in Xu et al. (2017) that they tend to show stronger empirical performance than ARC methods.

Chapter 3

An Adaptive Sample Size Trust-Region Method

In the first section of this chapter, we present our trust-region method for the finite-sum minimization problem. Then we provide a theoretical analysis which shows that our adaptive sample size adjustment technique eventually increases the sample size to include all data points, which implies global convergence. In the final section we provide some guidance concerning the practical implementation of the method.

3.1 The ASTR-Algorithm

We call our method **A**daptive **S**ample Size **T**rust-**R**egion method, or ASTR for short. It is specifically designed to solve the finite-sum minimization problem

$$\min_{x \in \mathbb{R}^m} F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

where $n, m \in \mathbb{N}$ and $f_i \in C^2(\mathbb{R}^m, \mathbb{R})$ for all $i = 1, \ldots, n$. The method consists of outer and inner iterations, shown in Algorithm 3.1 and 3.2, respectively. In every inner iteration, a sample $S \subseteq \{1, \ldots, n\}$ is chosen and Algorithm 3.3 is used to compute a trust-region step for the function

$$F_S := \frac{1}{|S|} \sum_{i \in S} f_i.$$

After a certain number of inner iterations of Algorithm 3.2, a candidate for the next outer iterate is returned to Algorithm 3.1. There, the candidate is either accepted or rejected and the sample size is adjusted. We now describe the three algorithms in detail.

3.1.1 Outer Iterations

In iteration ν of Algorithm 3.1, Algorithm 3.2 is called with the current iterate x^ν, sample size s^ν, initial trust-region radius δ^ν and number of inner iterations R^ν as input arguments. It returns to Algorithm 3.1 a candidate for the next iterate $\widehat{x}^\nu$ and a prediction b^ν of the improvement in the objective function value if $\widehat{x}^\nu$ is accepted. Moreover, a value $\delta^{\nu+1}$ is returned, which is passed as the initial trust-region radius to Algorithm 3.2 in the next iteration of Algorithm 3.1.

The candidate $\widehat{x}^\nu$ is accepted if a^ν, the improvement in the objective function value, is nonnegative. Note that the computation of a^ν is an expensive operation if n is large, since F needs to be evaluated at $\widehat{x}^\nu$. The new sample size $s^{\nu+1}$ is chosen depending on the size of the ratio τ^ν of actual to predicted improvement. A small value of τ^ν indicates that the sampled functions used in the inner iterations do not approximate F accurately enough and that the sample size should therefore be increased. A large value of τ^ν, however, is an indicator that faster progress in the inner iterations might be possible if the sample size is decreased.

Note that every time the sample size is increased/decreased in the outer iteration, the inner iterations get more computationally expensive/cheap and the number of inner iterations should therefore also be decreased/increased. In Section 3.3 we explain how to update the sample size and the number of inner iterations in order to obtain a method with strong empirical performance.

3.1.2 Inner Iterations

In iteration k of Algorithm 3.2, a sample $S^{\nu,k}$ of size s^ν is chosen. For notational convenience, we define $F^{\nu,k} := F_{S^{\nu,k}}$ and $g^{\nu,k} := \nabla F^{\nu,k}(x^{\nu,k})$. If the gradient $g^{\nu,k}$ does not satisfy $\|g^{\nu,k}\|_2 \geq \varepsilon$, where ε is a preset threshold, no step is taken and a new sample is selected in the next iteration. Otherwise, a trust-region step $d^{\nu,k}$ is computed and the initial trust-region radius $\delta^{\nu,k+1}$ for the next iteration is determined with Algorithm 3.3.

After R iterations, the last inner iterate $x^{\nu,R}$ and the current trust-region radius $\delta^{\nu,R}$ are returned to Algorithm 3.1. Additionally, the average improvement on the sampled functions during the inner iterations

$$b^\nu := \frac{1}{R}\sum_{k=0}^{R-1} b^{\nu,k} = \frac{1}{R}\sum_{k=0}^{R-1}\Big(F^{\nu,k}(x^{\nu,k}) - F^{\nu,k}(x^{\nu,k+1})\Big)$$

is returned to Algorithm 3.1 as a prediction for the improvement on the objective function F if $x^{\nu,R}$ is accepted as the next outer iterate.

Algorithm 3.1 ASTR - Outer Iterations

```
 1: Input: Initial point $x^0 \in \mathbb{R}^m$, parameters $s^0, \widehat{R} \in \mathbb{N}$ and $\delta^0, \theta > 0$;
 2: for $\nu = 0, 1, \dots$ do
 3:     Select the number of inner iterations $R^\nu \in \{1, \dots, \widehat{R}\}$;
 4:     Compute $\widehat{x}^\nu$, $b^\nu$ and $\delta^{\nu+1}$ via Algorithm 3.2 with inputs $x^\nu, s^\nu, \delta^\nu, R^\nu$;
 5:     if $s^\nu < n$ then
 6:         Set $a^\nu = F(x^\nu) - F(\widehat{x}^\nu)$;
 7:         if $a^\nu \geq 0$ then set $x^{\nu+1} = \widehat{x}^\nu$ else set $x^{\nu+1} = x^\nu$;
 8:         if $b^\nu > 0$ then set $\tau^\nu = a^\nu / b^\nu$ else set $\tau^\nu = 0$;
 9:         if $\tau^\nu < \theta$ then
10:             Select $s^{\nu+1} \in \{s^\nu + 1, \dots, n\}$;
11:         else
12:             Select $s^{\nu+1} \in \{1, \dots, s^\nu\}$;
13:         end if
14:     else
15:         Set $x^{\nu+1} = \widehat{x}^\nu$ and $s^{\nu+1} = n$;
16:     end if
17: end for
```

3.1.3 Trust-region step

In iteration r of Algorithm 3 an (approximate) solution d^r to the trust-region subproblem

$$\min_{d \in \mathbb{R}^m} q^{\nu,k}(d) \quad \text{s.t.} \quad \|d\|_2 \leq \Delta_r, \tag{3.1}$$

is computed, where $q^{\nu,k}$ is a (quadratic) model of $F^{\nu,k}$ at $x^{\nu,k}$ defined as

$$q^{\nu,k}(d) := F^{\nu,k}(x^{\nu,k}) + \langle g^{\nu,k}, d \rangle + \frac{1}{2} d^\intercal A^{\nu,k} d,$$

and Δ_r is the current trust-region radius. The matrix $A^{\nu,k} \in \mathbb{S}^m$ can be used to include curvature information in the model. However, it is also possible to only use first-order information by setting $A^{\nu,k} = 0$.

An (approximate) solution d^r to problem (3.1) is accepted if the ratio

$$\rho^{\nu,k}(d^r) := \frac{F^{\nu,k}(x^{\nu,k} + d^r) - F^{\nu,k}(x^{\nu,k})}{q^{\nu,k}(0) - q^{\nu,k}(d^r)} \tag{3.2}$$

of the actual improvement on $F^{\nu,k}$ to the improvement predicted by the quadratic model is above a certain threshold η_1. The intuition behind this is that if the ratio $\rho^{\nu,k}(d^r)$ is above this threshold, this is an indication that

Algorithm 3.2 ASTR - Inner Iterations

```
1: Input: $x^\nu, \delta^\nu, s^\nu$ and $R^\nu$ from Algorithm 3.1 and parameter $\varepsilon > 0$;
2: Set $x^{\nu,0} = x^\nu$, $\delta^{\nu,0} = \delta^\nu$ and $R = R^\nu$;
3: for $k = 0, 1, \dots, R-1$ do
4:     Choose a sample $S^{\nu,k} \subseteq \{1, \dots, n\}$ of size $s^\nu$;
5:     if $\|g^{\nu,k}\|_2 \geq \varepsilon$ then
6:         Compute $d^{\nu,k}, \delta^{\nu,k+1}$ via Algorithm 3.3 with inputs $x^{\nu,k}$, $g^{\nu,k}$, $\delta^{\nu,k}$;
7:         Set $x^{\nu,k+1} = x^{\nu,k} + d^{\nu,k}$;
8:         Set $b^{\nu,k} = F^{\nu,k}(x^{\nu,k}) - F^{\nu,k}(x^{\nu,k+1})$;
9:     else
10:        Set $x^{\nu,k+1} = x^{\nu,k}$, $b^{\nu,k} = 0$ and $\delta^{\nu,k+1} = \delta^{\nu,k}$;
11:    end if
12: end for
13: Output: $\widehat{x}^\nu = x^{\nu,R}$, $b^\nu = \frac{1}{R}\sum_{k=0}^{R-1} b^{\nu,k}$, $\delta^{\nu+1} = \delta^{\nu,R}$;
```

the model $q^{\nu,k}$ is a good approximation of $F^{\nu,k}$ on the feasible set of problem (3.1) (the so called “trust-region”). As long as the ratio (3.2) is smaller then η_1, the trust-region radius is decreased and a new (approximate) solution of the trust-region subproblem is computed. Since we use the exact gradient of $F^{\nu,k}$ in our model $q^{\nu,k}$, it is always possible to find an acceptable (approximate) solution to the trust-region subproblem if the trust-region radius is sufficiently small (see Theorem 3.2.5). Note that if the ratio (3.2) is not only larger than η_1 but also larger than η_2, then the trust-region radius is increased such that larger steps might be taken in the next inner iteration.

3.2 Theoretical Analysis

In this section we prove that after a finite number of iterations, the sample size s^ν reaches n and the ASTR method becomes a full-batch trust-region method.

3.2.1 Assumptions

Assumption 3.2.1 *The function F is bounded below on $\mathbb{R}^m$, i.e., there exists a constant $\kappa_1 \in \mathbb{R}$ such that $F(x) \geq \kappa_1$ for all $x \in \mathbb{R}^m$.*

Assumption 3.2.2 *The functions f_i, $i = 1, \dots, n$, are twice continuously differentiable and their gradients ∇f_i are Lipschitz continuous.*

Algorithm 3.3 Trust-region step

```
1: **Input:** $x^{\nu,k}$, $g^{\nu,k}$ and $\delta^{\nu,k}$ from Algorithm 3.2 and parameters $\eta_1, \eta_2 \in (0,1)$ and $0 < \gamma_1 < 1 < \gamma_2$;
2: Set $r = -1$, $\Delta_0 = \delta^{\nu,k}$;
3: **repeat**
4:     $r \leftarrow r + 1$;
5:     Compute an (approximate) solution $d^r$ of problem
           $$\min_{d \in \mathbb{R}^m} q^{\nu,k}(d) \quad \text{s.t.} \quad \|d\|_2 \leq \Delta_r;$$
6:     Compute $\rho^{\nu,k}(d^r) := \frac{F^{\nu,k}(x^{\nu,k} + d^r) - F^{\nu,k}(x^{\nu,k})}{q^{\nu,k}(0) - q^{\nu,k}(d^r)}$;
7:     Set $\Delta_{r+1} = \gamma_1 \|d^r\|_2$;
8: **until** $\rho^{\nu,k}(d^r) < \eta_1$
9: **if** $\rho^{\nu,k}(d^r) \geq \eta_2$ and $\|d^r\|_2 = \Delta_r$ **then** set $\Delta_+ = \gamma_2 \Delta_r$ **else** set $\Delta_+ = \Delta_r$;
10: **Output:** $d^{\nu,k} = d^r$ and $\delta^{\nu,k+1} = \Delta_+$;
```

Note that Assumption 3.2.2 implies that the Hessians $D^2 f_i(x)$ are uniformly bounded in x for all i. From the triangular inequality it immediately follows that the Hessian of the function F_S is uniformly bounded in x and S, i.e., there exists a constant $\kappa_2 > 0$ such that the inequality

$$\|D^2 F_S(x)\|_2 \leq \kappa_2 \tag{3.3}$$

holds for any $S \subseteq \{1, \dots, n\}$ and $x \in \mathbb{R}^m$.

Assumption 3.2.3 *There exists a constant $\kappa_3 > 0$ such that for all ν, k we have that $\|A^{\nu,k}\|_2 \leq \kappa_3$.*

Assumption 3.2.3 is trivially satisfied if $A^{\nu,k} = 0$ for all ν, k. Due to (3.3) we know that Assumption 3.2.3 is also satisfied if we set $A^{\nu,k} = D^2 F_S(x)$ for any S and x.

Assumption 3.2.4 *There exists a constant $\kappa_4 \in (0,1)$ such that for all ν, k, r we have that*

$$q^{\nu,k}(0) - q^{\nu,k}(d^r) \geq \kappa_4 \|g^{\nu,k}\|_2 \min\left(\frac{\|g^{\nu,k}\|_2}{1 + \|A^{\nu,k}\|_2}, \Delta_r\right).$$

If d^r is a sufficiently accurate approximation of the exact solution of the trust-region subproblem, Assumption 3.2.4 is satisfied. Moreover, there exists a variety of methods for the inexact solution of the trust-region subproblem such that Assumption 3.2.4 is satisfied, e.g., the truncated conjugate gradient (CG) method by Toint (1981) and Steihaug (1983) or the truncated Lanczos method by Gould et al. (1999).

3.2.2 Theoretical Results

The first two theorems presented in this section are modifications of well known results in the literature on trust-region methods, see, e.g., Theorems 6.4.2 and 6.4.3 in Conn et al. (2000). We include the modified proofs of these two theorems for completeness.

Theorem 3.2.5 *Suppose A3.2.2, A3.2.3 and A3.2.4 hold. Then there exists a constant $\kappa_5 > 0$ such that for all ν, k and r the inequality $\|d^r\|_2 \leq \kappa_5$ implies that $\rho^{\nu,k}(d^r) \geq \eta_1$ holds.*

Proof. Define the constant

$$\kappa_5 := \frac{(1-\eta_1)\kappa_4 \varepsilon}{1+\kappa_2+\kappa_3}$$

and assume that $\|d^r\|_2 \leq \kappa_5$ holds. From the definition of $\rho^{\nu,k}(d^r)$ it follows that

$$\begin{aligned} 1-\rho^{\nu,k}(d^r) &= 1 - \frac{F^{\nu,k}(x^{\nu,k}) - F^{\nu,k}(x^{\nu,k}+d)}{q^{\nu,k}(0) - q^{\nu,k}(d^r)} \\ &= \frac{F^{\nu,k}(x^{\nu,k}+d) - q^{\nu,k}(d^r)}{q^{\nu,k}(0) - q^{\nu,k}(d^r)}, \end{aligned} \tag{3.4}$$

where we used that $q^{\nu,k}(0) = F^{\nu,k}(x^{\nu,k})$. From Assumption 3.2.4 we know that

$$q^{\nu,k}(0) - q^{\nu,k}(d^r) \geq \kappa_4 \|g^{\nu,k}\|_2 \min(\frac{\|g^{\nu,k}\|_2}{1+\|A^{\nu,k}\|_2}, \Delta_r),$$

where Δ_r denotes the trust-region radius in the subproblem that was used to compute the trust-region step d^r. With $\Delta_r \geq \|d^r\|_2$, $\|g^{\nu,k}\|_2 \geq \varepsilon$ and Assumption 3.2.3 we obtain

$$q^{\nu,k}(0) - q^{\nu,k}(d^r) \geq \kappa_4 \varepsilon \min(\frac{\varepsilon}{1+\kappa_3}, \|d^r\|_2).$$

Moreover, due to $\eta_1, \kappa_4 \in (0,1)$ and $\kappa_2 \geq 0$ we know that $\|d^r\|_2 \leq \kappa_5$ implies $\|d^r\|_2 \leq \frac{\varepsilon}{1+\kappa_3}$. Thus, we have

$$q^{\nu,k}(0) - q^{\nu,k}(d^r) \geq \kappa_4 \varepsilon \|d^r\|_2.$$

This inequality together with (3.4) yields

$$1 - \rho^{\nu,k}(d^r) \leq \frac{F^{\nu,k}(x^{\nu,k} + d^r) - q^{\nu,k}(d^r)}{\kappa_4 \varepsilon \|d^r\|_2}.$$

Now, it follows from Taylor's theorem that for some λ in the line segment $[x^{\nu,k}, x^{\nu,k} + d]$ it holds that

$$F^{\nu,k}(x^{\nu,k} + d^r) = F^{\nu,k}(x^{\nu,k}) + \langle g^{\nu,k}, d^r \rangle + \frac{1}{2}(d^r)^\intercal D^2 F^{\nu,k}(\lambda) d^r$$

and together with Assumptions 3.2.2 and 3.2.3 we obtain

$$\begin{aligned} F^{\nu,k}(x^{\nu,k} + d^r) - q^{\nu,k}(d^r) &= \frac{1}{2}(d^r)^\intercal D^2 F^{\nu,k}(\lambda) d^r - \frac{1}{2}(d^r)^\intercal A^{\nu,k} d^r \\ &\leq \frac{1}{2}\left(\|D^2 F^{\nu,k}(\lambda)\|_2 + \|A^{\nu,k}\|_2\right) \|d^r\|_2^2 \\ &\leq (\kappa_2 + \kappa_3)\|d^r\|_2^2. \end{aligned}$$

Thus, we have that

$$\begin{aligned} 1 - \rho^{\nu,k}(d^r) &\leq \frac{(\kappa_2 + \kappa_3)}{\kappa_4 \varepsilon}\|d^r\|_2 \leq \frac{(\kappa_2 + \kappa_3)}{\kappa_4 \varepsilon}\kappa_5 = \frac{(\kappa_2 + \kappa_3)}{\kappa_4 \varepsilon}\frac{(1 - \eta_1)\kappa_4 \varepsilon}{1 + \kappa_2 + \kappa_3} \\ &= \frac{(\kappa_2 + \kappa_3)}{1 + \kappa_2 + \kappa_3}(1 - \eta_1) \leq 1 - \eta_1, \end{aligned}$$

and therefore $\rho^{\nu,k}(d^r) \geq \eta_1$. □

The previous theorem guarantees that Algorithm 3.3 terminates after a finite number of steps. Moreover, it is instrumental in proving the following result.

Theorem 3.2.6 *Suppose A3.2.2, A3.2.3 and A3.2.4 hold. Then there exists a constant $\kappa_6 > 0$ such that $\delta^{\nu,k} \geq \kappa_6$ holds for all ν, k.*

Proof. Define the constant

$$\kappa_6 := \min\left(\frac{\gamma_1(1 - \eta_1)\kappa_4 \varepsilon}{1 + \kappa_2 + \kappa_3}, \delta^{0,0}\right).$$

Assume, for the purpose of deriving a contradiction, that (ν, k) is the first iteration such that $\delta^{\nu,k} < \kappa_6$. Since $\kappa_6 \leq \delta^{0,0}$ we have $(\nu, k) \neq (0, 0)$. Moreover, due to $\delta^{\nu-1,R} = \delta^{\nu,0}$ we have $k \geq 1$. The value $\delta^{\nu,k}$ is calculated via Algorithm 3.3 with $\Delta_0 = \delta^{\nu,k-1}$ as the initial trust-region. Since (ν, k) is the first iteration such that $\delta^{\nu,k} < \kappa_6$, we know that $\Delta_0 = \delta^{\nu,k-1} \geq \kappa_6$. Since $\delta^{\nu,k} = \Delta_+$ and $\Delta_+ \geq \Delta_r$ it follows that $\Delta_r < \kappa_6$. Thus, there must exist a smallest index $j \in \{1, \dots, r\}$ such that $\Delta_j < \kappa_6$. Clearly, if j is the first iteration such that $\Delta_j < \kappa_6$ holds, it must hold that $\rho^{\nu,k}(d^{j-1}) < \eta_1$. Consequently, we have that

$$\Delta_j = \gamma_1 \|d^{j-1}\|_2$$

and thus

$$\|d^{j-1}\|_2 = \frac{\Delta_j}{\gamma_1} \leq \frac{\kappa_6}{\gamma_1} \leq \frac{(1-\eta_1)\kappa_4 \varepsilon}{1+\kappa_2+\kappa_3} = \kappa_5,$$

with κ_5 as defined in the proof of Theorem 3.2.5. From Theorem 3.2.5 it now follows that the inequality $\rho^{\nu,k}(d^{j-1}) \geq \eta_1$ holds, which is a contradiction since we already argued that $\rho^{\nu,k}(d^{j-1}) < \eta_1$. □

Theorem 3.2.7 *Suppose A3.2.2, A3.2.3 and A3.2.4 hold. Then there exists a constant $\kappa_7 > 0$ such that for all ν either $b^\nu = 0$ or $b^\nu \geq \kappa_7$ holds.*

Proof. For any k, if $\|g^{\nu,k}\|_2 < \varepsilon$, then $b^{\nu,k} = 0$. On the other hand, if $\|g^{\nu,k}\|_2 \geq \varepsilon$, then Theorem 3.2.5 guarantees that Algorithm 3.3 terminates with a trust-region step $d^{\nu,k}$ that satisfies $\rho^{\nu,k}(d^{\nu,k}) \geq \eta_1$. Thus, we obtain from (3.2) and Assumption 3.2.4 that

$$\begin{aligned} b^{\nu,k} &= F^{\nu,k}(x^{\nu,k}) - F^{\nu,k}(x^{\nu,k+1}) \geq \eta_1 \left(q^{\nu,k}(0) - q^{\nu,k}(d^{\nu,k})\right) \\ &\geq \eta_1 \kappa_4 \|g^{\nu,k}\|_2 \min\left(\frac{\|g^{\nu,k}\|_2}{1+\|A^{\nu,k}\|_2}, \Delta^{\nu,k}\right), \end{aligned}$$

where $\Delta^{\nu,k}$ denotes the trust-region radius in the subproblem that was used to compute the trust-region step $d^{\nu,k}$. From Theorem 3.2.6 it follows that

$$\Delta^{\nu,k} \geq \frac{\delta^{\nu,k}}{\gamma_2} \geq \frac{\kappa_6}{\gamma_2}$$

and from Assumption 3.2.3 we know that

$$\frac{\|g^{\nu,k}\|_2}{1+\|A^{\nu,k}\|_2} \geq \frac{\|g^{\nu,k}\|_2}{1+\kappa_3}.$$

We therefore have that

$$b^{\nu,k} \geq \eta_1 \kappa_4 \varepsilon \min\left(\frac{\varepsilon}{1+\kappa_3}, \frac{\kappa_6}{\gamma_2}\right).$$

Thus, since $b^\nu = \frac{1}{R^\nu} \sum_{k=0}^{R^\nu - 1} b^{\nu,k}$, we either have $b^\nu = 0$ or $b^\nu \geq \kappa_7$, with

$$\kappa_7 := \frac{\eta_1 \kappa_4 \varepsilon}{\widehat{R}} \min\left(\frac{\varepsilon}{1+\kappa_3}, \frac{\kappa_6}{\gamma_2}\right) > 0.$$

□

Concerning the previous theorem, we note that $b^\nu = 0$ can only occur if $\|g^{\nu,k}\|_2 < \varepsilon$ for all inner iterations.

Theorem 3.2.8 *Suppose A3.2.1, A3.2.2, A3.2.3 and A3.2.4 hold. Then there exists a $\nu_0 \in \mathbb{N}$ such that $s^\nu = n$ holds for all $\nu \geq \nu_0$.*

Proof. We show that there exists a $\nu_0 \in \mathbb{N}$ such that $s^{\nu_0} = n$ since this implies the assertion in the theorem. Assume, for the purpose of deriving a contradiction, that $s^\nu < n$ for all $\nu \in \mathbb{N}$. Since $\tau^\nu < \theta$ implies $s^{\nu+1} > s^\nu$, there does not exist a $\bar{\nu} \in \mathbb{N}$ such that $\tau^\nu < \theta$ for all $\nu > \bar{\nu}$. Thus, there exists a subsequence $(x^{\nu(j)})$ with $\tau^{\nu(j)} \geq \theta$ for all $j \in \mathbb{N}$. This implies that $b^{\nu(j)} > 0$ for all $j \in \mathbb{N}$. From Theorem 3.2.7 we now obtain that $b^{\nu(j)} > \kappa_7$ and therefore

$$F(x^{\nu(j)}) - F(\widehat{x}^{\nu(j)}) = a^{\nu(j)} \geq \theta b^{\nu(j)} \geq \theta\kappa_7$$

for all $j \in \mathbb{N}$. Since the sequence $(F(x^\nu))$ is monotonically nonincreasing we obtain for all $i \in \mathbb{N}$

$$\begin{aligned} F(x^0) - F(x^{\nu(i)+1}) &= \sum_{\nu=0}^{\nu(i)} (F(x^\nu) - F(x^{\nu+1})) \geq \sum_{j=0}^{i} (F(x^{\nu(j)}) - F(x^{\nu(j)+1})) \\ &= \sum_{j=0}^{i} (F(x^{\nu(j)}) - F(\widehat{x}^{\nu(j)})) \geq (i+1)\theta\kappa_7, \end{aligned}$$

and therefore

$$F(x^0) - F(x^{\nu(i)+1}) \overset{i\to\infty}{\to} +\infty,$$

in contradiction to Assumption 3.2.1. □

Theorem 3.2.8 ensures that after a finite number of iterations, the ASTR method becomes a standard (full-batch) trust-region method. It implies that global convergence of the ASTR method follows from the global convergence results about standard trust-region methods, e.g., Theorem 6.4.6 in Conn et al. (2000).

3.3 Practical Considerations

In the description of our algorithm in Section 3.1 we left out several details that do not need specification in order to prove the theoretical results in Section 3.2, but which are nonetheless important with regard to the practical implementation of the method. The purpose of this section is to close this gap.

3.3.1 Random Sampling and Sample Size Adjustment

We propose to select the samples in the inner iterations of our algorithm uniformly at random, although other selection strategies (deterministic and stochastic) are possible and may be worth investigating.

Friedlander and Schmidt (2012) and Byrd et al. (2012) showed that when the sample size is increased geometrically in stochastic gradient decent, then the expected optimality gap converges linearly . Inspired by this strategy, we propose to choose a constant $\omega > 1$ and set $s^{\nu+1} = \min(\lceil \omega s^{\nu} \rceil, n)$ whenever $\tau^{\nu} < \theta$. If $\tau^{\nu} \geq \theta$, one can simply set $s^{\nu+1} = s^{\nu}$, i.e., the sample size is never decreased. We leave the question of whether strategies for decreasing the sample size can lead to performance benefits for future research.

3.3.2 Incorporation of Curvature Information and the Solution of the Trust-Region Subproblems

For $A^{\nu,k} = 0$ the solution of the trust-region subproblem (3.1) is $d^r = -\Delta_r g^{\nu,k}/\|g^{\nu,k}\|_2$. Thus, if $A^{\nu,k} = 0$ for all ν, k, the ASTR method is a adaptive sample size gradient method.

It is one of the strengths of the ASTR method that any kind of curvature information can be used. However, one has to keep in mind that the choice of $A^{\nu,k}$ is tightly coupled with effort necessary to compute an (approximate) solution to the trust-region subproblem. Fortunately, the trust-region subproblem is a problem that has been studied for decades and one can choose from a wide variety of methods in order to determine exact or inexact solutions, see, for example, Conn et al. (2000). Consequently, there is a lot of flexibility for investigating different ways to incorporate curvature information.

The most straightforward way to incorporate curvature information is to set $A^{\nu,k} = D^2 F^{\nu,k}(x^{\nu,k})$ as soon as the sample size s^{ν} is considered large enough for the sampled Hessian to contain meaningful curvature information. If the decision variable is very high dimensional, the (sampled) Hessian of the objective is expensive to compute and might be too large to store. However, if the truncated CG method is used for the solution of the trust-region subproblems, only matrix-vector products of $A^{\nu,k}$ and certain vectors need to be computed. These matrix-vector products can be efficiently computed for various problems in supervised machine learning without ever forming the matrix $A^{\nu,k}$ explicitly, see Pearlmutter (1994). This technique is known as "Hessian-free" optimization.

Note that if this "Hessian-free" technique is used, the cost of multiplying $A^{\nu,k}$ with a vector depends on the size of the sample used for the computation of $A^{\nu,k}$. This cost can therefore be reduced by setting $A^{\nu,k} = D^2 F_{S_H^{\nu,k}}(x^{\nu,k})$ for some subsample $S_H^{\nu,k} \subseteq S^{\nu,k}$, provided that s^ν is large enough. In Xu et al. (2017) one can find some guidance on how to subsample the Hessian in a trust-region framework when exact gradient information is used.

Also note that for nonconvex problems, $A^{\nu,k}$ can be indefinite and directions of negative curvature can be exploited. This might be particularly useful in the proximity of saddle points, which are considered one of the main obstacles when training neural networks with current methods, see, for example, Dauphin et al. (2014).

3.3.3 Adjusting the Number of Inner Iterations

The last detail that needs to be specified is how the number of inner iterations $R^{\nu+1}$ should be adjusted depending on the updated sample size $s^{\nu+1}$. We suggest to select $R^{\nu+1}$ in a way such that the total computational cost in all the inner iterations combined is approximately equal to the cost of evaluating the objective function F in the outer iteration. Consequently, the number of inner iterations depends on the kind of curvature information used and the method for the solution of the trust-region subproblem.

To be more concrete: Assume that evaluating the objective function is half as expensive as computing the gradient, and that the computation of a Hessian-vector product costs approximately the same as the computation of a gradient. This is indeed the case for various applications, see Section 4 for some examples.

If $A^{\nu,k} = 0$ for some ν and all k, the solution of the trust-region subproblem (3.1) is given explicitly by $d^r = -\Delta_r g^{\nu,k}/\|g^{\nu,k}\|_2$. Thus, the cost of one inner iteration corresponds to the cost of evaluating the gradient $g^{\nu,k}$. Consequently, R^ν should satisfy the equation $R^\nu \cdot s^\nu/n = 0.5$ and we obtain $R^\nu = n/(2s^\nu)$ for the number of inner iterations of Algorithm 3.2.

If $A^{\nu,k} = D^2 F_{S_H^{\nu,k}}(x^{\nu,k})$ for some ν and all k, and the truncated CG method is used to approximately solve the trust-region subproblems, analogous reasoning leads to the formula $R^\nu = n/((2+\bar{\alpha}) \cdot s^\nu + \bar{\beta} \cdot 2 \cdot s_H^\nu)$, where $\bar{\alpha}$ denotes the average number of iterations of Algorithm 3, $\bar{\beta}$ denotes the average number of iterations the truncated CG method requires to find an approximate solution to the trust-region subproblems and $s_H^\nu := |S_H^{\nu,k}|$, where we assume that the size of the subsample $S_H^{\nu,k}$ is fixed during the inner iterations.

Chapter 4

Numerical Experiments

In this section we compare the ASTR method with a mini-batch stochastic gradient descent method (SGD), the SVRG method by Johnson and Zhang (2013) and a full-batch Trust-Region Newton-CG method (TR). We consider three classification problems: logistic regression (convex), nonlinear least-squares (nonconvex) and neural network training (nonconvex). For each problem and data set, we report the training errors of the methods against CPU time measurements. The algorithms were implemented in Python 3.6 and the computations were performed on an Intel Core i7-9700K with 32 GB of main memory.

In contrast to the SGD and SVRG methods, who depend on hyper-parameter tuning for reasonable performance, we did not perform hyper-parameter tuning to individual problems or data sets for the ASTR or TR methods, i.e., we always used the same hyper-parameters.

TR: For the standard Trust-Region Newton-CG method, we used $\delta^0 = 1$ as the initial trust-region radius and the standard parameters from the literature for the trail point acceptance and trust-region radius update, see Conn et al. (2000). The maximum number of conjugate-gradient iterations was set to 30.

ASTR: For the parameters that the ASTR and TR methods have in common, we used the same values. For the additional parameters, we chose $\theta = 0.5$ and $s^0 = 0.01 \cdot n$. We increased the sample size as described in Section 3.3.1 with $\omega = 2$. The parameter ε was set close to machine precision. For the curvature information in the ASTR method, we chose $A^{\nu,k} := D^2 F_{S_H^{\nu,k}}(x^{\nu,k})$, where $S_H^{\nu,k} \subseteq S^{\nu,k}$ with $s_H^\nu := |S_H^{\nu,k}| = 0.1 \cdot s^\nu$ and adjusted the number of inner iterations as it was described in section 3.3.3 (with $\bar{\alpha} = 5$ and $\bar{\beta} = 20$). When s^ν reaches its maximal size of n, s_H^ν is doubled in every outer iteration until it reaches n as well.

SGD: Two hyper-parameters where tuned for each problem and data set. The best combination of a step-size $t \in \{10^{-6}, 10^{-5} \ldots, 1, 10\}$ and mini-batch size $s = \lceil \zeta \cdot n \rceil$ for $\zeta \in \{\frac{1}{n}, 10^{-5}, 10^{-4} \ldots, 10^{-1}\}$ was selected.

SVRG: For each problem and data set we tried all combinations of step-sizes $t \in \{10^{-6}, 10^{-5} \ldots, 1, 10\}$ and number of inner iterations $K = \lceil \mu \cdot n \rceil$ for $\mu \in \{10^{-4}, 10^{-3}, \ldots, 1, 2\}$, and selected the combination that achieved the minimal training error.

For each problem and data set, a starting point x^0 was randomly generated and used by each of the methods. We ran each method for a fixed time budget. This was also the time budget that SGD and SVRG were tuned to. In order to report the training error $F(x^k) - F^\star$, the value $F^\star$ was determined by running the full-batch TR-Algorithm until it was unable to improve the objective value due to numeric precision.

4.1 Logistic Regression

Given a data set $(z^i, y_i) \in \mathbb{R}^m \times \{-1, 1\}$, $i = 1, \ldots, n$, we consider the ℓ_2-regularized logistic regression problem

$$\min_x \; F(x) = \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i(x^\intercal z_i)}\right) + \lambda \|x\|_2^2 \quad \text{with } \lambda = \frac{1}{n}.$$

Since the objective F is strongly convex, there exists a globally minimal point and it coincides with the unique critical point of F. We test the methods on the binary classification data sets described in Table A.1 in the appendix.

In Figure 4.1 we report the results concerning the minimization of the training error. We observe consistent superior performance of the ASTR method compared to the full-batch TR and the tuned SGD and SVRG methods.

We note that ASTR and the tuned SGD method consistently need less CPU time than the other methods in order to achieve a high test accuracy. ASTR is faster than SGD for the data sets *odd_even*, *skin*, *covertype* and *HIGGS*, and equally fast for all the remaining data sets. The test accuracies for *odd_even* and *HIGGS* are shown in the upper left panels of Figures 4.2 and 4.3, respectively.

We also compared the performance of the algorithms with respect to *effective gradient evaluations*, a platform and implementation independent measure often used in the literature, see, for example, Bollapragada et al. (2018c). In order to determine the number of effective gradient evaluations per iteration for each method, we made use of the fact that for each of the problems considered in this section, evaluating the objective is half as expensive as computing its gradient. And the latter operation costs the same as computing a Hessian-vector product. As to be expected, the qualitative results of the comparison of the algorithms remains unchanged when this alternative measure is used, see, for example, the upper right panels of Figures 4.2 and 4.3, where the training errors for the data sets *odd_even* and *HIGGS* are depicted. However, since the methods we compare are very dissimilar, we believe that CPU times are more transparent and therefore more appropriate in order to evaluate the performance of the methods. Note that the last row of Figures 4.2 and 4.3 shows the behavior of the sample sizes used in the gradient and Hessian matrix approximations (s and s_H, respectively), and of the number of inner iterations of the ADST method.

In Figures A.1 - A.8 in the appendix we report all the details concerning the results of our numerical experiments on logistic regression problems for each data set.

4.2 Nonlinear Least-Squares

We now focus on binary classification with squared loss as a concrete instance of a nonlinear (and nonconvex) least-squares problem. Given a data set $(z^i, y_i) \in \mathbb{R}^m \times \{0,1\}$, $i = 1, \dots, n$, we consider the problem

$$\min_x \; F(x) = \frac{1}{n} \sum_{i=1}^{n} \Big(y_i - \varphi(x^\intercal z^i) \Big)^2,$$

where φ denotes the sigmoid function, i.e., $\varphi(t) = \frac{1}{1+e^{-t}}$. We use the same data sets that were used for logistic regression in the previous section.

In Figure 4.4 we report the results of the numerical experiments. Again, our results show strong performance of the ASTR method. On all data sets, except for *w8a* and *skin*, ASTR is clearly superior to the other methods concerning the minimization of the training error. For the data set *skin*, it seems like ASTR, SGD and SVRG approximate a local minimal point, whereas TR approximates either a better local or the actual global minimal point.

In Figures A.9 - A.16 in the appendix we report additional details on our numerical experiments on nonlinear least-squares problems. Again, we observe that the test accuracies of ASTR and the tuned SGD method are comparable and that they are superior to the SVRG and TR methods. Only on the data set *skin*, the TR method approximates a minimal point with much better generalization properties than the local minimal point approximated by the other methods.

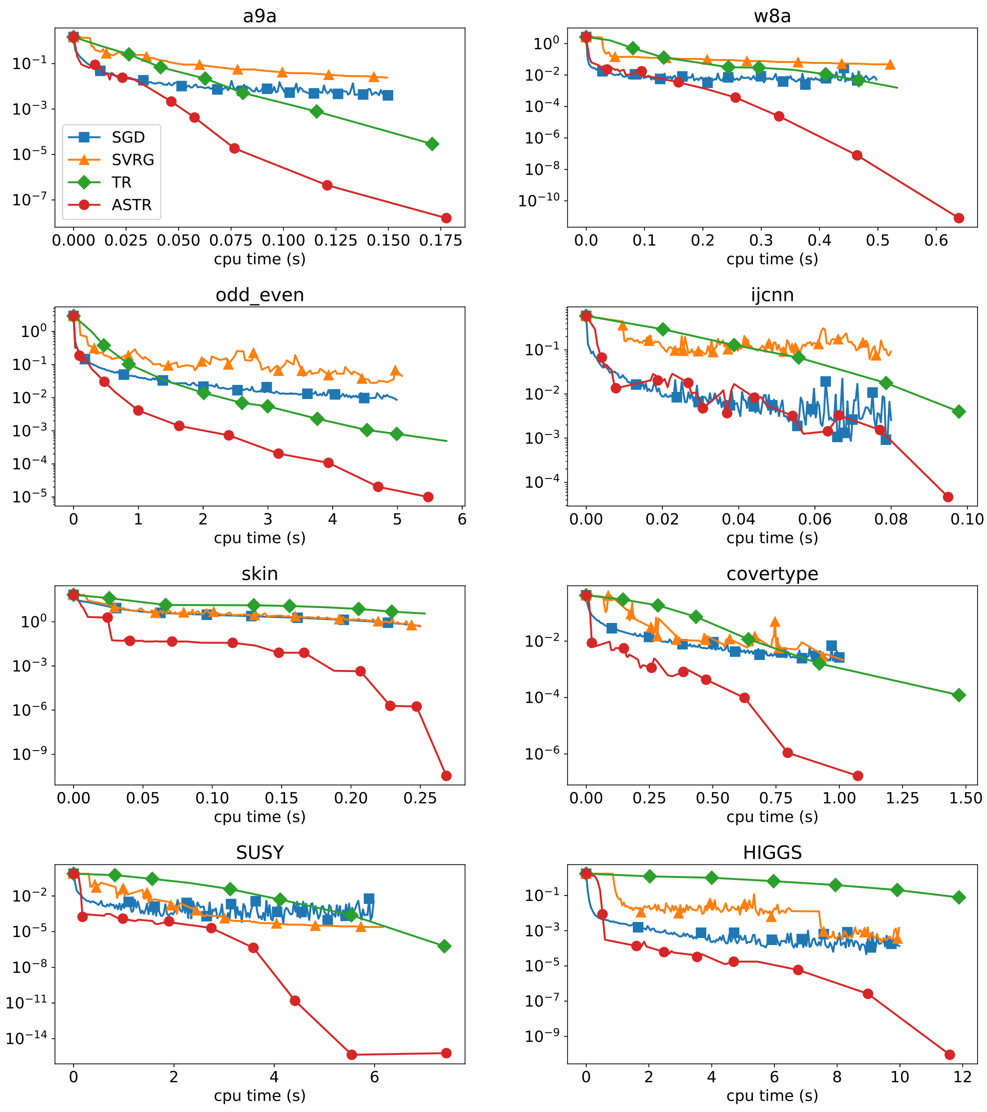


Figure 4.1: Logistic regression error of SGD, SVRG, TR and ASTR for all data sets.

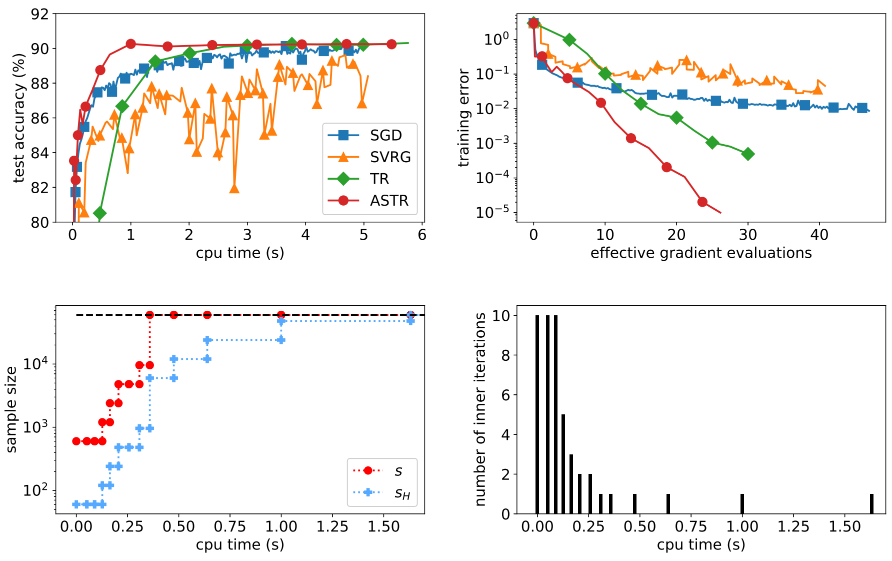


Figure 4.2: Further details on the performance of SGD, SVRG, TR and ASTR on the logistic regression problem with the *odd_even* data set. In the plots in the second row, the behaviour of the sample sizes and the number of inner iterations of ASTR is depicted.

4.3 Neural Network Training

Finally, we also considered the problem of training a simple two layer feed-forward neural network on the popular *MNIST* data set of handwritten digits, see Lecun et al. (1998). Details concerning this data set can be found in Table A.1 in the appendix. The fully connected two layer neural network has 748 input neurons, 100 hidden neurons and 10 output neurons. The hidden neurons implement the logistic function, the output neurons the softmax function. Thus, if we have a data set $(z^i, y_i) \in \mathbb{R}^{784} \times \{0,1\}^{10}$, $i = 1, \dots, n$, and choose the cross-entropy loss function, we arrive at the optimization problem

$$\min_{x\in\mathbb{R}^m} \; F(x) := -\sum_{i=1}^{n}\sum_{j=1}^{10} y_j^i \ln([h(z^i, x)]_j),$$

where $h(\cdot, x)$ denotes the function that implements the neural network with weight vector x.

In Figure 4.5 one can observe that ASTR achieves better results than the other methods concerning the training error. With regard to the test accuracy, ASTR performes on par with the tuned SGD method.

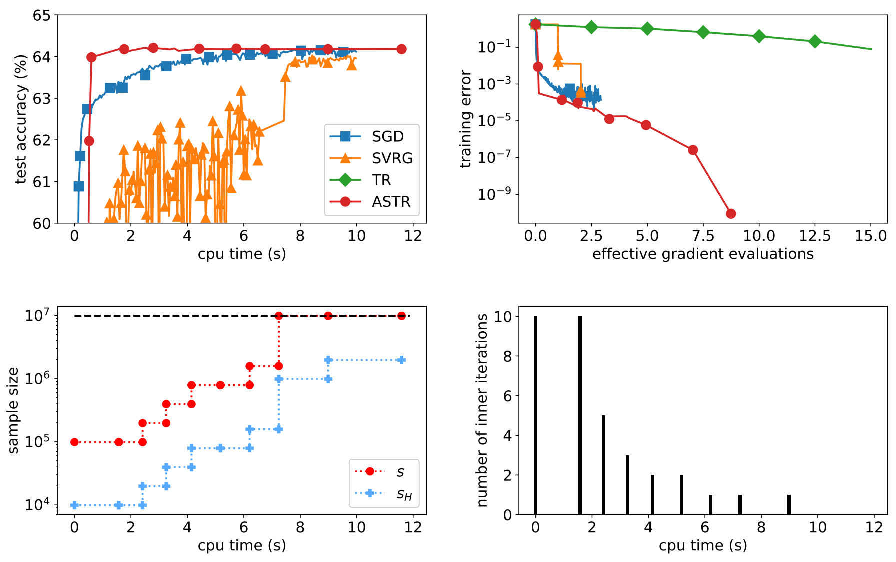


Figure 4.3: Further details on the performance of SGD, SVRG, TR and ASTR on the logistic regression problem with the *HIGGS* data set. In the plots in the second row, the behaviour of the sample sizes and the number of inner iterations of ASTR is depicted.

4.4 Conclusions

The results of the numerical experiments demonstrate that the ASTR method has significant advantages compared to the SGD and SVRG methods. On all the logistic regression problems, the majority of the nonlinear least-squares problems and the neural network training problem, the ASTR method outperformed the other two methods with respect to the minimization of the training error. Concerning the test accuracies, the ASTR method was mostly on par with the tuned SGD method. Moreover, whereas a lot of experimentation was necessary in order for the SGD and SVRG methods to show good performance, the performance of the ASTR method was not dependent on tuning the parameters to individual problems or data set.

The neural network training problem from the previous section is quite simple compared to those typically encountered in practice. So far it is not clear whether the ASTR method can be beneficial for the training of these more complex neural networks. We leave this important question for future research.

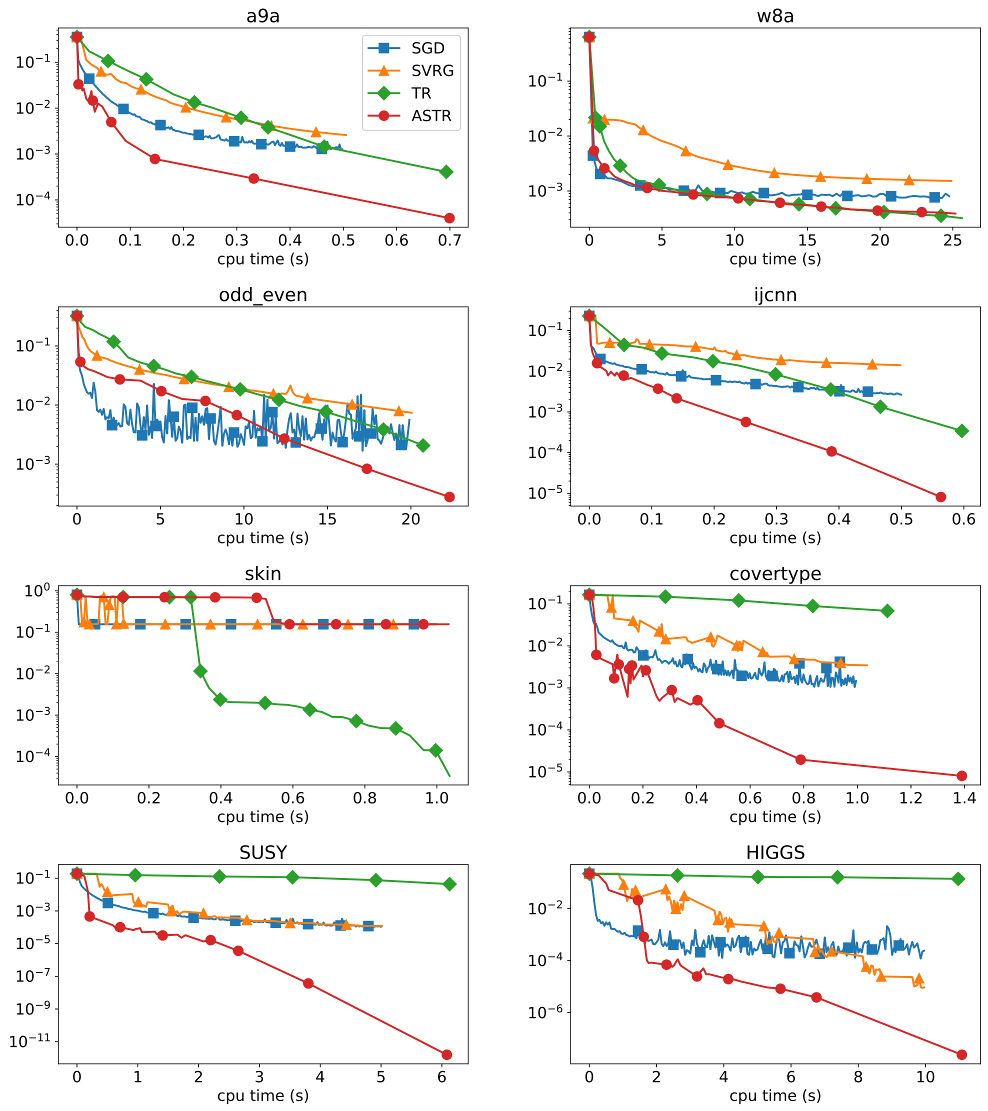


Figure 4.4: Nonlinear least-squares error of SGD, SVRG, TR and ASTR for all data sets.

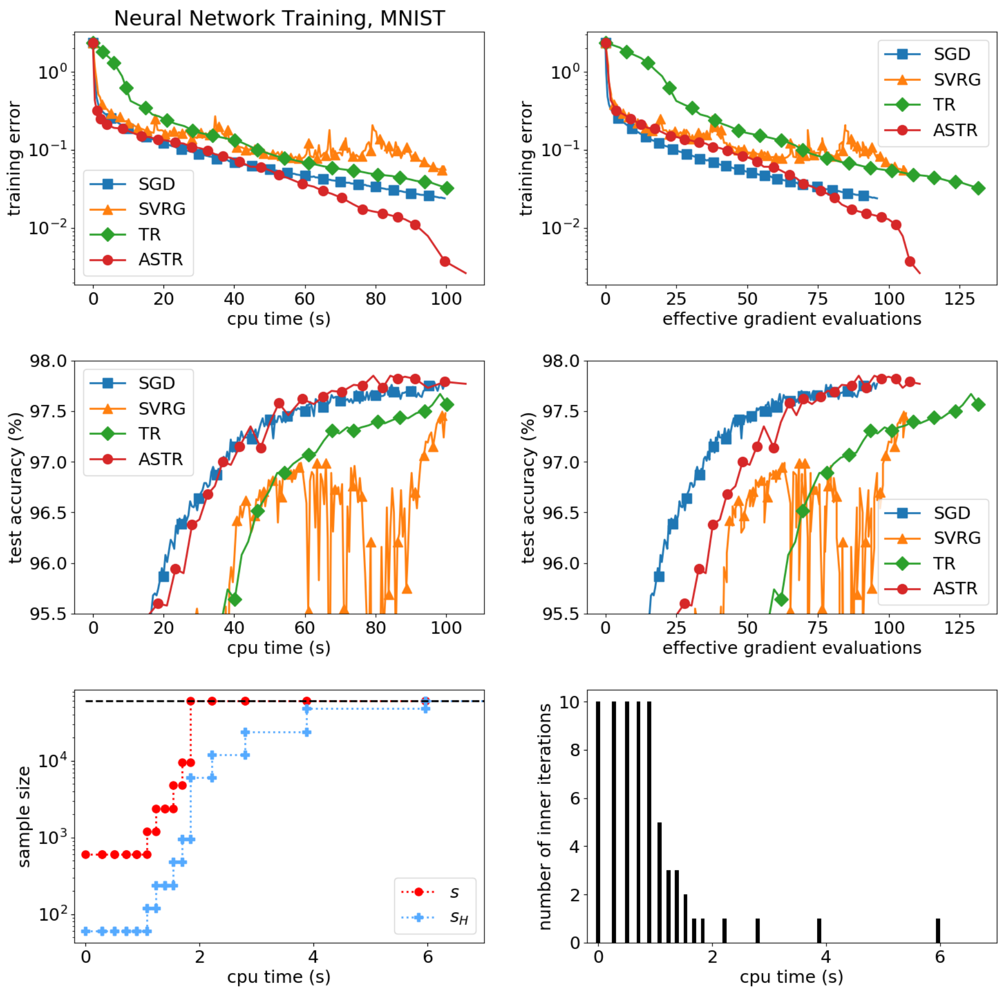


Figure 4.5: Performance of SGD, SVRG, TR and ASTR on the neural network training problem with the *MNIST* data set.

Part II

Computing Global Solutions of the Least-Squares Spline Approximation Problem with Free Knots

Chapter 5

Introduction

In Part I of this thesis we considered the general class of possibly nonconvex and large-scale finite-sum minimization problems and proposed a local optimization algorithm that is particularly well suited for the solution of problems from this class. However, there also exist nonconvex data approximation problems for which purely local optimization algorithms do not yield satisfactory results. In Part II of this thesis we consider a specific nonconvex data approximation problem that is known to be difficult to solve with local optimization methods, namely the least-squares spline approximation problem with free knots.

In this chapter we first explain what a spline function is and why it is a useful tool for data approximation. Then we formulate the least-squares spline approximation problem with fixed and free knots. After a review of existing approaches for the solution of the approximation problem with free knots, we give a short overview of our approach.

5.1 Curve Fitting with Splines

In the basic data approximation problem we are given n data points $(x_i, y_i) \in \mathbb{R}^2$ and a family of functions $h(\cdot, w) : \mathbb{R} \to \mathbb{R}$ parametrized with a vector $w \in \mathbb{R}^m$. The goal is to find a parameter $\bar{w}$ such that $h(x_i, \bar{w}) \approx y_i$ for all $i \in \{1, \ldots, n\}$. Thus, one aims to compute the global minimizer of the finite-sum problem

$$\min_{w \in \mathbb{R}^m} \sum_{i=1}^{n} \ell(h(x_i, w), y_i),$$

where $l : \mathbb{R}^2 \to \mathbb{R}$ is an error function that quantifies the discrepancy between $h(x_i, w)$ and the actual value y_i corresponding to x_i.

In many applications it is clear from the context which family of functions $h(\cdot, w)$ should be used and the parameters w might even have a physical meaning. However, there are also many cases where no specific family of functions $h(\cdot, w)$ is dictated by the application. In those cases spline functions are a popular tool for data approximation, see Dierckx (1996).

A spline function consists of several polynomial functions of degree $m \in \mathbb{N}$ each defined on a segment of the approximation interval $[x_1, x_m]$. The joint points of those segments are called knots. In order to simplify the exposition, we only consider cubic spline functions in this thesis, i.e, $m = 3$. The formal definition of a cubic spline function with k knots is

$$s(x, \beta^{(0)}, \ldots, \beta^{(k)}, \xi) := \begin{cases} p(x, \beta^{(0)}), & \text{for } x_1 \leq x \leq \xi_1 \\ p(x, \beta^{(1)}), & \text{for } \xi_1 < x \leq \xi_2, \\ \quad \vdots & \\ p(x, \beta^{(k)}), & \text{for } \xi_k < x \leq x_n, \end{cases}$$

with the ordered knot vector $\xi \in \mathbb{R}^k$, $x_1 < \xi_1 < \xi_2 < \ldots < \xi_k < x_n$, and $k+1$ cubic polynomials

$$p(x, \beta^{(j)}) := \beta_0^{(j)} + \beta_1^{(j)} x + \beta_2^{(j)} x^2 + \beta_3^{(j)} x^3$$

for $j \in \{0, \ldots, k\}$. Depending on the application, the cubic polynomials may have to satisfy certain continuity restrictions at the knots where they join, e.g., continuity and smoothness up to the second derivative. Note that we limit our exposition to cubic splines with the maximum number of continuity restrictions only for ease of presentation. It is straightforward to generalize our approach to piecewise polynomials of arbitrary degree and different continuity restrictions.

Another important choice concerns the type of error function. In this thesis we will mainly consider the most important and well-known error function, namely the least-squares criterion, i.e., $l(z, y) = (z - y)^2$. Depending on the application, other choices can be more appropriate, e.g., $l(z, y) = |z - y|$. However, these alternative error functions and their implications for the solution of the spline approximation problem are for the most part beyond the scope of this thesis.

5.2 The Least-Squares Spline Approximation Problem

In this section we start by formulating the least-squares spline approximation problem with fixed knots and describe how to efficiently compute its optimal solutions. Then we consider the problem with free knots and explain why the computation of globally optimal solutions is challenging.

5.2.1 The Problem with Fixed Knots

We first assume that the knot vector $\xi \in \mathbb{R}^k$ is fixed and the goal is to determine parameters optimal with respect to the least-squares criterion, i.e., we aim to solve the optimization problem

$$\min_{\beta \in \mathbb{R}^{4(k+1)}} \sum_{i=1}^{n} (s(x_i, \beta, \xi) - y_i)^2, \tag{5.1}$$

where $\beta \in \mathbb{R}^{4(k+1)}$ denotes the concatenation of the vectors $\beta^{(0)}, \dots, \beta^{(k)} \in \mathbb{R}^4$. Since the spline function s is linear in the decision variable β, the problem is a linear least-squares problem whose optimal solutions can be computed by solving a system of linear equations.

In order to derive this system of linear equations, we first reformulate problem (5.1) as the linearly constrained convex quadratic problem

$$\begin{aligned} \min_{\beta} \quad & \sum_{j=0}^{k} \sum_{i \in I_j} \left(p(x_i, \beta^{(j)}) - y_i \right)^2 \\ \text{s.t.} \quad & p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & \beta \in \mathbb{R}^{4(k+1)}, \end{aligned} \tag{5.2}$$

with the index sets

$$\begin{aligned} I_0 &:= \{ i \mid x_1 \leq x_i \leq \xi_1 \} \\ I_j &:= \{ i \mid \xi_j < x_i \leq \xi_{j+1} \}, \ j = 1, \dots, k-1, \\ I_k &:= \{ i \mid \xi_k \leq x_i \leq x_n \}. \end{aligned}$$

Note that we assume the maximum number of continuity restrictions in problem (5.2).

Let $X_j \in \mathbb{R}^{n_j \times 4}$ be the Vandermonde matrix corresponding to the n_j data points with indices in I_j, i.e., a matrix with the rows $(1, x_i, x_i^2, x_i^3)$, for $i \in I_j$. Let $y^j \in \mathbb{R}^{n_j}$ be a column vector of the concatenated y_i values with $i \in I_j$.

We can rewrite problem (5.2) as

$$\begin{aligned} \min_{\beta} \quad & \sum_{j=0}^{k} \|X_j \beta^{(j)} - y^j\|_2^2 \\ \text{s.t.} \quad & p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k \\ & \beta \in \mathbb{R}^{4(k+1)}. \end{aligned}$$

With the row vectors

$$A_j := \begin{pmatrix} 1 & \xi_j & \xi_j^2 & \xi_j^3 \end{pmatrix}, \quad B_j := \begin{pmatrix} 0 & 1 & 2\xi_j & 3\xi_j^2 \end{pmatrix}, \quad C_j := \begin{pmatrix} 0 & 0 & 2 & 6\xi_j \end{pmatrix},$$

for $j = 1, \dots, k$, we can define the matrix

$$D := \begin{pmatrix} A_1 & -A_1 & 0 & 0 & \cdots & 0 \\ B_1 & -B_1 & 0 & 0 & \cdots & 0 \\ C_1 & -C_1 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & -A_2 & 0 & \cdots & 0 \\ 0 & B_2 & -B_2 & 0 & \cdots & 0 \\ 0 & C_2 & -C_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & 0 & A_{k-1} & -A_{k-1} & 0 \\ 0 & \cdots & 0 & B_{k-1} & -B_{k-1} & 0 \\ 0 & \cdots & 0 & C_{k-1} & -C_{k-1} & 0 \\ 0 & \cdots & 0 & 0 & A_k & -A_k \\ 0 & \cdots & 0 & 0 & B_k & -B_k \\ 0 & \cdots & 0 & 0 & C_k & -C_k \end{pmatrix} \in \mathbb{R}^{3k \times 4(k+1)}$$

and rewrite our problem as

$$\begin{aligned} \min_{\beta} \quad & q(\beta) := \sum_{j=0}^{k} \|X_j \beta^{(j)} - y^j\|_2^2 \\ \text{s.t.} \quad & D\beta = 0, \\ & \beta \in \mathbb{R}^{4(k+1)}. \end{aligned}$$

We assume that D has full row rank, i.e., $\text{rank} D = 3k$. Since the objective function q is convex, a point $\beta^\star$ is optimal if and only if there exists a $\mu \in \mathbb{R}^{3k}$ such that the linear system of equations

$$\begin{aligned} \nabla q(\beta^\star) + D^\intercal \mu &= 0 \\ D\beta^\star &= 0. \end{aligned}$$

is satisfied. Noting that

$$\nabla q(\beta) = 2 \cdot \begin{pmatrix} X_0^\intercal X_0 \beta^{(0)} - X_0^\intercal y^0 \\ X_1^\intercal X_1 \beta^{(1)} - X_1^\intercal y^1 \\ \vdots \\ X_k^\intercal X_k \beta^{(k)} - X_k^\intercal y^k \end{pmatrix} \in \mathbb{R}^{4(k+1)}$$

we can define the matrix

$$X := 2 \cdot \begin{pmatrix} X_0^\intercal X_0 & 0 & \cdots & 0 \\ 0 & X_1^\intercal X_1 & \cdots & 0 \\ 0 & 0 & \ddots & X_k^\intercal X_k \end{pmatrix} \in \mathbb{R}^{4(k+1)\times 4(k+1)}$$

and the vector

$$b := 2 \cdot \begin{pmatrix} X_0^\intercal y^0 \\ \vdots \\ X_k^\intercal y^k \end{pmatrix} \in \mathbb{R}^{4(k+1)}$$

and reformulate this linear system as

$$\begin{pmatrix} X & D^\intercal \\ D & 0 \end{pmatrix} \begin{pmatrix} \beta \\ \mu \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{5.3}$$

This linear system has a unique solution if and only if $X|_{\mathrm{Kern}(D)}$ is non-singular. Note that X is positive definite if and only if all the Vandermonde matrices X_j, $j = 1, \ldots, k$, have full column rank, i.e. $\mathrm{rank}(X_j) = 4$. If $n_j \geq 4$ the matrix X_j has full column rank if and only if at least four of the data points x_i with $i \in I_j$ are distinct. Thus, the linear system (5.3) has a unique solution under very mild conditions.

If fewer continuity restrictions are needed, e.g, because the splines do not have to be smooth but only continuous at the knots, the matrix D changes accordingly. In case there are no continuity restrictions, the linear system (5.3) is equivalent to the $k+1$ linear systems

$$X_j^\intercal X_j \beta^{(j)} = X_j^\intercal y^j, \; j = 0, \ldots, k,$$

which are, of course, the normal equations for each cubic polynomial.

Summarizing the above, when the vector of knots ξ is fixed, one can solve a simple linear system in order to obtain an optimal solution of problem (5.2). However, the quality of the fit of the cubic spline function is highly dependent on the placement of the knots $\xi_1, \dots, \xi_k$. Thus, a natural question is whether it is possible to determine an optimal placement of the knots.

5.2.2 The Problem with Free Knots

If the knot vector $\xi \in \mathbb{R}^k$ is not fixed but enters the optimization problem as a decision variable, the problem is referred to as the least squares spline approximation problem with free knots:

$$\begin{aligned} \min_{\beta,\ \xi} \quad & \sum_{j=0}^{k} \sum_{i \in I_j} \left(p(x_i, \beta^{(j)}) - y_i \right)^2 \\ \text{s.t.} \quad & p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\ & \xi_j \le \xi_{j+1}, \quad j = 1, \dots, k-1, \\ & x_1 \le \xi_1,\ \xi_k \le x_n, \\ & \beta \in \mathbb{R}^{4(k+1)},\ \xi \in \mathbb{R}^k. \end{aligned} \tag{5.4}$$

Note that while the knot vector ξ is a decision variable, the number of knots k is still a fixed parameter of the problem.

Since the equality constraints are nonlinear in the decision variables ξ_j, problem (5.4) is not a convex optimization problem. Moreover, problem (5.4) typically possesses many locally minimal solutions that prevent local optimization algorithm from converging to the globally minimal solutions.

In order to visualize this, we reformulate the problem such that the vector ξ is the only decision variable. This is easily possible since we can obtain the optimal parameters $\beta(\xi)$ of a spline function with a fixed knot vector ξ by solving a linear system of equations, which is uniquely solvable under mild conditions. With this notation we can write problem (5.4) equivalently as the problem

$$\begin{aligned} \min_{\xi} \quad & \sum_{j=0}^{k} \sum_{i \in I_j} \left(p(x_i, \beta^{(j)}(\xi)) - y_i \right)^2 \\ \text{s.t.} \quad & \xi_j \le \xi_{j+1}, \quad j = 1, \dots, k-1, \\ & x_1 \le \xi_1,\ \xi_k \le x_n \\ & \xi \in \mathbb{R}^k. \end{aligned} \tag{5.5}$$

Note that the nonlinear equality constraints are eliminated from the problem since they are automatically satisfied due to the choice of the optimal parameter vector $\beta(\xi)$. However, the problem is still a nonconvex problem since $\beta(\cdot)$ is a nonlinear function of the knot vector ξ.

Suboptimal local solutions of problem (5.5) are already present when the dimension k of the knot vector is small. We visualize this for $k = 1$ and $k = 2$. Figure 5.1a shows the objective function of problem (5.5) for a synthetic data set with 200 data points and one free knot ξ_1. We observe that there exists a global and a local minimum. Moreover, a local optimization algorithm initialized with a random knot placement might approximate either one of these two locally optimal points with similar probability. The spline functions corresponding to these solutions are shown in Figure 5.1b. Based on these plots one could argue that the spline function corresponding to the globally optimal solutions approximates the data set more accurately, but it is also apparent that a cubic spline function with one free knot is not enough in order to obtain a good approximation of this data set.

The objective function of problem (5.5) for the same data set and two free knots ξ_1 and ξ_2 is depicted in Figure 5.2. The function has one globally minimal point $\bar{\xi} := (0.2632, 0.5166)^\intercal$ and two locally minimal points $\tilde{\xi} := (0.0939, 0.1003)^\intercal$ and $\widehat{\xi} := (0.8191, 0.8422)^\intercal$. Moreover, the spline functions corresponding to the local solutions approximate the data set significantly worse than the spline function corresponding to the globally optimal knot placement, as can be observed in Figure 5.3.

This demonstrates that local optimization algorithm are only useful for the solution of the least-squares spline approximation problem with free knots if they start from an initial solution close enough to the globally optimal solution. In this thesis, we propose a novel way to compute a solution close to the global optimal solution. Before we describe our approach, we review some of the existing approaches for the solution of the least-squares spline approximation problem with free knots.

5.3 Literature review of existing approaches

The earliest algorithmic approach to determine an optimal knot placement with respect to the least-square error for a fixed given number of knots is a discrete Newton method proposed by de Boor and Rice (1968), which can be used to approximate locally optimal solutions. Other local approaches include the use of Gauss-Newton-type methods, see, e.g., Gallant and Fuller (1973), Jupp (1978) and Schwetlick and Schütze (1995), and the Fletcher-Reeves nonlinear conjugate gradient (CG) method, see Dierckx (1996). With these approaches it is only possible to determine locally optimal solutions, and their quality is highly dependent on the initial solution provided to the algorithm.

In contrast, genetic algorithms, see, e.g., Pittman and Murthy (2000), Miyata and Shen (2003) and Spiriti et al. (2013), can escape low-quality locally optimal solutions, but the solution that is returned by these algorithms might be neither locally nor globally optimal, since these approaches are purely heuristic and cannot provide any certificate of optimality.

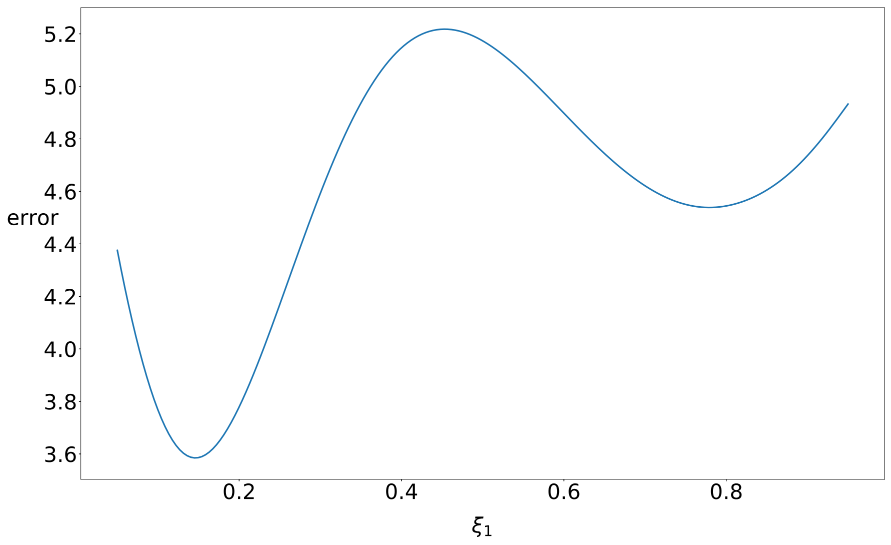
In Beliakov (2004) the cutting angle method is used in order to compue the globally optimal knot placement of the least-squares spline approximation problem with free knots. To the best of our knowledge, it is the only method from deterministic global optimization that was proposed specifically for the solution of this problem. However, as noted in the section on numerical experiments in Beliakov (2004), the computing time is prohibitive even for small problems.

There exist many approaches in the literature which, in contrast to the methods reviewed so far, do not seek an optimal knot placement with respect to the least-squares criterion, but instead determine a knot placement that satisfies other quality measuring criteria. Some suggest to look at the data and place the knots according to some rule of thumbs, such as placing them near points of inflection or at specific quantiles, see Wold (1974) and Ruppert (2002). Other approaches are based on forward knot addition and/or backward knot deletion, see, e.g, Smith (1982) and Stone et al. (1997). Moreover, Bayesian approaches are proposed in Denison et al. (1998) and DiMatteo et al. (2001), which employ a continuous random search methodology via reversible-jump Markov chain Monte Carlo methods. A more detailed review and comparison of these approaches can be found in Wand (2000) and Lee (2002).

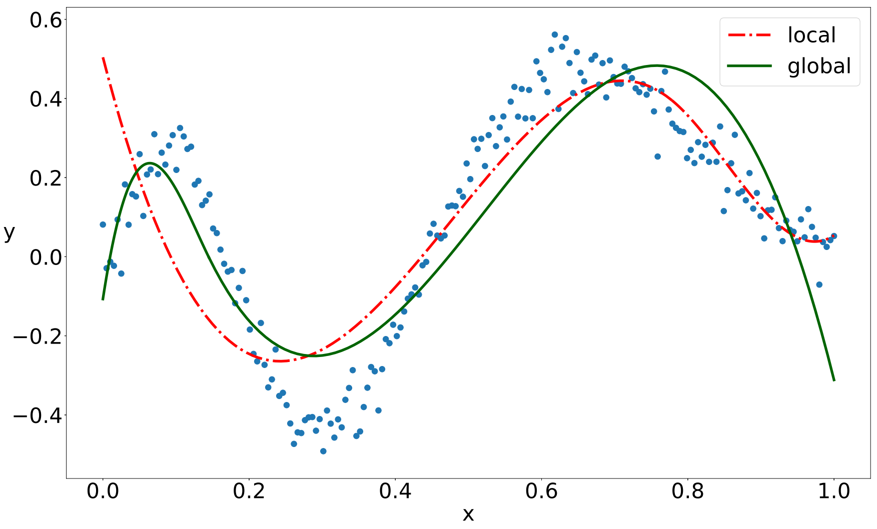
5.4 Overview of our Approach

In this article, we propose to solve a combinatorial formulation of the least-squares spline approximation problem with free knots in order to obtain a solution close to the globally optimal solution. We assume the number k of free knots to be fixed and are only concerned with the optimal placement of the knots. The first approach that we present is the formulation of the combinatorial least-squares spline approximation problem as a mixed-integer quadratically constrained problem (MIQCP). MIQCPs can be solved with commercial optimization solvers like Gurobi or CPLEX.

The second method that we propose is a branch-and-bound algorithm tailored specifically to the combinatorial least-squares spline approximation problem with free knots. In contrast to heuristic approaches, branch-and-bound methods maintain provable upper and lower bounds on the optimal value and terminate with a globally optimal solution after a finite number of steps. In the worst case, the computational effort grows exponentially in the problem size. Nevertheless, branch-and-bound methods have been successfully applied to a range of statistical problems like variable selection and clustering, see, for example, Hand (1981) and Brusco and Stahl (2005). And as we will demonstrate in the numerical experiments in Chapter 8, it is possible to solve the combinatorial formulation for problem sizes typically encountered in practice.

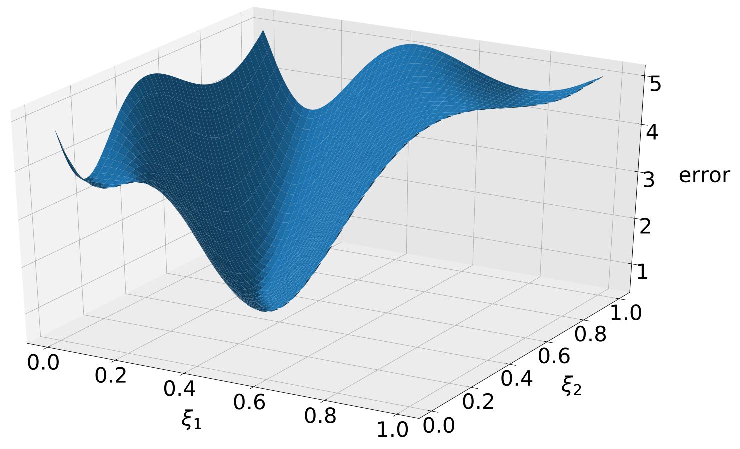


(a) Objective function of problem (5.5) for one free knot

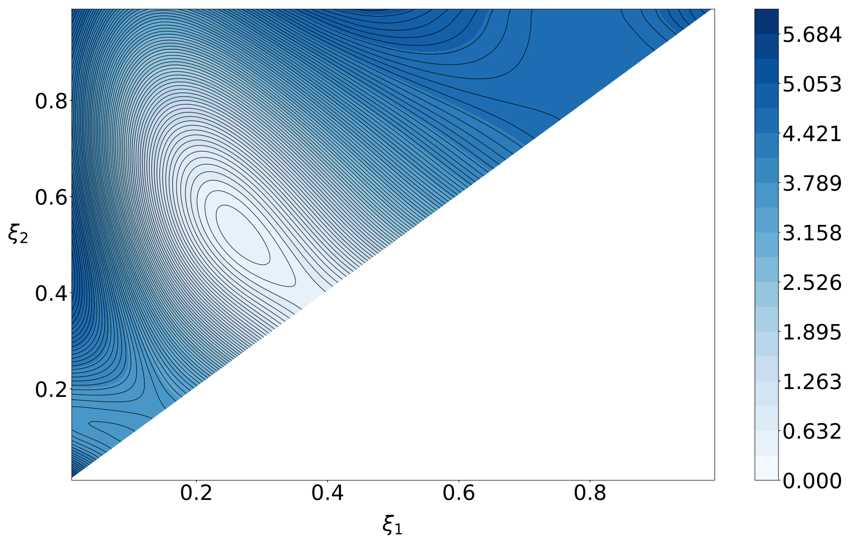


(b) Locally and globally optimal spline functions with one knot

Figure 5.1: Objective function of problem (5.5) with one free knot and the corresponding locally and globally optimal least-squares spline functions.



(a) Objective function for two free knots



(b) Contour plot of objective function for two free knots

Figure 5.2: Objective function of problem (5.5) with two free knots.

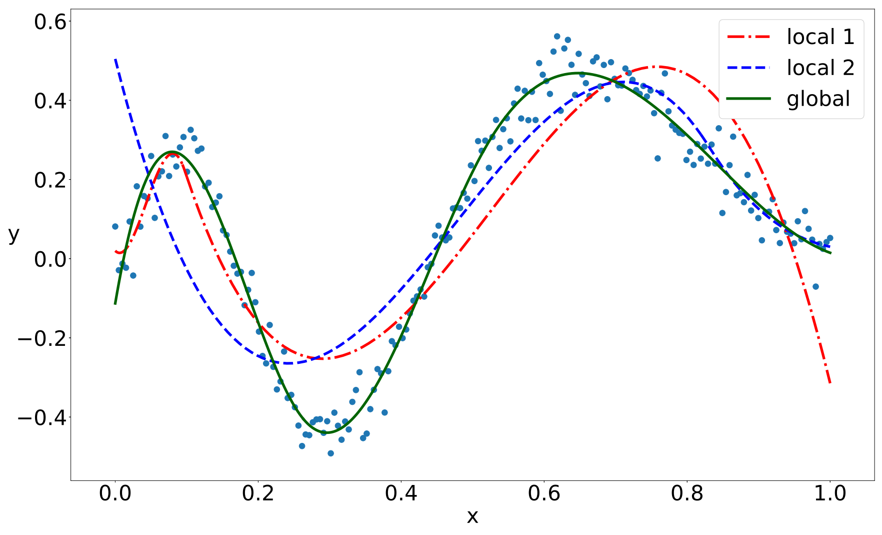


Figure 5.3: Least-squares splines corresponding to the two locally minimal knot placements and the least-squares spline corresponding to the globally optimal knot placement $\bar{\xi}$. The red and blue curves correspond to the knot placements $\tilde{\xi}$ and $\widehat{\xi}$, respectively.

Chapter 6

A Combinatorial Formulation

In this chapter we describe how the least-squares spline approximation problem with free knots can be reformulated as a combinatorial optimization problem. Then we explain how this combinatorial formulation can be modeled as a mixed-integer optimization problem. In the last section we comment on other error criterions and their implications for mixed-integer formulations.

6.1 Reformulation as a Set Partitioning Problem

We reformulate the least squares spline approximation problem with free knots as a type of set partitioning problem. It is not an equivalent reformulation, however, since we restrict the possible knot locations to a finite set. Nevertheless, as we will explain in the following, we restrict the possible knot locations in a very natural and intuitive way, and such that the optimal solution of the combinatorial formulation is a promising initial point for local optimization algorithms.

If our goal is to approximate the given dataset with a spline with k knots, then every data point needs to be assigned to exactly one of the $k+1$ polynomials. Therefore, instead of computing the optimal placement of the knots, we aim to find an optimal partition $\{I_j \mid j = 0, \ldots, k\}$ of the set of indices $\{1, \ldots, n\}$.

Due to the assumed ordering of the design points $x_1 < \ldots < x_n$, a partition is only feasible if for all $i, j \in \{0, \ldots, k\}$ with $i < j$ we have that $r < q$ for all $r \in I_i$ and $q \in I_j$. Since the spline function corresponding to a feasible partition $\{I_j \mid j = 0, \ldots, k\}$ has to satisfy continuity conditions at the knots, the exact location of these knots has to be fixed based solely on the partition. Clearly, the knot ξ_j has to be placed between the values

$\max_{i\in I_{j-1}} x_i$ and $\min_{i\in I_j} x_i$ for all $j = 1,\dots,k$. We propose to select the midpoint of these two values, i.e., we set $\xi_j = (\max_{i\in I_{j-1}} x_i + \min_{i\in I_j} x_i)/2$ for all $j = 1,\dots,k$. This simple and symmetric choice worked well in our numerical experiments.

Thus, in order to determine the optimal feasible partition, we solve the combinatorial optimization problem

$$\begin{aligned}
\min_{\beta, I_0,\dots,I_k} \quad & \sum_{j=0}^{k} \sum_{i\in I_j} \left(p(x_i,\beta^{(j)}) - y_i\right)^2 \\
\text{s.t.} \quad & \bigcup_{j=0}^{k} I_j = \{1,\dots,n\}, \\
& I_j \neq \emptyset, \quad j = 0,\dots,k, \\
& s < i \text{ for all } s \in I_l \text{ and } i \in I_j \text{ with } l < j, \\
& p(\xi_j,\beta^{(j-1)}) = p(\xi_j,\beta^{(j)}), \quad j = 1,\dots,k, \\
& p'(\xi_j,\beta^{(j-1)}) = p'(\xi_j,\beta^{(j)}), \quad j = 1,\dots,k, \\
& p''(\xi_j,\beta^{(j-1)}) = p''(\xi_j,\beta^{(j)}), \quad j = 1,\dots,k, \\
& \beta \in \mathbb{R}^{4(k+1)},
\end{aligned} \tag{6.1}$$

where

$$\xi_j = (\max_{i\in I_{j-1}} x_i + \min_{i\in I_j} x_i)/2,$$

for all $j \in \{1,\dots,k\}$.

The total number of feasible set partitions in problem (6.1) is $\binom{n-1}{k}$. Note that we do not allow empty index sets in our feasible partitions since this would reduce the number of knots of the spline function.

We also remark that if the goal is to compute a piecewise polynomial, i.e., a spline function without any continuity restrictions, only the first three constraints in problem (6.1) are needed and the resulting combinatorial problem is an equivalent reformulation of the least-squares piecewise polynomial approximation problem with free knots.

6.2 Formulation as a Convex Mixed-Integer Quadratically Constrained Problem

A natural step towards the solution of the combinatorial problem (6.1) is to reformulate it as a type of mixed-integer optimization problem that can be solved with a commercial optimization solver like Gurobi or CPLEX. As will be explained in detail in the rest of this section, problem (6.1) can be equivalently reformulated as the following convex mixed-integer quadratically constrained problem (MIQCP):

$$
\begin{aligned}
\min_{\alpha,\beta,z,q,w} \quad & \sum_{i=1}^{n} \alpha_i \\
\text{s.t. } & [z_{ij} = 1] \Longrightarrow [q_{ij} = p(x_i, \beta^{(j)}) - y_i], \ \forall (i,j) \in J_1 \times J_2, \\
& q_{ij}^2 \leq \alpha_i, \ \forall (i,j) \in J_1 \times J_2, \\
& \sum_{j=0}^{k} z_{ij} = 1, \ \forall i \in J_1, \\
& \sum_{i=1}^{n} z_{ij} \geq 1, \ \forall j \in J_2, \\
& \sum_{q=i+1}^{n} \sum_{r=0}^{j-1} (1 - z_{qr}) \geq (n-i) \cdot j \cdot z_{ij}, \ \forall i \in J_1 \setminus \{n\}, j \in J_2 \setminus \{0\}, \\
& w_{ij} \leq z_{ij}, \ \forall (i,j) \in \bar{J}, \\
& w_{ij} \leq z_{i+1,j+1}, \ \forall (i,j) \in \bar{J}, \\
& w_{ij} \geq z_{ij} + z_{i+1,j+1} - 1, \ \forall (i,j) \in \bar{J}, \\
& [w_{ij} = 1] \Longrightarrow [p(\gamma_i, \beta^{(j)}) - p(\gamma_i, \beta^{(j+1)}) = 0], \ \forall (i,j) \in \bar{J}, \\
& [w_{ij} = 1] \Longrightarrow [p'(\gamma_i, \beta^{(j)}) - p'(\gamma_i, \beta^{(j+1)}) = 0], \ \forall (i,j) \in \bar{J}, \\
& [w_{ij} = 1] \Longrightarrow [p''(\gamma_i, \beta^{(j)}) - p''(\gamma_i, \beta^{(j+1)}) = 0], \ \forall (i,j) \in \bar{J}, \\
& \alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^{4(k+1)}, z \in \{0,1\}^{n(k+1)}, q \in \mathbb{R}^{n(k+1)}, w \in \{0,1\}^{(n-1)k}
\end{aligned}
\tag{6.2}
$$

where $J_1 := \{1, \ldots, n\}$, $J_2 := \{0, \ldots, k\}$, $\bar{J} := (J_1 \setminus \{n\}) \times (J_2 \setminus \{k\})$ and $\gamma_i := (x_i + x_{i+1})/2$, for all $i \in J_1 \setminus \{n\}$.

In total, the problem contains $(n+4) \cdot (k+1) + n$ continuous variables, $2nk + n - k$ binary variables and $9n(k+1)$ constraints. Among those constraints are $n(k+1)$ convex quadratic inequality constraints and $4n(k+1)$ indicator constraints.

Indicator constraints have the general structure

$$[c = \delta] \Longrightarrow a^\intercal x \leq b, \tag{6.3}$$

where $x \in \mathbb{R}^n$ and $c \in \{0,1\}$ are variables and $\delta \in \{0,1\}$, $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are fixed parameters. The interpretation is that the inequality is only enforced if the condition $c = \delta$ is true, otherwise the inequality is ignored by the solver. Indicator constraints are an alternative to big-M formulations and can be handled directly by commercial optimization solvers like Gurobi or Cplex. The big-M formulation

$$d \cdot M \leq a^\intercal x - b \leq d \cdot M,$$

is equivalent to (6.3) if $M \in \mathbb{R}$ is a sufficiently large number and d is a binary variable that satisfies

$$d = \begin{cases} 0, & \text{if } c = \delta, \\ 1, & \text{otherwise.} \end{cases}$$

In general, the existence of a sufficiently large value M is dependent on the objective function and the other constraints in the optimization problem. If M is too small, the optimal solution might be excluded from the problem. On the other hand, large big-M constants can lead to numerical difficulties and excessively long run times. In contrast, indicator constraints are handled internally by the solvers and usually do not lead to numerical difficulties. However, they can lead to longer run times compared to well chosen big-M constants in some cases. Further below we will comment on whether a big-M formulation could be an appropriate alternative to the indicator constraints in problem (6.2).

The constraints in problem (6.2) can be grouped into three blocks: The first two constraints stem from an epigraph reformulation of the least-squares objective function, the next three constraints ensure that the partition of the data points encoded in the variables z_{ij} is feasible, and the remaining constraints enforce the continuity restrictions at the knots. In the following subsections we explain each of the three blocks in detail.

6.2.1 Feasible Partitions

For every index tuple $(i,j) \in J_1 \times J_2$ the binary variable z_{ij} encodes whether the ith design point x_i is assigned to the jth polynomial, or, equivalently, if the index i is contained in the index set I_j, i.e., for all $(i,j) \in J_1 \times J_2$ it holds that

$$z_{ij} = \begin{cases} 1, & \text{if } i \in I_j, \\ 0, & \text{if } i \notin I_j. \end{cases} \tag{6.4}$$

The constraints

$$\sum_{j=0}^{k} z_{ij} = 1, \ \forall i \in J_1, \tag{6.5}$$

$$\sum_{i=1}^{n} z_{ij} \geq 1, \ \forall j \in J_2, \tag{6.6}$$

$$\sum_{q=i+1}^{n} \sum_{r=0}^{j-1} (1 - z_{qr}) \geq (n-i) \cdot j \cdot z_{ij}, \ \forall i \in J_1 \setminus \{n\}, j \in J_2 \setminus \{0\}, \tag{6.7}$$

can be explained intuitively by observing that the constraints (6.5) and (6.6) ensure that each design point is assigned to exactly one polynomial, and that at least one design point is assigned to each polynomial. Moreover, constraint (6.7) states that a design point x_i can only be assigned to the jth polynomial if all the following design points are not assigned to prior polynomials.

Constraints (6.5), (6.6) and (6.7) are equivalent to the constraints

$$\bigcup_{j=0}^{k} I_j = \{1, \ldots, n\}, \tag{6.8}$$

$$I_j \neq \emptyset, \quad j = 0, \ldots, k, \tag{6.9}$$

$$s < i \text{ for all } s \in I_l \text{ and } i \in I_j \text{ with } l < j, \tag{6.10}$$

in problem (6.1). Clearly, conditions (6.6) and (6.9) are equivalent. The equivalence of the remaining conditions is shown in the following lemma.

Lemma 6.2.1 *If the binary vector $z \in \{0,1\}^{n(k+1)}$ is defined as in* (6.4)*, then conditions* (6.5) *and* (6.7) *are equivalent to conditions* (6.8) *and* (6.10)*.*

Proof. We first show that if conditions (6.8) and (6.10) are satisfied, then conditions (6.5) and (6.7) are satisfied as well. Note that conditions (6.8) and (6.10) are both satisfied. Condition (6.8) implies that $\sum_{j=0}^{k} z_{ij} \geq 1$ holds for all $i \in J_1$ and since condition (6.10) implies that every index is assigned to at most one index set, i.e., $\sum_{j=0}^{k} z_{ij} \leq 1$ for all $i \in J_1$, we arrive at condition (6.5).

Next, we choose some $i \in J_1 \setminus \{n\}$ and $j \in J_2 \setminus \{0\}$. If $z_{ij} = 0$ then the inequality in condition (6.7) is satisfied trivially. On the other hand, if $z_{ij} = 1$, then $i \in I_j$ holds. Let us define the set $K := \{i+1, \ldots, n\} \times \{0, \ldots, j-1\}$ and suppose there exist $(q,r) \in K$ with $z_{qr} = 1$. This would imply $q \in I_r$, $r < j$ and $q > i$, in contradiction to condition (6.10). Thus, $z_{qr} = 0$ and $(1 - z_{qr}) = 1$ holds for all $(q,r) \in K$. Consequently, we obtain

$$\sum_{q=i+1}^{n} \sum_{r=0}^{j-1} (1 - z_{qr}) = \sum_{(q,r) \in K} (1 - z_{qr}) = |K| = (n-i) \cdot j = (n-i) \cdot j \cdot z_{ij}$$

and condition (6.7) is satisfied.

For the other direction, let us assume that condition (6.8) or (6.10) is violated and show that this implies that condition (6.5) or (6.7) is violated. If condition (6.8) is violated, then there exists an index $i \in \{1, \dots, n\}$ which is not assigned to any index set. Therefore, we have $\sum_{j=0}^{k} z_{ij} = 0$ and condition (6.5) is violated. On the other hand, if (6.10) is violated, then there exist $l, j \in \{1, ..., k\}$ with $l < j$ and indices $s \in I_l$ and $i \in I_j$ such that $s \geq i$. Consequently, we have that $z_{sl} = 1$ and $z_{ij} = 1$. Thus, either condition (6.5) is violated, or it follows that $s \neq i$ and therefore $s > i$ must be true. Now, since $(s, l) \in K$ and $z_{sl} = 1$ we obtain the equality

$$\sum_{q=i+1}^{n} \sum_{r=0}^{j-1} (1 - z_{qr}) = |K \setminus \{(s,l)\}| = |K| - 1 = (n - i) \cdot j - 1.$$

However, if condition (6.7) was satisfied, the inequality

$$\sum_{q=i+1}^{n} \sum_{r=0}^{j-1} (1 - z_{qr}) \geq (n - i) \cdot j$$

would have to be satisfied as well. Since this is not the case, condition (6.7) is violated. □

To summarize the above, a feasible binary vector z in problem (6.2) corresponds to a feasible partition of the data points, in the sense that conditions (6.8) to (6.10) are satisfied.

6.2.2 Continuity Restrictions

The interpretation of the variable w_{ij} is that $w_{ij} = 1$ holds if and only if $i \in I_j$ and $i + 1 \in I_{j+1}$, i.e., two successive indices are assigned to different (successive) index sets. Another way to say this is that $w_{ij} = 1$ holds if and only if there is a knot between the data points x_i and x_{i+1}. Consequently, the constraints

$$[w_{ij} = 1] \Longrightarrow [p(\gamma_i, \beta^{(j)}) - p(\gamma_i, \beta^{(j+1)}) = 0], \ \forall (i, j) \in \bar{J}, \tag{6.11}$$

$$[w_{ij} = 1] \Longrightarrow [p'(\gamma_i, \beta^{(j)}) - p'(\gamma_i, \beta^{(j+1)}) = 0], \ \forall (i, j) \in \bar{J}, \tag{6.12}$$

$$[w_{ij} = 1] \Longrightarrow [p''(\gamma_i, \beta^{(j)}) - p''(\gamma_i, \beta^{(j+1)}) = 0], \ \forall (i, j) \in \bar{J} \tag{6.13}$$

from problem (6.2) enforce the continuity conditions at a value $\gamma_i = (x_i + x_{i+1})/2$ only if x_i and x_{i+1} belong to different polynomials. Note that, as in problem (6.1), knots are midpoints between two consecutive design points that are assigned to different polynomials.

An intuitive way to model the variable w would be via the nonlinear constraints

$$w_{ij} = z_{ij} \cdot z_{i+1,j+1}, \ \forall (i,j) \in \bar{J}. \tag{6.14}$$

Fortunately, there is a simple technique to replace these nonlinear constraints with linear ones. In order to explain this technique, suppose we are given three binary variables a, b and c and the nonlinear constraint

$$a = b \cdot c.$$

Then the same relationship can be modeled with the linear constraints

$$\begin{aligned} a &\leq b, \\ a &\leq c, \\ a &\geq b + c - 1. \end{aligned}$$

Since w and z both are binary decision variables in problem (6.2), the constraints

$$\begin{aligned} w_{ij} &\leq z_{ij}, \ \forall (i,j) \in \bar{J}, \\ w_{ij} &\leq z_{i+1,j+1}, \ \forall (i,j) \in \bar{J}, \\ w_{ij} &\geq z_{ij} + z_{i+1,j+1} - 1, \ \forall (i,j) \in \bar{J}, \end{aligned}$$

are equivalent to the nonlinear constraints (6.14).

As mentioned previously, on could also use a big-M formulation to model the continuity restrictions (6.11) to (6.13). However, it is not clear how to obtain a sufficiently large value for the constant M without knowledge of the optimal solution. The numerical tests that we conducted showed that constants M that are large enough so that the optimal solution is not excluded from the feasible set often lead to numerical difficulties and long run times. In contrast, indicator constraints neither increased the computation time nor lead to numerical issues on the problem instances that we tested. Moreover, they are easier to use since no big-M constant has to be specified.

6.2.3 Generalized Epigraph Reformulation

In the following we explain how the epigraph reformulation gives rise to the linear objective function and the first two constraints in problem (6.2).

We first note that the objective function of problem (6.1) can be rewritten as

$$\sum_{j=0}^{k} \sum_{i \in I_j} \left(p(x_i, \beta^{(j)}) - y_i \right)^2 = \sum_{i=0}^{n} \sum_{j=0}^{k} z_{ij} \left(p(x_i, \beta^{(j)}) - y_i \right)^2 .$$

In combination with the results from the previous subsections, it is clear that problem (6.1) is equivalent to the problem

$$\begin{aligned}
\min_{\beta,z,w} \quad & \sum_{i=0}^{n}\sum_{j=0}^{k} z_{ij}\left(p(x_i,\beta^{(j)}) - y_i\right)^2 \\
\text{s.t.} \quad & \sum_{j=0}^{k} z_{ij} = 1,\ \forall i \in J_1, \\
& \sum_{i=1}^{n} z_{ij} \geq 1,\ \forall j \in J_2, \\
& \sum_{q=i+1}^{n}\sum_{r=0}^{j-1}(1 - z_{qr}) \geq (n-i)\cdot j \cdot z_{ij},\ \forall i \in J_1 \setminus \{n\}, j \in J_2 \setminus \{0\}, \\
& w_{ij} \leq z_{ij},\ \forall (i,j) \in \bar{J}, \\
& w_{ij} \leq z_{i+1,j+1},\ \forall (i,j) \in \bar{J}, \\
& w_{ij} \geq z_{ij} + z_{i+1,j+1} - 1,\ \forall (i,j) \in \bar{J}, \\
& [w_{ij} = 1] \implies [p(\gamma_i,\beta^{(j)}) - p(\gamma_i,\beta^{(j+1)}) = 0],\ \forall (i,j) \in \bar{J}, \\
& [w_{ij} = 1] \implies [p'(\gamma_i,\beta^{(j)}) - p'(\gamma_i,\beta^{(j+1)}) = 0],\ \forall (i,j) \in \bar{J}, \\
& [w_{ij} = 1] \implies [p''(\gamma_i,\beta^{(j)}) - p''(\gamma_i,\beta^{(j+1)}) = 0],\ \forall (i,j) \in \bar{J}, \\
& \beta \in \mathbb{R}^{4(k+1)}, z \in \{0,1\}^{n(k+1)}, w \in \{0,1\}^{(n-1)k}.
\end{aligned} \tag{6.15}$$

This objective function, however, has a difficult nonlinear and nonconvex structure. In order to improve upon this formulation, we introduce the additional decision variables $\alpha \in \mathbb{R}^n$, replace the objective function with $\sum_{i=0}^{n}\alpha_i$ and add the constraints

$$\sum_{j=0}^{k} z_{ij}\left(p(x_i,\beta^{(j)}) - y_i\right)^2 \leq \alpha_i,\ \forall i \in J_1. \tag{6.16}$$

This type of reformulation is sometimes referred to as the *generalized epigraph reformulation*, see, e.g., Stein (2018). Due to the first constraint in problem (6.11), we have that for each $i \in J_1$ there only exists a single index $j \in J_2$ such that $z_{ij} = 1$. Thus, the constraints (6.16) are equivalent to the quadratic indicator constraints

$$[z_{ij} = 1] \implies \left(p(x_i,\beta^{(j)}) - y_i\right)^2 \leq \alpha_i,\ \forall i \in J_1.$$

Since the quadratic functions are convex in the decision variables, the resulting problem is a convex MIQCP. However, so far optimization solvers can only handle linear indicator constraints. By introducing an additional decision variable $q \in \mathbb{R}^{n(k+1)}$ we arrive at the first two constraints of problem (6.2)

$$[z_{ij} = 1] \Longrightarrow [q_{ij} = p(x_i, \beta^{(j)}) - y_i], \ \forall (i,j) \in J_1 \times J_2, \tag{6.17}$$
$$q_{ij}^2 \leq \alpha_i, \ \forall (i,j) \in J_1 \times J_2, \tag{6.18}$$

and thus at the convex MIQCP (6.2), which can be solved with standard solvers.

Again, one might ask the question whether the indicator constraints (6.17) could be replaced by big-M constraints. But also in this case there is no obvious way to select a value for M that leads to good performance both in terms of computing time and solution quality. Therefore, we will not further consider big-M formulations in this thesis.

In our numerical experiments, the computing time decreased significantly when bounds were added to the decision variable q, i.e., when the additional constraints $-\bar{q} \leq q \leq \bar{q}$ were imposed for some $\bar{q} > 0$. If the optimal solution is not known, this is not possible without the risk of excluding the optimal solution from the feasible set. However, bounds on q only lead to the exclusion of the optimal solution of problem (6.2) if in this optimal solution there exist $(i,j) \in J_1 \times J_2$ with $z_{ij} = 1$ such that $|p(x_i, \beta^{(j)}) - y_i| > \bar{q}$, i.e., the y component of a data point (x_i, y_i) that is assigned to the jth polynomial in the optimal solution has to have a distance to the polynomial function evaluated at x_i that is larger than $\bar{q}$. So, by setting $\bar{q} := \max_i y_i - \min_i y_i$, one can argue that in typical applications these bounds should not affect the optimal solution. Indeed, if the optimal solution does not satisfy these bounds, this might be an indication that there are outliers in the data or that the spline function with the chosen number of free knots is not appropriate for the approximation of the given data set. Note that this same reasoning is not possible for the bounds imposed by big-M constants, since they have to hold for all $(i,j) \in J_1 \times J_2$, and not only for those that satisfy $z_{ij} = 1$.

Lastly, we mention that, due to the convexity of the quadratic constraints, it is straightforward to linearize these constraints and in this way approximate the convex MIQCP with an mixed-integer linear problem (MILP). Since we already impose bounds on q in our MIQCP formulation, we can simply choose m points $r_t \in [-\bar{q}, \bar{q}]$, compute for $t = 1, \ldots, m-1$, the parameters

$$a_t = \frac{r_{t+1}^2 - r_t^2}{r_{t+1} - r_t},$$
$$b_t = r_t^2 - a_t \cdot r_t,$$

of the secants passing through the points (r_t, r_t^2), $t = 1, \ldots, m$, and replace the quadratic constraints

$$q_{ij}^2 \leq \alpha_i,\ \forall (i,j) \in J_1 \times J_2,$$

with the linear constraints

$$a_t q_{ij} + b_k \leq \alpha_i,\ \forall (i,j) \in J_1 \times J_2,\ t = 1, \ldots, m-1.$$

We tested this approach for different choices of m and points $r_t \in [-\bar{q}, \bar{q}]$. However, it was not possible to obtain a decrease in the computing time without significantly worsening the quality of the obtained solution. Therefore, we will not consider this linearization further in this thesis.

6.3 Alternative Error Functions

The main focus of this thesis lies on the computation of spline functions that minimize the least-squares error. Nevertheless, in this section we want to consider two error functions that could be alternatives to the least-squares objective function

$$\sum_{j=0}^{k} \sum_{i \in I_j} \left(p(x_i, \beta^{(j)}) - y_i \right)^2,$$

and we explain what implication the choice of error function has for the spline approximation problem with fixed knots and the mixed-integer formulation of the combinatorial formulation of the spline approximation with free knots.

6.3.1 Absolute Error Function

First we consider the absolute error function

$$\sum_{j=0}^{k} \sum_{i \in I_j} |p(x_i, \beta^{(j)}) - y_i|,$$

which is less sensitive to outliers than the least-squares error function and therefore more appropriate in applications where a certain robustness with respect to outliers in the data is required.

The absolute error spline approximation problem with fixed knots

$$\begin{aligned}
\min_{\beta} \quad & \sum_{j=0}^{k} \sum_{i \in I_j} |p(x_i, \beta^{(j)}) - y_i| \\
\text{s.t.} \quad & p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& \beta \in \mathbb{R}^{4(k+1)},
\end{aligned}$$

can be equivalently reformulated via the generalized epigraph reformulation in order to obtain the linear optimization problem (LP)

$$\begin{aligned}
\min_{\beta,\ \alpha} \quad & \sum_{j=0}^{k} \sum_{i \in I_j} \alpha_{ji} \\
\text{s.t.} \quad & p(x_i, \beta^{(j)}) - y_i \leq \alpha_{ji},\ j = 0, \dots, k,\ i \in I_j \\
& y_i - p(x_i, \beta^{(j)}) \leq \alpha_{ji},\ j = 0, \dots, k,\ i \in I_j \\
& p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& \beta \in \mathbb{R}^{4(k+1)}, \alpha \in \mathbb{R}^{n(k+1)}.
\end{aligned}$$

Therefore, just like the optimal least-squares spline with fixed knots can be computed very efficiently by solving a linear system of equations, the absolute error spline approximation problem with fixed knots can be solved to optimality very efficiently by solving an LP.

In the case of the problem with free knots, a similar derivation as in Section 6.2 leads to a mixed integer formulation of the combinatorial formulation of the absolute error spline approximation problem, where the only difference to problem (6.2) is that the first two constraints

$$\begin{aligned}
& [z_{ij} = 1] \implies [q_{ij} = p(x_i, \beta^{(j)}) - y_i],\ \forall (i,j) \in J_1 \times J_2, \\
& q_{ij}^2 \leq \alpha_i,\ \forall (i,j) \in J_1 \times J_2,
\end{aligned}$$

are replaced with the constraints

$$\begin{aligned}
& [z_{ij} = 1] \implies [p(x_i, \beta^{(j)}) - y_i \leq \alpha_i],\ \forall (i,j) \in J_1 \times J_2, \\
& [z_{ij} = 1] \implies [y_i - p(x_i, \beta^{(j)}) \leq \alpha_i],\ \forall (i,j) \in J_1 \times J_2,
\end{aligned}$$

and the decision variable q is removed from the problem. Although this is only a small modification, a major difference to problem (6.2) is that this new problem is an MILP, since there are no quadratic constraints anymore.

6.3.2 Maximum Error Function

The second error function we consider is the maximum error function

$$\max_{j\in\{0,\ldots,k\}} \max_{i\in I_j} \ |p(x_i, \beta^{(j)}) - y_i|,$$

which is more sensitive to outliers than the least-squares error function and can be more appropriate in applications where outliers occur due to the variability of the underlying process and not because of measurement errors.

The maximum error spline approximation problem with fixed knots can be formulated as the LP

$$\begin{aligned}
\min_{\beta,\ \alpha} \quad & \alpha \\
\text{s.t.} \quad & p(x_i, \beta^{(j)}) - y_i \leq \alpha,\ j = 0, \ldots, k,\ i \in I_j \\
& y_i - p(x_i, \beta^{(j)}) \leq \alpha,\ j = 0, \ldots, k,\ i \in I_j \\
& p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j = 1, \ldots, k, \\
& p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j = 1, \ldots, k, \\
& p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j = 1, \ldots, k, \\
& \beta \in \mathbb{R}^{4(k+1)}, \alpha \in \mathbb{R}.
\end{aligned}$$

Also in this case, an epigraph reformulation was used in order to obtain this formulation.

The mixed-integer formulation in the case of free knots is again quite similar to the MIQCP (6.2), the only difference being that the decision variable $\alpha \in \mathbb{R}$ is minimized in the objective function and the constraints

$$\begin{aligned}
&[z_{ij} = 1] \Longrightarrow [q_{ij} = p(x_i, \beta^{(j)}) - y_i],\ \forall (i,j) \in J_1 \times J_2, \\
&q_{ij}^2 \leq \alpha_i,\ \forall (i,j) \in J_1 \times J_2,
\end{aligned}$$

are replaced with the constraints

$$\begin{aligned}
&[z_{ij} = 1] \Longrightarrow [p(x_i, \beta^{(j)}) - y_i \leq \alpha],\ \forall (i,j) \in J_1 \times J_2, \\
&[z_{ij} = 1] \Longrightarrow [y_i - p(x_i, \beta^{(j)}) \leq \alpha],\ \forall (i,j) \in J_1 \times J_2.
\end{aligned}$$

Thus, both of the considered alternative objective functions lead to MILPs that can be expected to be easier to solve than the convex MIQCP (6.2). However, the empirical evaluation of this claim is beyond the scope of this thesis.

Chapter 7

A Branch-and-Bound Algorithm for the Combinatorial Formulation

In this chapter our branch-and-bound (B&B) algorithm for the solution of the combinatorial formulation (6.1) of the least-squares spline approximation problem with free knots is described. We start by sketching the main idea of B&B algorithms before going into the details our algorithm. A general description of branch-and-bound methods in combinatorial optimization can be found in the books Nemhauser and Wolsey (1988), Bertsimas and Tsitsiklis (1997), Papadimitriou and Steiglitz (1982).

7.1 General Outline of the Branch-and-Bound Algorithm

The main idea of B&B algorithms is to subsequently divide the problem into subproblems (*branching*) and compute lower bounds of the minimal values of those subproblems. All problems are stored in a list together with the corresponding lower bound. In case there are subproblems in the list whose lower bound exceeds a known upper bound for the optimal value of problem (6.1), these can be excluded from the list without further consideration as none of their subproblems can contain a globally minimal point. This is also known as *fathoming* or *pruning.* The algorithm terminates with the optimal solution after a finite number of steps.

7.2 Branching Strategy

As a first step in describing our branching strategy, we define the subproblems into which the problem is subsequently divided in our B&B algorithm.

For given index sets $\Lambda_0, \dots, \Lambda_k \subseteq \{1, \dots, n\}$ we formulate the optimization problem

$$\begin{aligned}
S(\Lambda_0, \dots, \Lambda_k) : \min_{\beta, I_0, \dots, I_k} \quad & \sum_{j=0}^{k} \sum_{i \in I_j} \left(p(x_i, \beta^{(j)}) - y_i \right)^2 \\
\text{s.t.} \quad & \Lambda_j \subseteq I_j, \quad j = 0, \dots, k, \\
& \bigcup_{j=0}^{k} I_j = \{1, \dots, n\}, \\
& I_j \neq \emptyset, \quad j = 0, \dots, k, \\
& r < q \text{ for all } r \in I_i \text{ and } q \in I_j \text{ with } i < j, \\
& p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j = 1, \dots, k, \\
& \beta \in \mathbb{R}^{4(k+1)},
\end{aligned}$$

where

$$\xi_j = (\max_{i \in I_{j-1}} x_i + \min_{i \in I_j} x_i)/2,$$

for all $j \in \{1, \dots, k\}$. We call the index sets $\Lambda_0, \dots, \Lambda_k$ restriction sets, since by adding indices to these sets we restrict the set of feasible partitions in problem $S(\Lambda_0, \dots, \Lambda_k)$.

Definition 7.2.1 *Restriction sets* $\Lambda_0, \dots, \Lambda_k \subseteq \{1, \dots, n\}$ *are called* valid*, if there exists a partition* $I_0, \dots, I_k$ *of the set* $\{1, \dots, n\}$ *such that the conditions*

$$\Lambda_j \subseteq I_j, \quad j = 0, \dots, k, \tag{7.1}$$

$$I_j \neq \emptyset, \quad j = 0, \dots, k, \tag{7.2}$$

$$r < q \textit{ for all } r \in I_i \textit{ and } q \in I_j \textit{ with } i < j, \tag{7.3}$$

are satisfied.

The basic idea of our B&B algorithm is to subsequently assign indices to the restriction sets in order to split problem (6.1) into subproblems $S(\Lambda_0, \dots, \Lambda_k)$ that allow the efficient computation of nontrivial lower bounds.

Since we can without loss of generality assume that the first point x_1 is assigned to the first polynomial, i.e., $1 \in I_0$, and the last point x_n is assigned to the last polynomial, i.e., $n \in I_k$, we start the B&B algorithm with the initial restriction sets $\Lambda_0 := \{1\}$, $\Lambda_1 := \emptyset, \ldots, \Lambda_{k-1} := \emptyset, \Lambda_k := \{n\}$ and the set of unassigned indices $I = \{2, ..., n-1\}$.

In order to divide a subproblem $S(\Lambda_0, \ldots, \Lambda_k)$ further into subproblems, we choose an index $i \in I$ and assign it to one of the restriction sets Λ_0,...,Λ_k. When we add an index i to a restriction set Λ_j for which $i < \min(\Lambda_j)$ holds, then we also add the indices $\{i+1, \ldots, \min(\Lambda_j) - 1\}$ to Λ_j, since those indices cannot be assigned to another restriction set without making $S(\Lambda_0, \ldots, \Lambda_k)$ infeasible. Analogously, if $i > \max(\Lambda_j)$, then we add the indices $\{\max(\Lambda_j) + 1, \ldots, i-1\}$ to Λ_j. Note that this implies that an index which is added to a restriction set is always the smallest or largest index in this restriction set. The following lemma gives simple conditions that ensure that the resulting restriction sets are valid, i.e., the new subproblem still permits feasible partitions.

Lemma 7.2.2 *Let $\Lambda_0, \ldots, \Lambda_k \subset \{1, ..., n\}$ be valid restriction sets, $1 \in \Lambda_0$, $n \in \Lambda_k$ and $j \in \{0, ..., k\}$. In addition, let $i \in \{1, ..., n\}$ be an index such that $i \notin \Lambda_m$ for all $m \in \{0, \ldots, k\}$. Then the sets $\bar{\Lambda}_j := \Lambda_j \cup \{i\}$, $\bar{\Lambda}_m := \Lambda_m$, $\forall m \neq j$, are valid restriction sets if and only if the following conditions are satisfied:*

$$i < \min(\Lambda_j) \implies \max(\Lambda_\ell) \leq i + \ell - j, \tag{7.4}$$

$$i > \max(\Lambda_j) \implies \min(\Lambda_r) \geq i + r - j, \tag{7.5}$$

where $\ell := \max\{\ m \mid \Lambda_m \neq \emptyset,\ m \in \{0, \ldots, j-1\}\}$ and $r := \min\{\ m \mid \Lambda_m \neq \emptyset,\ m \in \{j+1, \ldots, k\}\}$.

Proof. Since the sets $\Lambda_0, \ldots, \Lambda_k$ are valid restriction sets, there exists a partition $I_0, \ldots, I_k$ of the set $\{1, ..., n\}$ such that the conditions (7.1) to (7.3) are satisfied.

We show that if $i < \min(\Lambda_j)$, then condition (7.4) is satisfied if and only if the sets $\bar{\Lambda}_0, \ldots, \bar{\Lambda}_k$ are valid restriction sets. The argumentation that if $i > \max(\Lambda_j)$ condition (7.5) is satisfied if and only if the sets $\bar{\Lambda}_0, \ldots, \bar{\Lambda}_k$ are valid restriction sets is analogous and therefore omitted.

Thus, we assume that $i < \min(\Lambda_j)$ is true. Since $i < \min(\Lambda_j)$ and $1 \in \Lambda_0$ it follows that $j \geq 1$ and the set $\{\ m \mid \Lambda_m \neq \emptyset,\ m \in \{0, \ldots, j-1\}\}$ is nonempty, i.e., the index ℓ is well defined.

Algorithm 7.1 Branching strategy

```
 1: Input: Restriction sets $\Lambda_0, \dots, \Lambda_k$;
 2: Choose an unassigned index $i \in \{1, \dots, n\}$ (i.e., an index which is not
    yet contained in one of the restriction sets $\Lambda_0, \dots, \Lambda_k$);
 3: for $j = 0, \dots, k$ do
 4:     if $i$ and $j$ satisfy conditions (7.4) and (7.5) then
 5:         if $i < \min(\Lambda_j)$ then
 6:             Set $\widetilde{\Lambda}_j := \Lambda_j \cup \{i, \dots, \min(\Lambda_j) - 1\}$;
 7:         else
 8:             Set $\widetilde{\Lambda}_j := \Lambda_j \cup \{\max(\Lambda_j) + 1, \dots, i\}$;
 9:         end if
10:         Add $S(\Lambda_0, \dots, \Lambda_{j-1}, \widetilde{\Lambda}_j, \Lambda_{j+1}, \dots, \Lambda_k)$ to list of subproblems $\mathcal{P}$;
11:     end if
12: end for
13: Output: List of new subproblems $\mathcal{P}$;
```

If we assume that condition (7.4) is satisfied, we can define the sets

$$\bar{I}_m := \begin{cases} \{\min(\Lambda_\ell), \dots, i + \ell - j\}, & \text{if } m = \ell, \\ \{i + m - j\}, & \text{if } \ell < m < j, \\ \{i, \dots, \max(\Lambda_j)\}, & \text{if } m = j, \\ I_m, & \text{if } m < \ell \text{ or } j < m, \end{cases}$$

for all $m \in \{0, \dots, k\}$. The sets $\bar{I}_0, \dots, \bar{I}_k$ form a partition of the set $\{1, ..., n\}$ such that the conditions (7.1) to (7.3) are satisfied for the restriction sets $\bar{\Lambda}_0, \dots, \bar{\Lambda}_k$. Consequently, these restriction sets are valid.

On the other hand, assume that condition (7.4) is violated, i.e., $\max(\Lambda_\ell) \geq i + \ell - j + 1$. If $\max(\Lambda_\ell) > i$, it is clear that there exists no partition $I_0, \dots, I_k$ of the set $\{1, ..., n\}$ such that condition (7.3) is satisfied. If $\max(\Lambda_\ell) < i$, then $i + \ell - j + 1 \leq \max(\Lambda_\ell) < i$ and thus there are $j - \ell - 1 > 0$ empty restriction sets between the sets Λ_ℓ and Λ_j. In every partition $I_0, \dots, I_k$ of the set $\{1, ..., n\}$ that satisfies conditions (7.1) and (7.3) the $j - \ell - 1$ index sets I_m with $\ell < m < j$ cannot all be nonempty, since there are only $i - \max(\Lambda_\ell) - 1 \leq j - l - 2$ unassigned indices that can be assigned to these sets without violating condition (7.1) or (7.3). But then condition (7.2) is violated and the restriction sets $\bar{\Lambda}_0, \dots, \bar{\Lambda}_k$ are not valid. □

Our branching strategy is described more formally in Algorithm 7.1. Note that there are different possibilities to choose an unassigned index in Algorithm 7.1. We propose to take the midpoint of the (ordered) set of currently unassigned indices.

If one obtains restriction sets $\Lambda_0, \dots, \Lambda_k$ with Algorithm 7.1 such that between two nonempty sets Λ_ℓ and Λ_r there are $q = \min(\Lambda_r) - \max(\Lambda_\ell) - 1 > 0$ empty sets $\Lambda_m = \emptyset$ with $\ell < m < r$, then one can set $\Lambda_m := \{\max(\Lambda_\ell) + m - \ell\}$ for $\ell < m < r$, since this is the only way to assign these indices such that one obtains valid restriction sets. We made use of this in our implementation, but it is not explicitly mentioned in Algorithm 7.1.

Example 7.2.3 *Suppose we are given the restriction sets* $\Lambda_0 = \{1, 2\}, \Lambda_1 = \emptyset, \Lambda_2 = \emptyset$ *and* $\Lambda_3 = \{9\}$. *Assume that in Algorithm 7.1 the so far unassigned index* $i = 5$ *is assigned to* Λ_3. *The restriction sets obtained with Algorithm 7.1 are then* $\Lambda_0 = \{1, 2\}, \Lambda_1 = \emptyset, \Lambda_2 = \emptyset$ *and* $\widetilde{\Lambda}_3 = \{5, 6, 7, 8, 9\}$. *However, in addition to that, the empty index sets* Λ_1 *and* Λ_2 *can be filled and we obtain the restriction sets* $\Lambda_0 = \{1, 2\}, \widetilde{\Lambda}_1 = \{3\}, \widetilde{\Lambda}_2 = \{4\}$ *and* $\widetilde{\Lambda}_3 = \{5, 6, 7, 8, 9\}$.

7.3 Computation of Lower Bounds

In order to obtain a nontrivial lower bound of the optimal value of problem $S(\Lambda_0, \dots, \Lambda_k)$ for given restriction sets $\Lambda_0, \dots, \Lambda_k$, we compute the optimal value of the optimization problem

$$
\begin{aligned}
L(\Lambda_0, \dots, \Lambda_k): \quad \min_{\beta} \quad & \sum_{j=0}^{k} \sum_{i \in \Lambda_j} \left(p(x_i, \beta^{(j)}) - y_i \right)^2 \\
\text{s.t.} \quad & p(\xi_j, \beta^{(j-1)}) = p(\xi_j, \beta^{(j)}), \quad j \in J(\Lambda_0, \dots, \Lambda_k), \\
& p'(\xi_j, \beta^{(j-1)}) = p'(\xi_j, \beta^{(j)}), \quad j \in J(\Lambda_0, \dots, \Lambda_k), \\
& p''(\xi_j, \beta^{(j-1)}) = p''(\xi_j, \beta^{(j)}), \quad j \in J(\Lambda_0, \dots, \Lambda_k), \\
& \beta \in \mathbb{R}^{4(k+1)},
\end{aligned}
$$

where

$$\xi_j := (\max_{i \in I_{j-1}} x_i + \min_{i \in I_j} x_i)/2,$$

for all $j \in J(\Lambda_0, \dots, \Lambda_k)$ and

$$J(\Lambda_0, \dots, \Lambda_k) := \{ j \in \{1, \dots, k\} \mid 1 + \max(I_{j-1}) = \min(I_j) \}.$$

Note that $L(\Lambda_0, \dots, \Lambda_k)$ is a least-squares spline approximation problem with fixed knots, and can be solved analogously to problem (5.2). The only difference is that only the knots ξ_j with $j \in J(\Lambda_0, \dots, \Lambda_k)$ and the corresponding continuity restrictions are included in the problem. Note that the index set $J(\Lambda_0, \dots, \Lambda_k)$ contains only the indices of knots which are not subject to change when the problem $S(\Lambda_0, \dots, \Lambda_k)$ is further divided into subproblems in subsequent branching steps.

Lemma 7.3.1 *Let v_L be the optimal value of problem $L(\Lambda_0, \dots, \Lambda_k)$ and v_S the optimal value of problem $S(\Lambda_0, \dots, \Lambda_k)$. Then $v_L \leq v_S$.*

Proof. Let $\beta^\star$ and $I_0^\star, ..., I_k^\star$ denote an optimal solution of $S(\Lambda_0, \dots, \Lambda_k)$. Since the partition $I_0^\star, ..., I_k^\star$ is feasible for problem $S(\Lambda_0, \dots, \Lambda_k)$, it must hold that $\Lambda_j \subseteq I_j^\star$ for all $j \in \{0, \dots, k\}$. Therefore the inequality

$$\sum_{j=0}^{k} \sum_{i \in \Lambda_j} \left(p(x_i, \beta^{(j),\star}) - y_i\right)^2 \leq \sum_{j=0}^{k} \sum_{i \in I_j^\star} \left(p(x_i, \beta^{(j),\star}) - y_i\right)^2$$

is satisfied. Due to $J(\Lambda_0, \dots, \Lambda_k) \subseteq \{1, \dots, k\}$ we have that every vector β that is feasible for problem $S(\Lambda_0, \dots, \Lambda_k)$ is also feasible for problem $L(\Lambda_0, \dots, \Lambda_k)$. Consequently, the optimal solution $\beta^\star$ is feasible for problem $L(\Lambda_0, \dots, \Lambda_k)$ and we obtain that

$$v_L \leq \sum_{j=0}^{k} \sum_{i \in \Lambda_j} \left(p(x_i, \beta^{(j),\star}) - y_i\right)^2 \leq \sum_{j=0}^{k} \sum_{i \in I_j^\star} \left(p(x_i, \beta^{(j),\star}) - y_i\right)^2 = v_S.$$

□

The optimal value of $L(\Lambda_0, \dots, \Lambda_k)$ is increasing as more indices are added to the restriction sets $\Lambda_0, \dots, \Lambda_k$. When the set of unassigned indices is empty, then the optimal value of $L(\Lambda_0, \dots, \Lambda_k)$ coincides with the optimal value of the subproblem $S(\Lambda_0, \dots, \Lambda_k)$. Consequently, we can use this technique to efficiently compute lower bounds for the subproblems and we obtain a B&B algorithm that terminates with an optimal solution after a finite number of steps.

We also want to shortly comment on the use of other error functions, as we did for the mixed-integer formulation in Section 6.3. For both error functions presented in Section 6.3, problem $L(\Lambda_0, \dots, \Lambda_k)$ becomes an LP. And although LPs can be solved very efficiently via simplex or interior point methods, the time necessary to compute an optimal solution of $L(\Lambda_0, \dots, \Lambda_k)$ should be expected to increase. However, possibly by incorporating a technique for warm starting the solution process of the linear programming solver, the B&B approach could still be a competitive method in these cases.

Algorithm 7.2 Branch-and-bound algorithm for problem (6.1)

```
 1: Input: $n$ data points $(x_i, y_i) \in \mathbb{R}^2$, number of free knots $k$;
 2: Initialize list of subproblems as $\mathcal{L} := \{\big(l_S, S(\Lambda_0, \ldots, \Lambda_k)\big)\}$, where $l_S := 0$
    and $\Lambda_0 := \{1\}, \Lambda_1 := \emptyset, \ldots, \Lambda_{k-1} := \emptyset, \Lambda_k := \{n\}$;
 3: Use a heuristic method to determine initial partition $(I_0^\star, \ldots, I_k^\star)$ and
    initial upper bound $u \in [0, \infty)$;
 4: while $\mathcal{L} \neq \emptyset$ do
 5:     Select $\big(l_S, S(\Lambda_0, \ldots, \Lambda_k)\big) \in \mathcal{L}$ with smallest lower bound $l_S$;
 6:     Create set $\mathcal{P}$ of new subproblems using Algorithm 7.1;
 7:     for $S(\Lambda_0, \ldots, \Lambda_k) \in \mathcal{P}$ do
 8:         Compute the optimal value $l_S$ of problem $L(\Lambda_0, \ldots, \Lambda_k)$;
 9:         if $l_S < u$ then
10:             Add $\big(l_S, S(\Lambda_0, \ldots, \Lambda_k)\big)$ to $\mathcal{L}$;
11:             if $\bigcup_{j=0}^{k} \Lambda_j = \{1, \ldots, n\}$ then
12:                 Set $u := l_S$ and $I_0^\star := \Lambda_0, \ldots, I_k^\star := \Lambda_k$;
13:             end if
14:         end if
15:     end for
16:     for $\big(l_S, S(\Lambda_0, \ldots, \Lambda_k)\big) \in \mathcal{L}$ with $l_S \geq u$ do
17:         Remove $\big(l_S, S(\Lambda_0, \ldots, \Lambda_k)\big)$ from $\mathcal{L}$;
18:     end for
19: end while
20: Output: Globally optimal partition $(I_0^\star, \ldots, I_k^\star)$ for problem (6.1);
```

7.4 The Branch-and-Bound Algorithm

In Algorithm 7.2 our branch-and-bound method to solve the combinatorial formulation (6.1) of the least-squares spline problem with free knots is described formally.

The initial partition and initial upper bound can be determined by any heuristic method, or one could simply choose equally spaced knots. Moreover, in addition to the upper bounds obtained in Algorithm (7.2) when the set of unassigned indices is empty, every subproblem $S(\Lambda_0, \ldots, \Lambda_k)$ can be used to compute an upper bound for the globally minimal value of problem (6.1). One only has to generate new restriction sets $\bar{\Lambda}_0, \ldots, \bar{\Lambda}_k$ such that all remaining indices are assigned, i.e., $\bigcup_{j=0}^{k} \bar{\Lambda}_j = \{1, ..., n\}$, and $\Lambda_j \subseteq \bar{\Lambda}_j$ holds for all $j \in \{0, ..., k\}$. Then the optimal value of problem $L(\bar{\Lambda}_0, \ldots, \bar{\Lambda}_k)$ is an upper bound for the globally minimal value of problem (6.1).

In our numerical tests upper bounds only had a minor influence on the runtime of the algorithm. Therefore, we did not compute additional upper bounds during the optimization process. However, upper bounds can help to keep the list of subproblems $\mathcal{L}$ short and ensure that the main memory does not become a limiting factor.

Chapter 8

Numerical Experiments

In this chapter we present numerical experiments that aim to answer the question whether the combinatorial approach to the least-squares spline approximation problem with free knots makes it possible to compute high-quality solutions to problems of realistic sizes within reasonable computing times. First we compare the quality of the optimal solution of the combinatorial formulation with the results obtained with existing approaches. Then we investigate whether the B&B or the MIQCP approach are more efficient for the computation of the optimal solution of the combinatorial formulation (6.1) of the spline approximation problem. Finally, we also compare the B&B and MIQCP approaches concerning the computation of optimal least-squares piecewise polynomials, without any continuity restrictions.

8.1 Comparison with Existing Approaches

We consider four different approaches in this section: A variant of the Fletcher-Reeves nonlinear conjugate gradient method (FR) presented in Dierckx (1996), the continuous genetic algorithm (CG) proposed in Spiriti et al. (2013), the combinatorial formulation (CF) presented in Chapter 6 and a combination of the combinatorial formulation with the Fletcher-Reeves nonlinear conjugate gradient method (CF+FR).

We use two real data sets from the literature to empirically compare the quality of the solutions obtained with each of these approaches. Since the goal is to compare the quality of the solutions and not the computing times, it is possible to use implementations of the methods in different programming languages. For the CG method we use the implementation in the R-package *freeknotsplines* by Spiriti et al. (2018). For the other methods we use our own Python implementations. The FR method is always initialized with the

number of knots	FR	CG	CF	CF+FR
2	2.1157	2.0815	2.0741	**2.0703**
3	1.9209	0.5104	0.5006	**0.4651**
4	0.0904	0.0669	0.0681	**0.0654**
5	0.0962	**0.0077**	0.0093	0.0088

Table 8.1: Least-squares errors of the solutions obtained with the FR, CG, CF and CF+FR methods when applied to the *titanium heat* data set.

equidistant knot placement.

Titanium Heat Data

The *titanium heat* data set was introduced by de Boor and Rice (1968) as an example of a data set that is difficult to approximate with classical methods. It is actual data that expresses a thermal property of titanium. It consists of 49 data points and contains a significant amount of noise.

Table 8.1 shows the least-squares errors of the spline functions corresponding to the knot placements computed with the four approaches for different numbers of free knots. We can observe that if the FR method is initialized with the equidistant knot placement, it approximates local solutions with a substantially worse least-squares error then the other methods, in particular for three and five free knots. The CG and CF approaches both yield high-quality solutions, with the CF approach being superior for two and three free knots and slightly inferior for four and five free knots. The CF+FR approach, where we use the FR method to locally improve the knot placement obtained with the CF approach, yields the best knot placement for all problems except the one with five free knots, where it obtains a local solution slightly inferior to the solution obtained with the CG approach.

The CG and CF approaches both seem to find solutions close to the globally optimal solution and therefore both of them could be combined with the FR method in order to find high-quality solutions. However, there is a theoretically and practically relevant difference. Whereas there is no optimality guarantee except local optimality if we use the knot placement obtained with the CG approach as the initial solution of the FR method, the CF+FR approach guarantees that the approximated local solution is at least as good as the optimal solution of the combinatorial formulation.

In Figure 8.1 the spline functions corresponding to the knot placements computed with the four methods are depicted for the case of three free knots. Clearly, the solution obtained by the FR method is not satisfactory. The other three methods yield similarly good approximations, the only visible difference being the approximation quality near the peak in the data set.

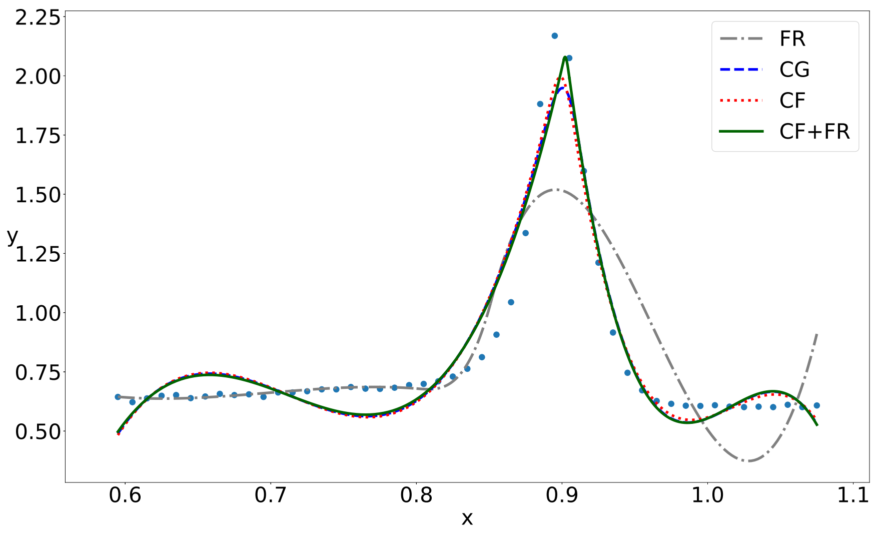


Figure 8.1: Spline functions with three free knots corresponding to the solutions obtained with the FR, CG, CF and CF+FR methods when applied to the *titanium heat* data set.

Angular Displacement Data

The *angular displacement* data set was introduced in Pezzack et al. (1977) for the evaluation of differentiation methods of film data of human motion. We use the modification of this data set proposed by Lanshammar (1982), which contains larger errors and is supposed to be more realistic than the original data set. It consists of 142 data points.

In Table 8.2 the least-squares errors of the spline functions corresponding to the knot placements computed with the four approaches for different numbers of free knots are presented. The overall picture is similar to the one obtained from the *titanium heat* data set. The FR method seems to approximate the globally minimal point for two and three free knots, even when it is initialized with the equidistant knot placement. However, the least-squares error of the locally minimal point approximated in case of four and five free knots is much worse than the result obtained with the other methods. The

number of knots	FR	CG	CF	CF+FR
2	**11.4483**	11.4486	11.4496	**11.4483**
3	1.5298	1.5335	1.5384	**1.5297**
4	1.5266	0.1475	0.1493	**0.1465**
5	1.3663	**0.0544**	0.0552	0.0545

Table 8.2: Least-squares errors of the solutions obtained with the FR, CG, CF and CF+FR methods when applied to the *angular displacement* data set.

CG and CF methods both find high-quality solutions in all cases and are only slightly inferior to the combined CF+FR method.

The similarity of the results obtained with the CG, CF and CF+FR approaches is also visible in Figure 8.2, where only two curves can be distinguished: the suboptimal fit obtained with the FR method initialized with the equidistant knot placement, and the high quality approximation obtained with the other methods.

We stress again that while all methods, except the purely local FR method, compute high-quality solutions for this problem, only the CF and CF+FR approaches yield solutions that are guaranteed to be as good or better than the optimal solution of the combinatorial formulation.

8.2 Comparison of MIQCP and Branch-and-Bound Approaches

In the previous section we saw that it is possible to obtain high-quality solutions of the least-squares spline approximation problem with free knots by solving the combinatorial reformulation of the continuous original problem. In this section we want to determine whether the MIQCP or the B&B approach is more efficient for the solution of the combinatorial formulation. First we consider the two real data sets from the previous section to gain some preliminary insights into the relative performance of the approaches. Then we use a larger testbed of synthetic data sets in order to decide which approach should be preferred in practice.

The B&B algorithm was implemented in Python 3.6. We used the commercial optimization solver Gurobi (version 8.1.1) and its Python API to model and solve the MIQCPs. The computations were performed on an Intel Core i7-9700K with 32 GB of main memory. The time limit for each

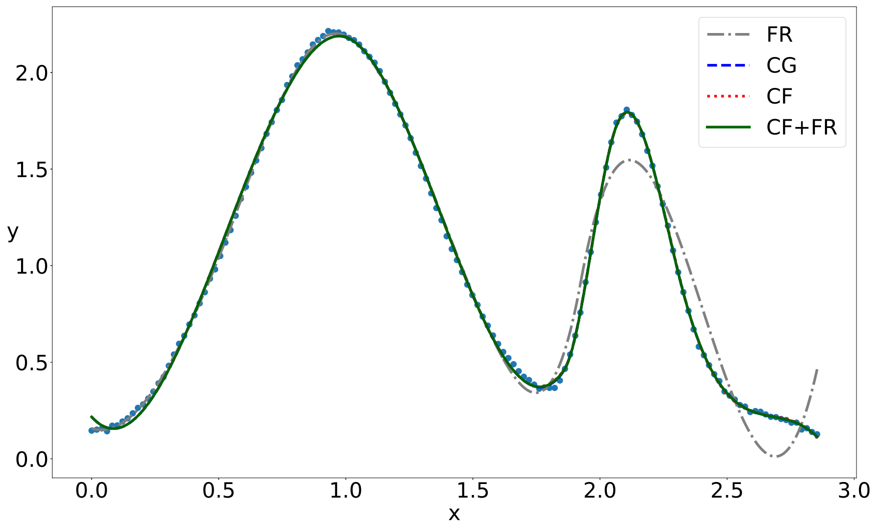


Figure 8.2: Spline functions with five free knots corresponding to the solutions obtained with the FR, CG, CF and CF+FR methods when applied to the *angular displacement* data set.

algorithm was set to 3600 seconds.

8.2.1 Titanium Heat and Angular Displacement Data Sets

Tables 8.3 and 8.4 show the least-squares errors of the knot placements computed via the B&B and MIQCP approaches for the *titanium heat* and *angular displacement* data sets and the corresponding CPU times. As was to be expected, both algorithms seem to approximate the same optimal solutions, except for the case of five free knots and the displacement data set, where the MIQCP approach is not able to compute the optimal solution within the given time limit. The main observation is that the B&B algorithm needs significantly less CPU time than the MIQCP approach. It is consistently faster by an order of magnitude.

We can also make some preliminary observations concerning the growth in computation time when the number of data points or free knots is increased. The *angular displacement* data set contains almost exactly three times as many data points as the *titanium heat* data set. This leads to about a tenfold increase in computation time for the B&B method. In contrast, the

	B&B		MIQCP	
number of knots	time	error	time	error
2	**0.07**	2.0741	0.88	2.0741
3	**0.40**	0.5006	2.00	0.5006
4	**1.99**	0.0681	9.31	0.0681
5	**5.55**	0.0093	12.09	0.0093

Table 8.3: CPU times (in seconds) and least-squares errors of the solutions obtained with the B&B and MIQCP approaches for the *titanium heat* data set.

	B&B		MIQCP	
number of knots	time	error	time	error
2	**0.61**	11.4496	78.77	11.4496
3	**8.14**	1.5384	379.31	1.5384
4	**44.95**	0.1493	1250.79	0.1493
5	**446.37**	**0.0552**	3600.01	0.0563

Table 8.4: CPU times (in seconds) and least-squares errors of the solutions obtained with the B&B and MIQCP approaches for the *angular displacement* data set.

computation time of the MIQCP method increases a hundredfold. However, increasing the number of free knots by one leads to an increase in the computing time of the B&B algorithm by a factor between 5 and 10, whereas the computing time of the MIQCP approach increases only by a factor between 1.5 and 5. We will further investigate the dependency of the computation time on the number of data points and knots in the next section.

8.2.2 Synthetic Data Sets

The experiments on the *titanium heat* and *angular displacement* data sets gave a first impression about the relative performance of the B&B and MIQCP approaches. In this section we present the results of more extensive numerical experiments with synthetic data sets. The functions used to

number of data points	number of free knots
50	$\{2, 3, 4, 5, 6, 7\}$
100	$\{2, 3, 4, 5, 6\}$
200	$\{2, 3, 4, 5\}$
400	$\{2, 3, 4\}$
800	$\{2, 3\}$

Table 8.5: All the combinations of data set sizes and number of free knots that were considered when generating instances of the spline approximation problem with the functions in Table A.2.

generate the synthetic data sets are listed in Table A.2 in the appendix, along with further details on the nature and magnitude of the added noise and the sources that first used these synthetic data sets in the literature on free knot spline approximation. Each of these functions was used to generate five synthetic data sets of different sizes. In combination with varying numbers of free knots we obtained 20 problem instances for each of the 12 functions, yielding 240 problem instances in total. Table 8.5 shows all the combinations of data set sizes and number of free knots that were considered.

All the results of the experiments are listed in Tables A.3 to A.14 in the appendix. Concerning the CPU time required by the approaches, the B&B algorithm was faster than the MIQCP approach on all but three problem instances. Of the 240 problem instances, the B&B approach was able to solve 218 (90.8%) within the specified time limit of 3600 seconds. In contrast, the MIQCP approach was only able to solve 141 (58.8%) of the problem instances in the given time limit. Moreover, there exists no problem instance in our testbed that could be solved with the MIQCP but not with the B&B approach.

In particular when the size of the data set is large, the MIQCP approach frequently fails to compute the optimal solution within the time limit. Among 24 problem instances with data sets containing 800 points, only one could be solved to optimality with the MIQCP approach, whereas the B&B algorithm computed the optimal solution for 23 of these problem instances. These findings support the preliminary observations made in the previous section that the computing time required by the B&B algorithm grows slower in the size of the data set than the computing time of the MIQCP approach.

In order to further investigate the dependency of the computing time on the size of the data set, we consider problem instances with three free

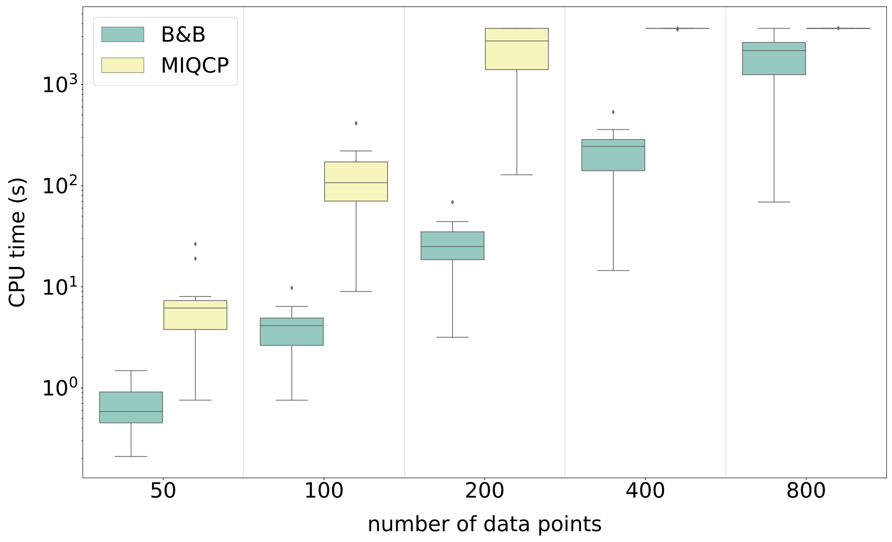


Figure 8.3: CPU times (in seconds) required by the B&B and MIQCP approaches in order to compute the optimal spline functions with three free knots. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

knots and compare the CPU times required by each method for different data set sizes. Figure 8.3 depicts the CPU times as box plots on a log scale. From this figure on can observe that the required CPU times of the B&B and MIQCP approaches increase exponentially in the number of data points, but the growth constant is smaller for the B&B algorithm. For most problems, the CPU times needed by the B&B algorithm are an order of magnitude smaller than the CPU times required by the MIQCP approach. This can also be observed in Table 8.6, which shows the average CPU times and the percentages of solved problem instances. Note that unsolved problem instances enter the average CPU times with the maximal computing time of 3600 seconds. Thus, if there are many unsolved instances, the average CPU time shown in the table do not give an accurate value for the average CPU time in absence of any time limit, which should be expected to be much higher.

Another dependency worth investigating is the dependence of the computing time on the number of free knots. In order to visualize this, we fix the size of the data set to 50 and compare the CPU times required by each method for problem instances with different numbers of free knots. From the box plots in Figure 8.4 we conclude that the CPU time required by both

	B&B		MIQCP	
number of data points	time	solved	time	solved
50	0.71	100%	7.89	100%
100	4.06	100%	152.13	100%
200	27.18	100%	2425.97	66.7%
400	222.81	100%	3591.73	8.3%
800	1927.69	91.7%	3602.16	0%

Table 8.6: Percentage of solved instances and average CPU times (in seconds) required by the B&B and MIQCP approaches to compute the optimal spline functions with three free knots. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

methods grows exponentially in the number of free knots, and the growth constants seem to be roughly similar. This can also be observed in Table 8.7, where the average CPU time required by the MIQCP approach is consistently 8-10 times larger then the CPU time required by the B&B algorithm, as long as both methods solve all problem instances in the given time limit, which is the case for up to five knots.

In summery it can be said that while both approaches enable the computation of high-quality solutions to least-squares spline approximation problems with free knots of practically relevant sizes, the conclusion of the numerical experiments presented in this section is that the B&B algorithm is superior to the MIQCP approach in terms of computing time.

8.2.3 Computation of Piecewise Polynomials

In the previous sections, we focused on the computation of cubic spline functions with the maximum number of continuity restrictions, since this is arguably the most relevant case in practice. However, as me mentioned in the beginning, our approaches can also be used if there are fewer or no continuity restriction. In this section we want to compare the B&B and MIQCP approaches when it comes to computing the optimal solution of the least-squares piecewise polynomial approximation problem with free knots, i.e., when no continuity conditions have to be satisfied at the knots.

We again used each of the functions from Table A.2 to generate five synthetic data sets of different sizes, and in combination with varying numbers

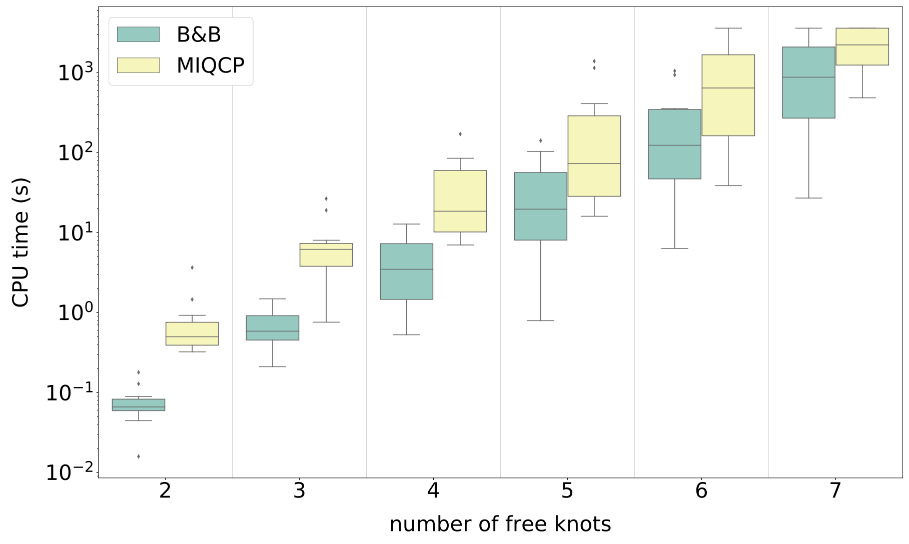


Figure 8.4: CPU times (in seconds) required by the B&B and MIQCP approaches in order to compute the optimal spline functions for 50 data points. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

of free knots, we obtained 240 problem instances in total. The combinations of data set sizes and numbers of free knots that we considered are shown in Table 8.8.

The results of the numerical experiments are listed in Tables A.15 to A.25 in the appendix. As was to be expected, the computing time required to compute an optimal piecewise polynomial function is considerably lower than the computing time needed to determine the optimal least-squares spline functions. This holds true for both approaches. On problem instances that could be solved in the piecewise polynomial and in the spline function approximation case, the B&B approach for the computation of the optimal piecewise polynomial needs on average only 18.9% of the CPU time required for the computation of the optimal spline function. Similarly, the MIQCP approach for the computation of the optimal piecewise polynomial needs on average only 15.9% of the CPU time required for the computation of the optimal spline function.

Concerning the relative performance of the B&B and MIQCP approaches, we observe that of the 240 problem instances, 204 (85%) could be solved within the given time limit of 3600 seconds with the B&B approach, whereas only 149 (62.1%) could be solved with the MIQCP approach. Of the 141

	B&B		MIQCP	
number of free knots	time	solved	time	solved
2	0.08	100%	0.85	100%
3	0.71	100%	7.88	100%
4	4.79	100%	42.08	100%
5	37.76	100%	307.34	100%
6	283.99	100%	1222.19	75.0%
7	1287.84	83.3%	2265.58	58.3%

Table 8.7: Percentage of solved instances and average CPU times (in seconds) required by the B&B and MIQCP approaches to compute the optimal spline functions for 50 data points. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

number of data points	number of free knots
100	$\{2, 3, 4, 5, 6, 7\}$
200	$\{2, 3, 4, 5, 6\}$
400	$\{2, 3, 4, 5\}$
800	$\{2, 3, 4\}$
1600	$\{2, 3\}$

Table 8.8: All the combinations of data set sizes and number of free knots that were considered when generating instances of the piecewise polynomial approximation problem with the functions in Table A.2.

problems that could be solved with both approaches, the B&B algorithm was faster for 129 problems and the average CPU time required by the MIQCP approach was more than three times as large as the average CPU time required by the B&B algorithm.

However, the relative performance of the approaches is highly dependent on the number of data points and knots. Of the 24 problem instances with data sets containing 1600 points, only one could be solved with the MIQCP approach, whereas the B&B approach was able to solve all of the 24 instances. On the other hand, of the 12 problems instances with 7 free knots, the MIQCP approach is able to solve 9, whereas the B&B approach can only solve 5 in the given time limit.

We illustrate the dependency of the CPU time on the number of data points in Figure 8.5, which depicts the CPU times required by each method for problem instances with three free knots and different data set sizes as box plots on a log scale. We observe that the CPU times required by the B&B algorithm are two orders of magnitude smaller than the CPU times required by the MIQCP approach. This can also be seen in Table 8.9, where the average CPU times and the percentage of solved problem instances are listed.

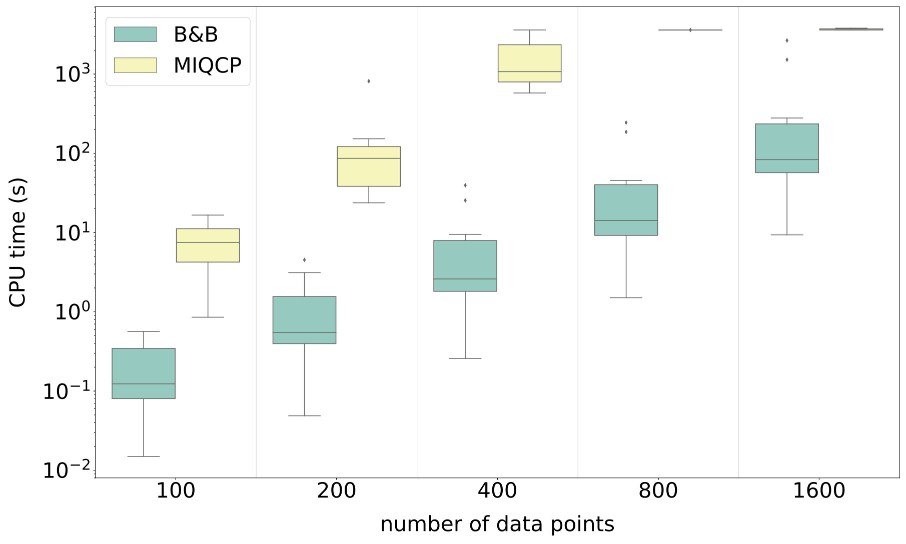


Figure 8.5: CPU times (in seconds) required by the B&B and MIQCP approaches in order to compute the optimal piecewise polynomials with three free knots. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

	B&B		MIQCP	
number of data points	time	solved	time	solved
100	0.20	100%	7.60	100%
200	1.19	100%	139.95	100%
400	8.08	100%	1629.47	83.3%
800	49.56	100%	3600.13	0%
1600	429.10	100%	3664.32	0%

Table 8.9: Percentage of solved instances and average CPU times (in seconds) required by the B&B and MIQCP approaches to compute the optimal piecewise polynomials with three free knots. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

We also illustrate the dependency of the CPU time on the number of free knots. For this purpose, we consider the problem instances with 100 data points and consider the CPU time required by each method for different numbers of free knots. From the box plots in Figure 8.6 and the corresponding Table 8.10 we observe that although the MIQCP approach is slower for most problem instances, the computing time grows slower in the number of free knots than the computing time of the B&B approach. In fact, for seven free knots, the average CPU time required by the MIQCP approach is smaller than the average CPU time of the B&B approach. Therefore, the MIQCP approach might be the more appropriate choice for problems with seven or more free knots.

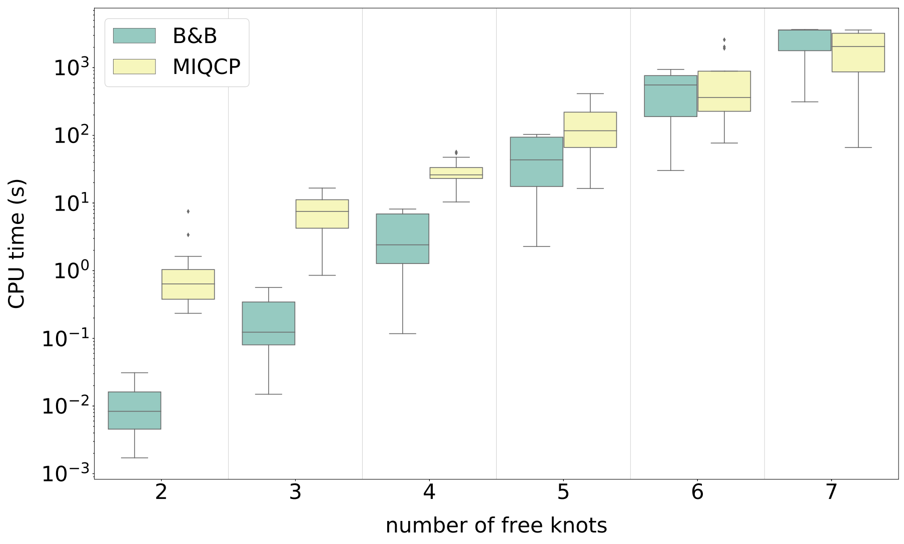


Figure 8.6: CPU times (in seconds) required by the B&B and MIQCP approaches in order to compute the optimal piecewise polynomials for 100 data points. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

	B&B		MIQCP	
number of free knots	time	solved	time	solved
2	0.01	100%	1.43	100%
3	0.20	100%	7.60	100%
4	3.63	100%	30.03	100%
5	50.48	100%	159.45	100%
6	492.54	100%	755.92	100%
7	2683.70	41.7 %	1967.34	75.0%

Table 8.10: Percentage of solved instances and average CPU times (in seconds) required by the B&B and MIQCP approaches to compute optimal piecewise polynomials for 100 data points. There are 12 synthetic data sets for each data set size and the time limit for each algorithm was set to 3600 seconds.

Chapter 9

Conclusions and Outlook

In this thesis we proposed new algorithms for the solution of nonconvex data approximation problems.

In Part I we proposed an adaptively sampled trust-region method for finite-sum minimization. We showed theoretically that the sample size is eventually increased to the size of the whole training data set, which implies global convergence. Our numerical experiments demonstrated strong performance of our method, which did not involve any hyper-parameter tuning to individual problems or data sets.

A promising avenue for future research is the incorporation of different kinds of curvature information. Limited-memory techniques could be used to make our method applicable to high-dimensional problems, see Burdakov et al. (2017) and Erway et al. (2019). Approximations to the diagonal of the Hessian, as described in Gower et al. (2018), could make the method more efficient for very large data sets and the training of more complex neural networks.

In Part II of this thesis, we proposed to solve a combinatorial formulation of the least-squares spline approximation problem with free knots. We demonstrated that the optimal solution of this combinatorial problem is close to the globally optimal solution and can therefore serve as an initial solution for a local optimization algorithm. We developed two approaches for the solution of the combinatorial optimization problem, namely an MIQCP formulation and a tailored B&B method. Our numerical experiments confirmed that both approaches enable the solution of problems of practically relevant sizes, and that the B&B algorithm is superior in terms of the required CPU time.

There are several possible directions for future research. One promising idea is to accelerate the solution of the linear systems of equations in the B&B approach by switching to the B-Spline basis. On the one hand, the linear systems that have to be solved in order to determine lower bounds for the subproblems become banded and therefore the method of Cholesky for positive band matrices could be used, see Martin et al. (1965). On the other hand, and going even further, one could make use of the fact that the restriction sets $\bar{\Lambda}_0, \ldots, \bar{\Lambda}_k$ obtained with Algorithm (7.1) are oftentimes very similar to the sets $\Lambda_0, \ldots, \Lambda_k$ that were provided as input to the algorithm. In many cases, only a single index is added to one of the restriction sets. Thus, in order to compute a lower bound for the subproblem $S(\bar{\Lambda}_0, \ldots, \bar{\Lambda}_k)$ one typically has to solve a linear system of equations that is quite similar to the linear system that was solved in a prior iteration when a lower bound of the subproblem $S(\Lambda_0, \ldots, \Lambda_k)$ was computed. Thus, it should be expected that the efficiency of the B&B algorithm would improve significantly by employing the orthogonalization method of Cox (1981), saving the QR decomposition for each subproblem and solving the modified linear systems with Givens rotations, see, e.g., Gentleman (1973).

Finally, it would be interesting to compare the B&B and mixed-integer approaches for spline approximation problems with other error functions. As discussed in Section 6.3, the absolute and maximum error functions lead to MILPs, which might be easier to solve than the convex MIQCPs that have to be solved when the least-squares error function is used. In contrast, the iterations of the B&B algorithm likely become more expensive, since an LP has to be solved in every iteration, instead of a system of linear equations. The question of whether this is sufficient to make the mixed-integer approach superior to the B&B algorithm in these cases is left for future research.

Bibliography

P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

G. Beliakov. Least squares splines with free knots: global optimization approach. *Applied Mathematics and Computation*, 149(3):783–798, 2004.

S. Bellavia, S. Gratton, and E. Riccietti. A levenberg–marquardt method for large nonlinear least-squares problems with dynamic accuracy in functions and gradients. *Numerische Mathematik*, 140(3):791–825, 2018.

A. S. Berahas and M. Takáč. A robust multi-batch l-bfgs method for machine learning. *Optimization Methods and Software*, 15:1–29, 2019.

A. S. Berahas, J. Nocedal, and M. Takáč. A multi-batch l-bfgs method for machine learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 1063–1071, 2016.

D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1st edition, 1997.

J. Blanchet, C. Cartis, M. Menickelly, and K. Scheinberg. Convergence rate analysis of a stochastic trust-region method via supermartingales. *INFORMS Journal on Optimization*, 1(2):92–119, 2019.

R. Bollapragada, R. Byrd, and J. Nocedal. Adaptive sampling strategies for stochastic optimization. *SIAM Journal on Optimization*, 28(4):3312–3343, 2018a.

R. Bollapragada, R. H. Byrd, and J. Nocedal. Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39 (2):545–578, 2018b.

R. Bollapragada, D. Mudigere, J. Nocedal, H.-J. M. Shi, and P. T. P. Tang. A progressive batching l-bfgs method for machine learning. *arXiv preprint arXiv:1802.05374*, 2018c.

A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

M. J. Brusco and S. Stahl. *Branch-and-Bound Applications in Combinatorial Data Analysis.* Springer, New York, 2005.

O. Burdakov, L. Gong, S. Zikrin, and Y.-X. Yuan. On efficiently combining limited-memory and trust-region techniques. *Mathematical Programming Computation*, 9(1):101–134, 2017.

R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal. On the use of stochastic hessian information in unconstrained optimization. *SIAM Journal on Optimization*, 21(3):977–995, 2011.

R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1): 127–155, 2012.

R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2): 1008–1031, 2016.

C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, 169(2):337–375, 2018.

C. Cartis, N. I. M. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011a.

C. Cartis, N. I. M. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function- and derivative-evaluation complexity. *Mathematical Programming*, 130(2):295–319, 2011b.

C.-C. Chang and C.-J. Lin. Libsvm. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

R. Chen, M. Menickelly, and K. Scheinberg. Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169(2):447–487, 2018.

R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of svms for very large scale problems. *Advances in Neural Information Processing Systems*, 633–640. MIT Press, 2002.

A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-region methods*, volume 1 of *MPS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

M. G. Cox. The least squares solution of overdetermined linear equations having band or augmented band structure. *IMA Journal of Numerical Analysis*, 1 (1):3–22, 1981.

F. Curtis. A self-correcting variable-metric algorithm for stochastic optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, 632–641, 2016.

F. E. Curtis and D. P. Robinson. Exploiting negative curvature in deterministic and stochastic optimization. *Mathematical Programming*, 176(1-2):69–94, 2019.

F. E. Curtis and K. Scheinberg. Optimization methods for supervised machine learning: From linear models to deep learning. In INFORMS Tutorials in Operations Research, 89–114, INFORMS, 2017.

Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, 2933–2941, 2014.

S. De, A. Yadav, D. Jacobs, and T. Goldstein. Automated inference with adaptive batches. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 1504–1513, 2017.

C. de Boor and J. R. Rice. Least squares cubic spline approximation, ii - variable knots. *Purdue University Report*, (April 1968):No. CSD TR 21, 1968.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems*, 2014.

D. G. T. Denison, B. K. Mallick, and A. F. M. Smith. Automatic bayesian curve fitting. *Journal of the Royal Statistical Society, Series B (Methodological)*, 60 (3):333–350, 1998.

P. Dierckx. *Curve and surface fitting with splines.* Monographs on Numerical Analysis, Clarendon Press, Oxford, 1996.

I. DiMatteo, C. R. Genovese, and R. E. Kass. Bayesian curve-fitting with free-knot splines. *Biometrika*, 88(4):1055–1071, 2001.

J. B. Erway, J. Griffin, R. F. Marcia, and R. Omheni. Trust-region algorithms for training responses: machine learning methods using indefinite hessian approximations. *Optimization Methods and Software*, 13:1–28, 2019.

M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

J. H. Friedman and B. W. Silverman. Flexible parsimonious smoothing and additive modeling. *Technometrics*, 31(1):3–21, 1989.

A. R. Gallant and W. A. Fuller. Fitting segmented polynomial regression models whose join points have to be estimated. *Journal of the American Statistical Association*, 68(341):144–147, 1973.

W. M. Gentleman. Least squares computations by givens transformations without square roots. *IMA Journal of Applied Mathematics*, 12(3):329–336, 1973.

N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint. Solving the trust-region subproblem using the lanczos method. *SIAM Journal on Optimization*, 9(2): 504–525, 1999.

R. Gower, D. Goldfarb, and P. Richtarik. Stochastic block bfgs: Squeezing more curvature out of data. In *Proceedings of The 33rd International Conference on Machine Learning*, 1869–1878, 2016.

R. Gower, N. Le Roux, and F. Bach. Tracking the gradients using the hessian: A new look at variance reducing stochastic methods. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, 707–715, 2018.

Gratton. Complexity and global rates of trust-region methods based on probabilistic models. *Preprint 17-09, Department of Mathematics, University of Coimbra*, 2017.

A. Griewank. The modification of newton's method for unconstrained optimization by bounding cubic terms. *Technical Report NA/12. Departement of Applied Mathematics and Theoretical Physics, University of Cambridge*, 1981.

D. J. Hand. Branch and bound in statistical data analysis. *Journal of the Royal Statistical Society, Series D (The Statistician)*, 30(1):1–13, 1981.

Y. Hu. An algorithm for data reduction using splines with free knots. *IMA Journal of Numerical Analysis*, 13:365–381, 1993.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 315–323, 2013.

D. L. B. Jupp. Approximation to data by splines with free knots. *SIAM Journal on Numerical Analysis*, 15(2):328–343, 1978.

J. M. Kohler and A. Lucchi. Sub-sampled cubic regularization for non-convex optimization. *Proceedings of the 34th International Conference on Machine Learning*, 1895-1904, 2017.

H. Lanshammar. On practical evaluation of differentiation techniques for human gait analysis. *Journal of Biomechanics*, 15(2):99–105, 1982.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

T. C. Lee. On algorithms for ordinary least squares regression spline fitting: A comparative study. *Journal of Statistical Computation and Simulation*, 72 (8):647–663, 2002.

M. J. Lindstrom. Penalized estimation of free knot splines. *Journal of Computational and Graphical Statistics*, 8(2):333–352, 1999.

E. Mammen and S. van de Geer. Locally adaptive regression splines. *The Annals of Statistics*, 25(1):387–413, 1997.

J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning*, 735-742, 2010.

J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. *In Proceedings of the 28th International Conference on Machine Learning*, 1033–1040, 2011.

R. S. Martin, G. Peters, and J. H. Wilkinson. Symmetric decomposition of a positive definite matrix. *Numerische Mathematik*, 7(5):362–383, 1965.

S. Miyata and X. Shen. Adaptive free-knot splines. *Journal of Computational and Graphical Statistics*, 12(1):197–213, 2003.

R. Mohr and O. Stein. An adaptive sample size trust-region method for finite-sum minimization. *arXiv preprint arXiv:1910.03294*, 2019.

A. Mokhtari, Alej, and R. Ribeiro. Global convergence of online limited memory bfgs. *Journal of Machine Learning Research*, 16(98):3151–3181, 2015.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley-Interscience, New York, 1988.

Y. Nesterov and B. T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, second edition, 2006.

C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Upper Saddle River, 1982.

B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994.

J. C. Pezzack, R. W. Norman, and D. A. Winter. An assessment of derivative determining techniques used for motion analysis. *Journal of Biomechanics*, 10(5-6):377–382, 1977.

J. Pittman and C. Murthy. Fitting optimal piecewise linear functions using genetic algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):701–718, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, MIT Press, Cambridge, 1998.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

F. Roosta-Khorasani and M. W. Mahoney. Sub-sampled newton methods. *Mathematical Programming*, 174(1-2):293–326, 2019.

D. Ruppert. Selecting the number of knots for penalized splines. *Journal of Computational and Graphical Statistics*, 11(4):735–757, 2002.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, 2017.

N. N. Schraudolph, Yu Jin, and S. Günter. A stochastic quasi-newton method for online convex optimization. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 436–443, 2007.

H. Schwetlick and T. Schütze. Least squares approximation by splines with free knots. *BIT Numerical Mathematics*, 35(3):361–384, 1995.

P.L. Smith. Curve fitting and modeling with splines using statistical variable selection techniques. Report NASA 166034, NASA Langley Research Center, Hampton, 1982.

J. Sohl-Dickstein, B. Poole, and S. Ganguli. Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. In *Proceedings of the 31st International Conference on Machine Learning*, 604–612, 2014.

S. Spiriti, R. Eubank, P. W. Smith, and D. Young. Knot selection for least-squares and penalized splines. *Journal of Statistical Computation and Simulation*, 83 (6):1020–1036, 2013.

S. Spiriti, P. Smith, and P. Lecuyer. freeknotsplines: Algorithms for implementing free-knot splines, 2018. URL `https://CRAN.R-project.org/package=freeknotsplines`.

T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

O. Stein. *Grundzüge der Globalen Optimierung.* Lehrbuch. Springer Spektrum, Berlin, 2018.

C. J. Stone, M. H. Hansen, C. Kooperberg, and Y. K. Truong. Polynomial splines and their tensor products in extended linear modeling. *The Annals of Statistics*, 25(4):1371–1425, 1997.

P. L. Toint. Towards an efficient sparsity exploiting newton method for minimization. In *Sparse Matrices and Their Uses*, 57–88, Academic press, 1981.

M. P. Wand. A comparison of regression spline smoothing procedures. *Computational Statistics*, 15:443–462, 2000.

S. Wold. Spline functions in data analysis. *Technometrics*, 16(1):1–11, 1974.

P. Xu, F. Roosta-Khorasani, and M. W. Mahoney. Second-order optimization for non-convex machine learning: An empirical study. *arXiv preprint arXiv:1708.07827*, 2017.

P. Xu, F. Roosta-Khorasani, and M. W. Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 24(3):1238, 2019.

Z. Yao, P. Xu, F. Roosta-Khorasani, and M. W. Mahoney. Inexact non-convex newton-type methods. *arXiv preprint arXiv:1802.06925*, 2018.

Y.-X. Yuan. Recent advances in trust region algorithms. *Mathematical Programming*, 151(1):249–281, 2015.

C. Zhou, W. Gao, and D. Goldfarb. Stochastic adaptive quasi-newton methods for minimizing expected values. In *Proceedings of the 34th International Conference on Machine Learning*, 4150–4159, 2017.

S. Zhou and X. Shen. Spatially adaptive regression splines and accurate knot selection schemes. *Journal of the American Statistical Association*, 96(453): 247–259, 2001.

Appendix A

Results of the Numerical Experiments

A.1 Large-Scale Finite-Sum Minimization

For each problem and data set, we plot the training error and the test accuracy against CPU-time and effective gradient evaluations. Moreover, the plots in the last row provide additional details concerning the ADST method. On the left, we report the behavior of the sample sizes used in the gradient and Hessian matrix approximations (s and s_H, respectively), on the right the number of inner iterations is depicted.

In Figures A.1 - A.8 we report the results for the logistic regression experiments on the 8 binary classification data sets from Table A.1. Figures A.9 - A.16 show the performance of the methods for the nonlinear least-squares problems on the same binary classification data sets.

name	# training/test points	# features	# classes	source
a9a	32,561/16,281	123	2	Platt (1998)
w8a	49,749/14,951	300	2	Platt (1998)
odd_even	60,000/10,000	784	2	Lecun et al. (1998)
ijcnn	127,522/14,169	22	2	Chang and Lin (2011)
skin	220,551/24,506	3	2	Chang and Lin (2011)
covertype	522,911/58,101	54	2	Collobert et al. (2002)
SUSY	4,500,000/500,000	18	2	Baldi et al. (2014)
HIGGS	9,900,000/1,100,000	28	2	Baldi et al. (2014)
MNIST	60,000/10,000	784	10	Lecun et al. (1998)

Table A.1: Data sets used in the numerical experiments. If no split of the data set into training and test set was provided in the source, we chose 10% of the data points randomly as our test set. Otherwise we kept the original split into training and test points, except for the data set *ijcnn*, where we shrank the size of the test set from 65% to 10% of the data points.

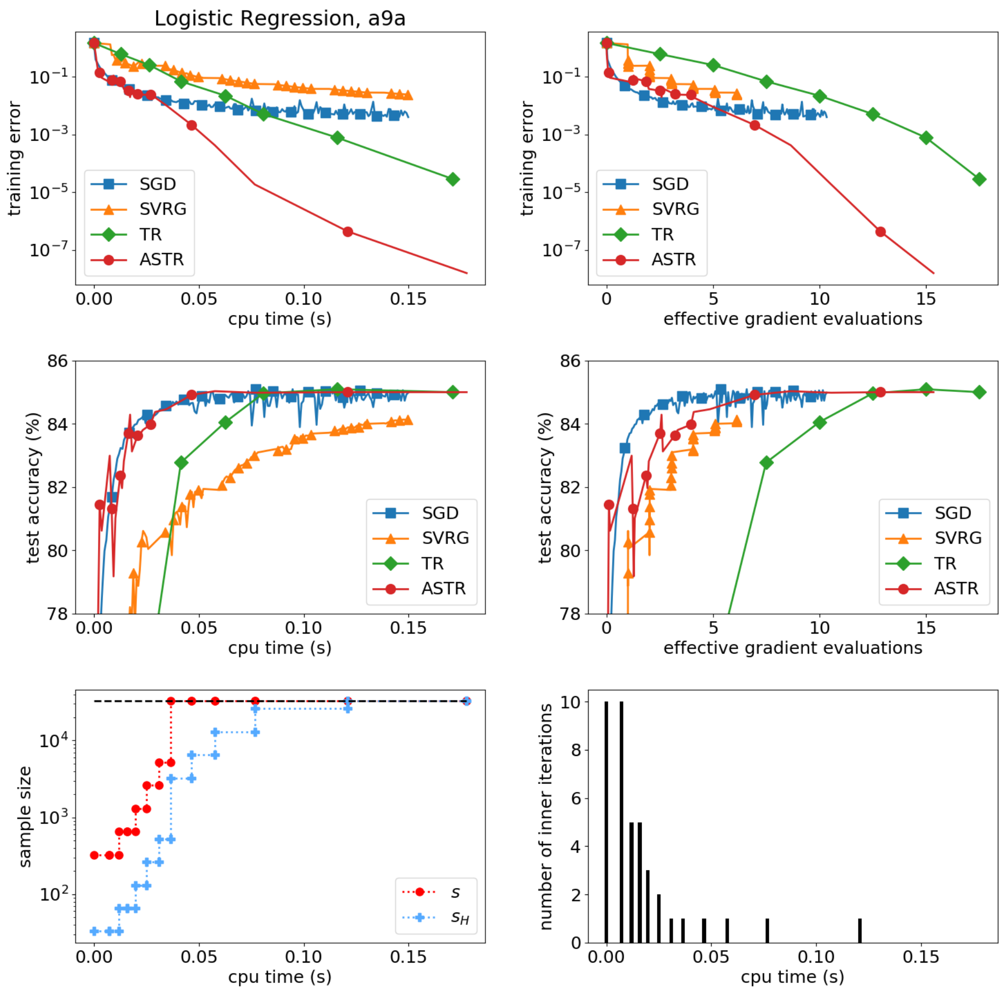


Figure A.1: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *a9a*.

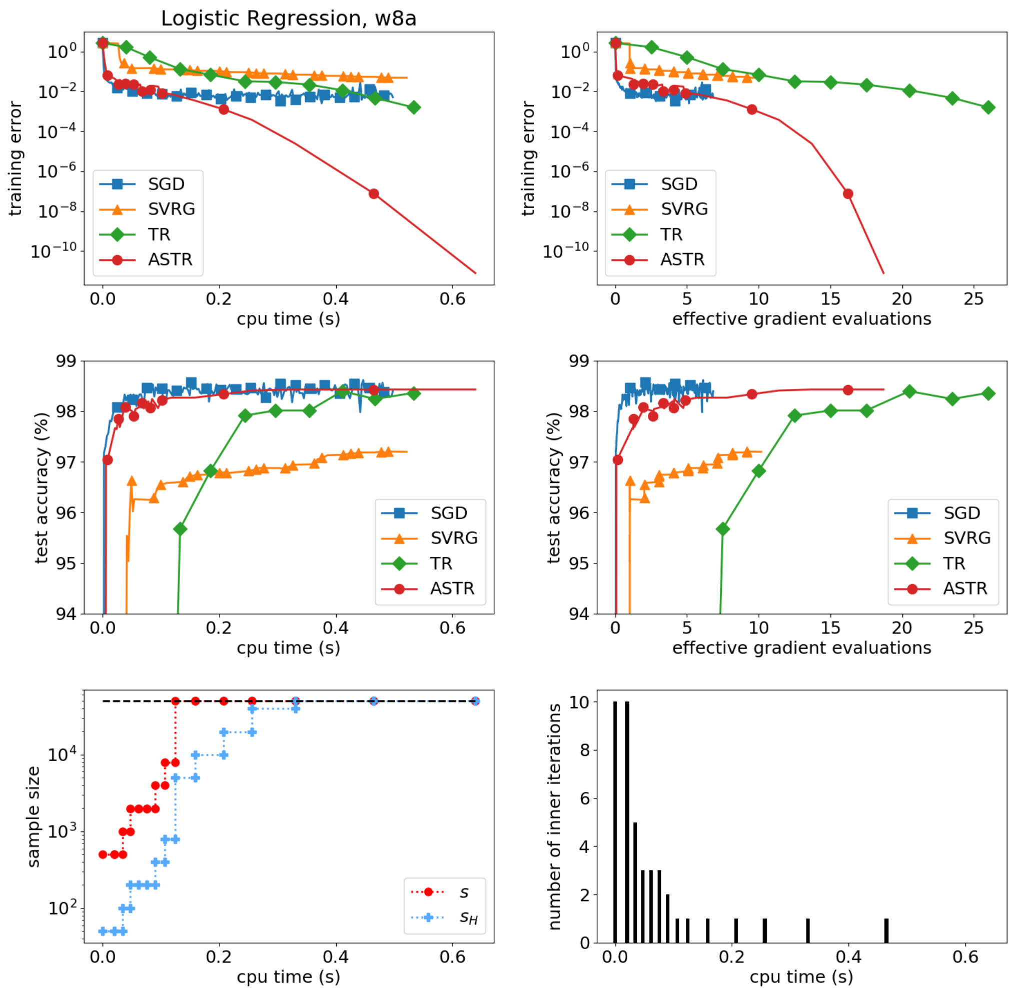


Figure A.2: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *w8a*.

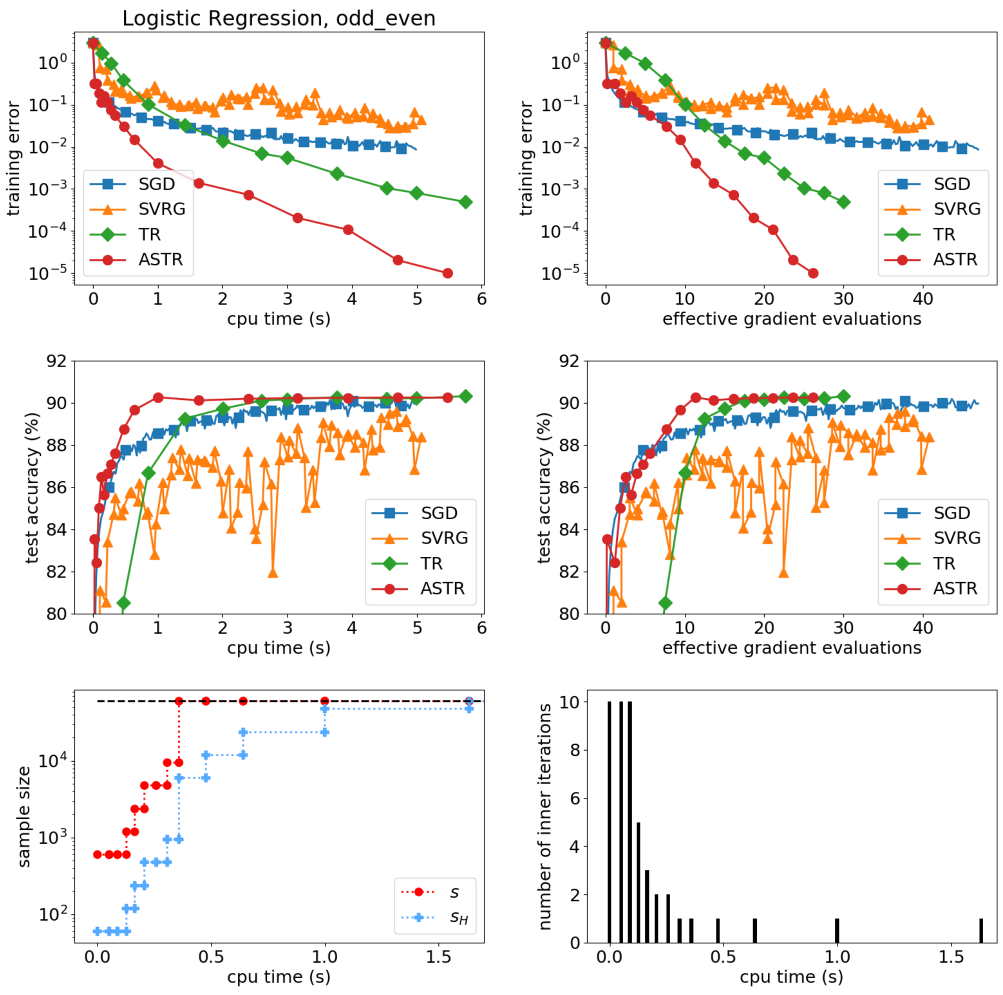


Figure A.3: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *odd_even*.

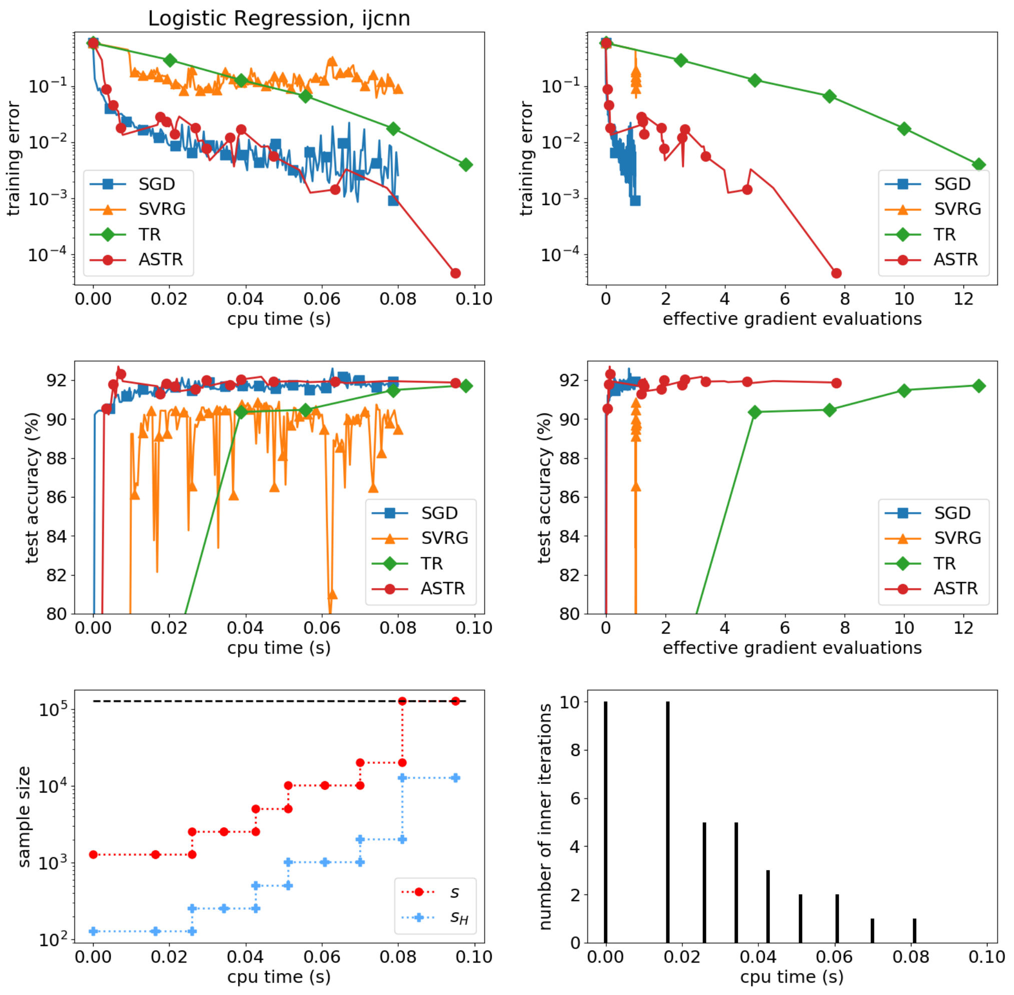


Figure A.4: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *ijcnn*.

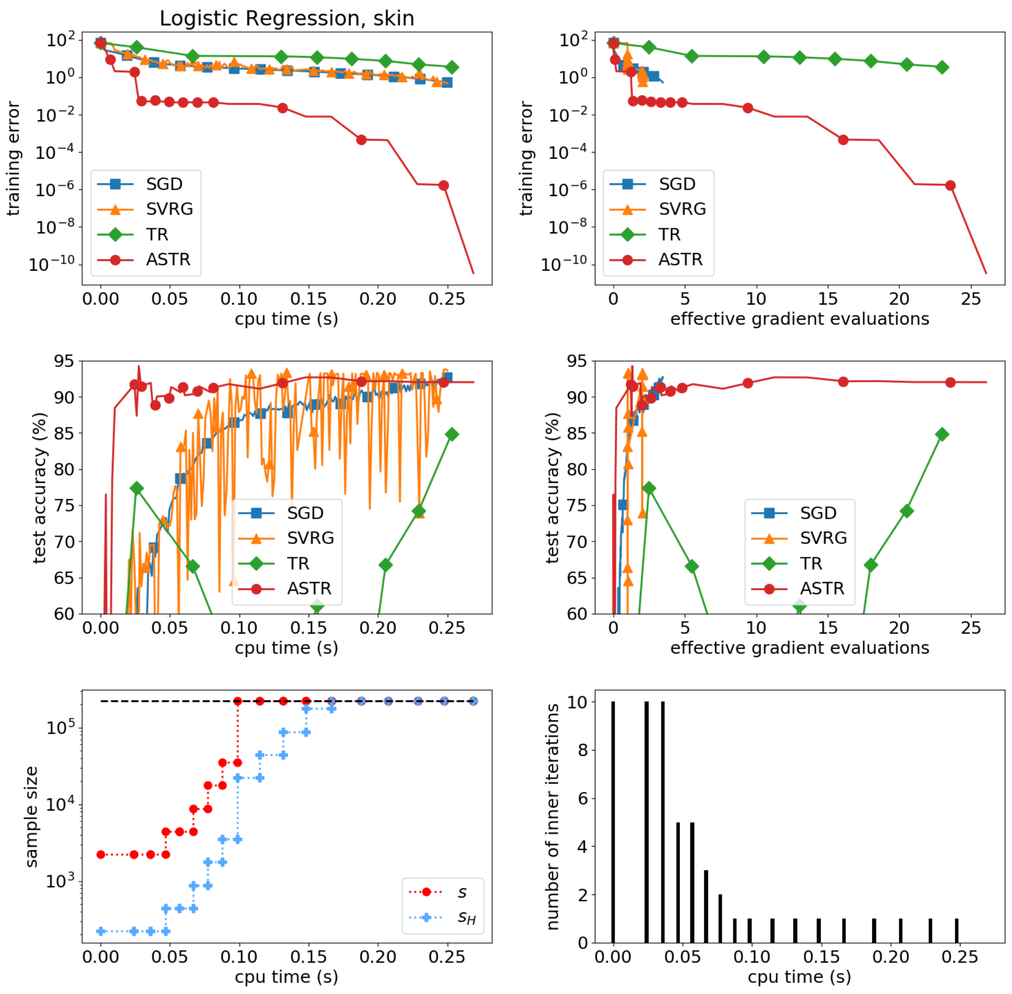


Figure A.5: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *skin*.

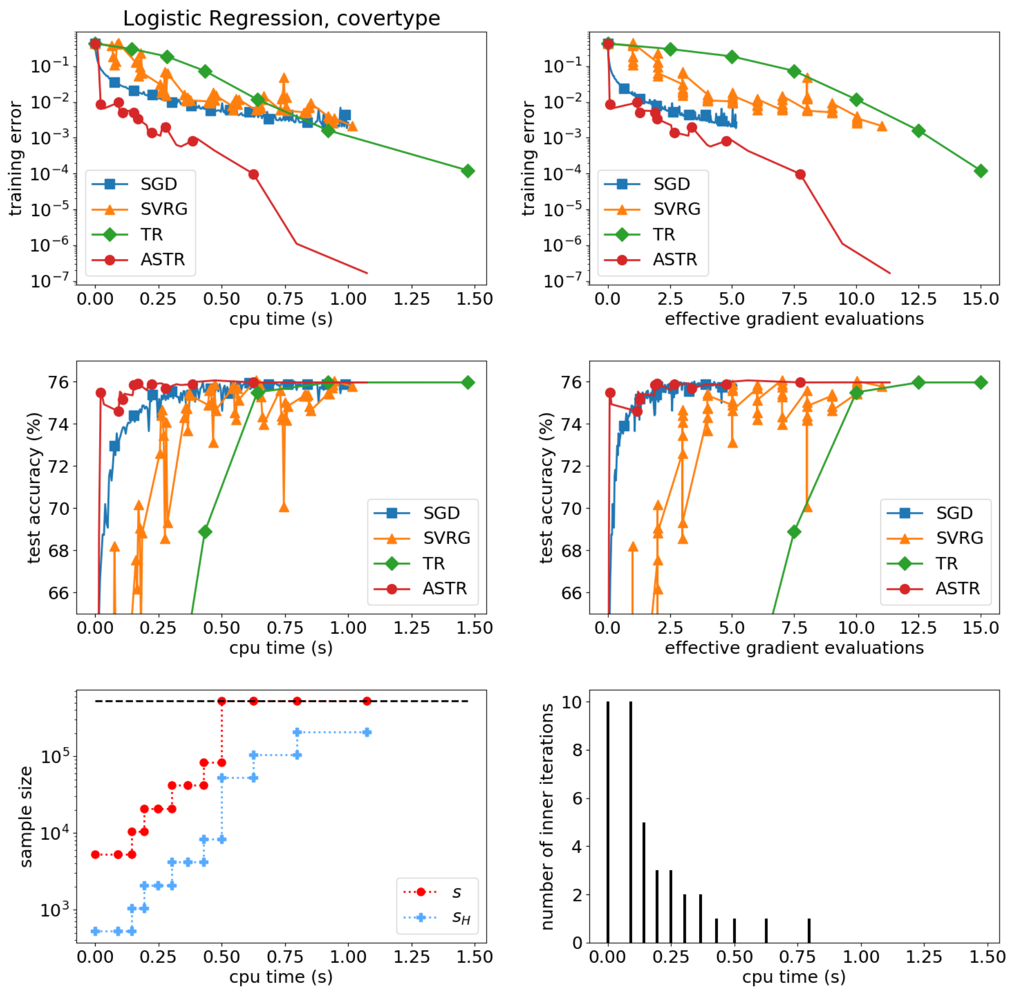


Figure A.6: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *covertype*.

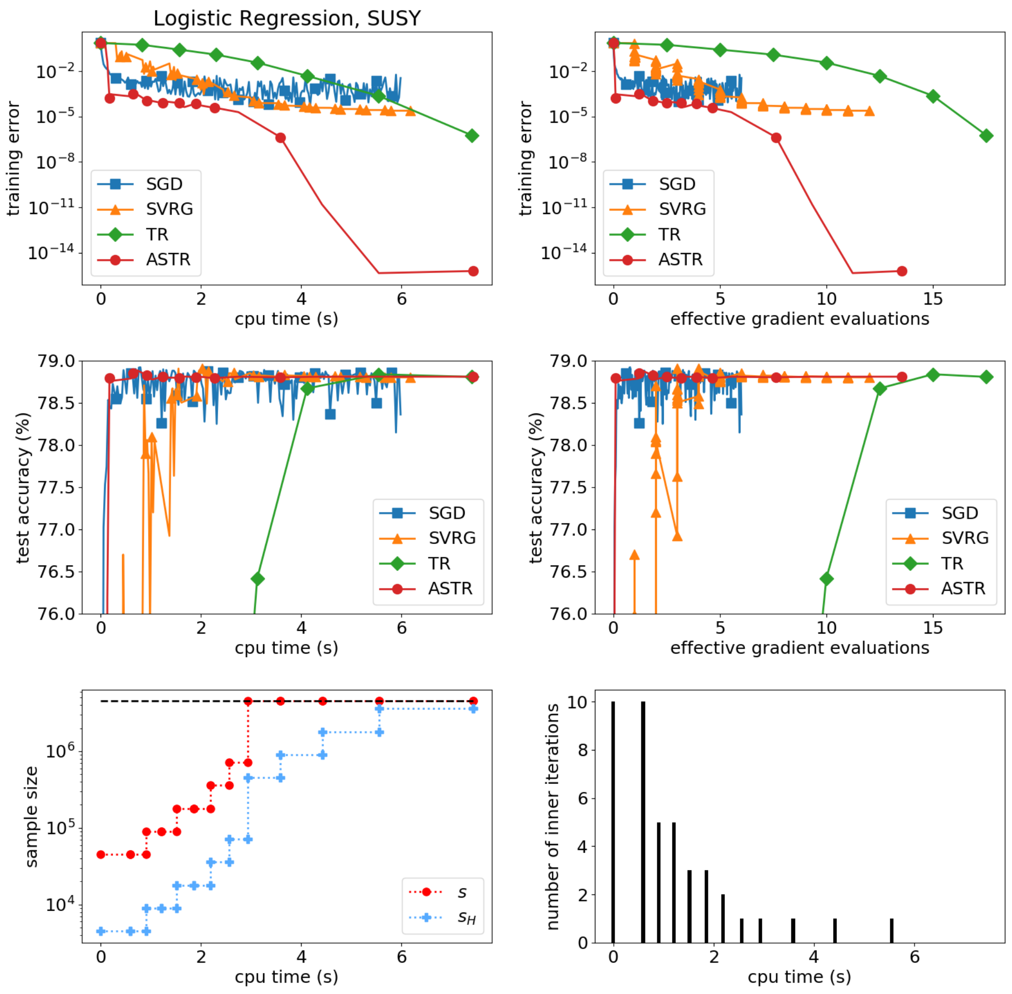


Figure A.7: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *SUSY*.

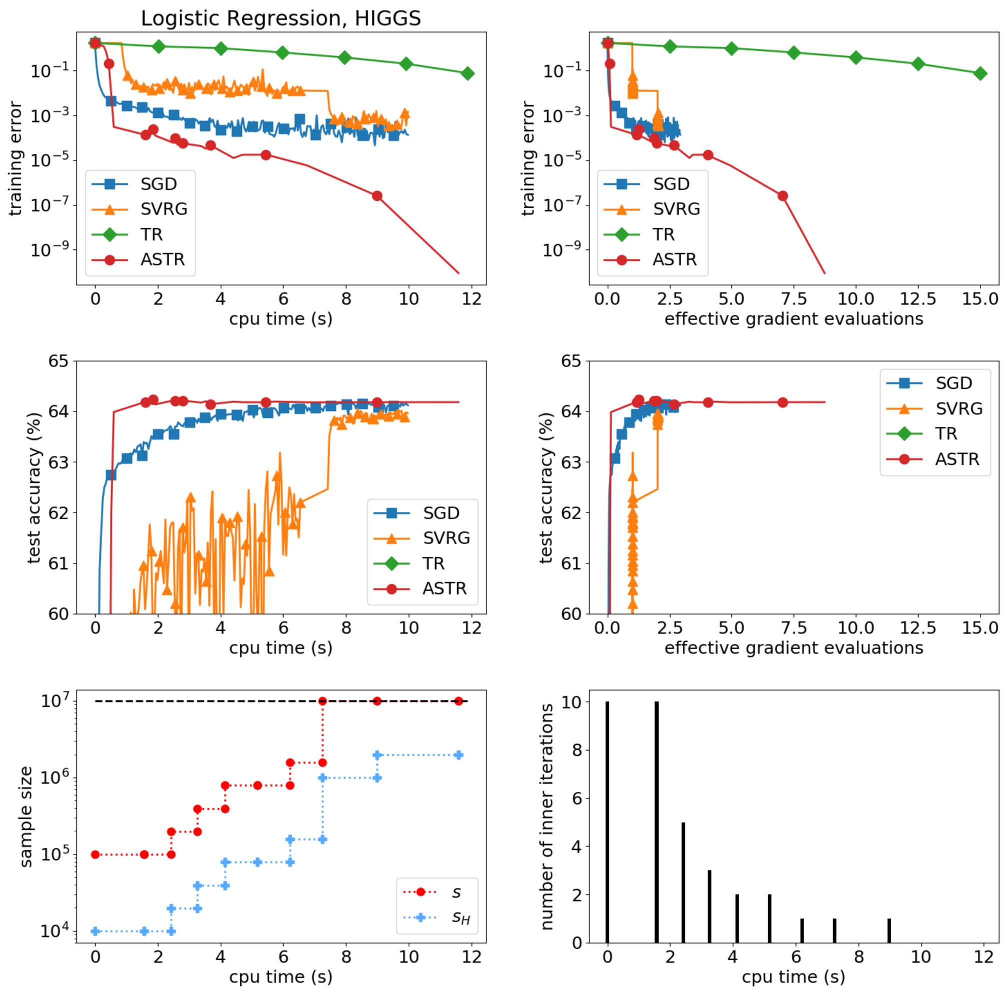


Figure A.8: Performance of SGD, SVRG, TR and ASTR on the logistic regression problem with data set *HIGGS*.

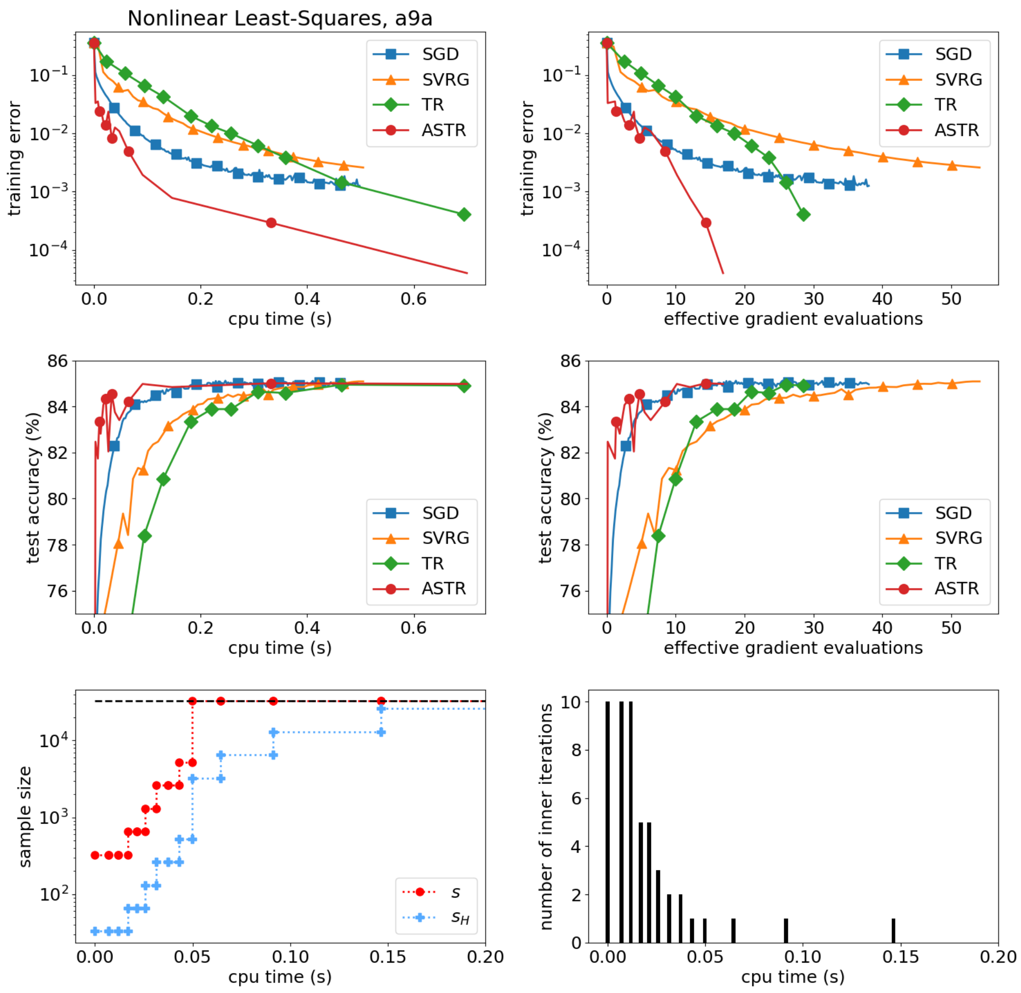


Figure A.9: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *a9a*.

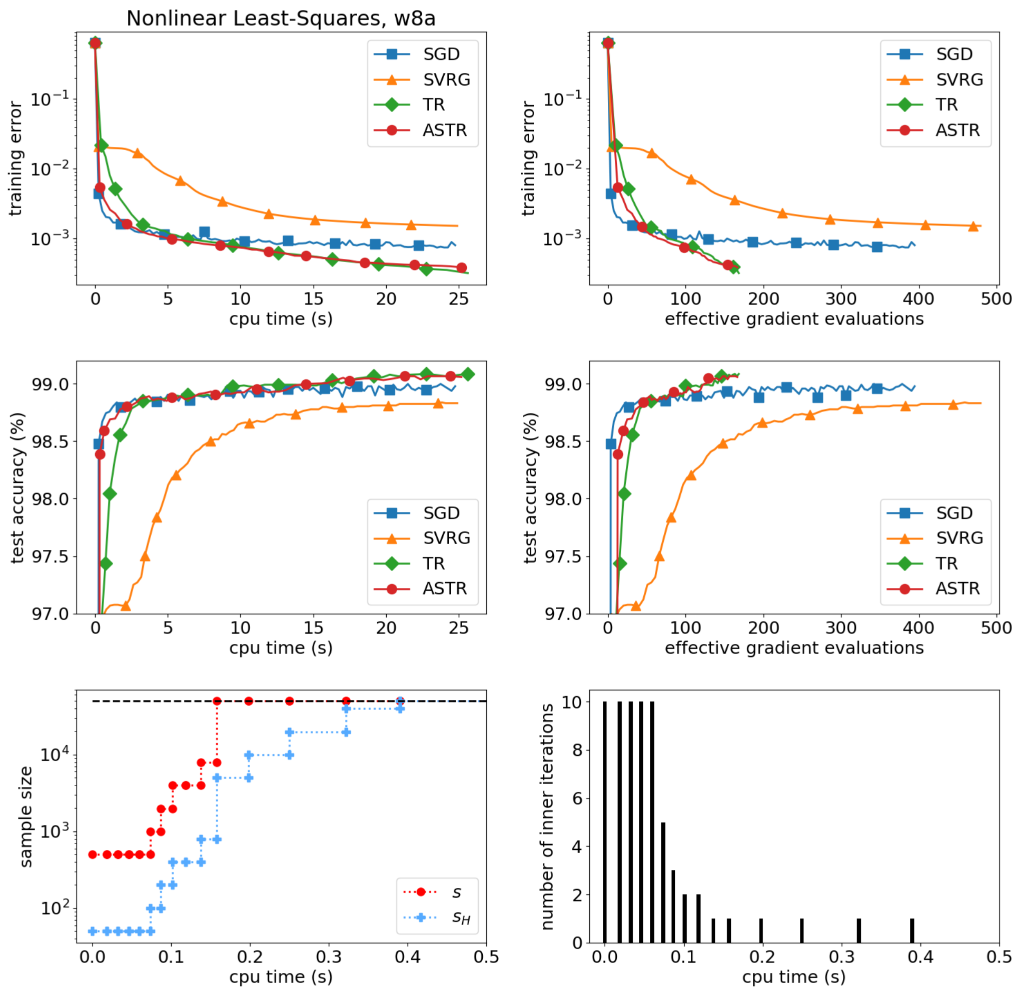


Figure A.10: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *w8a*.

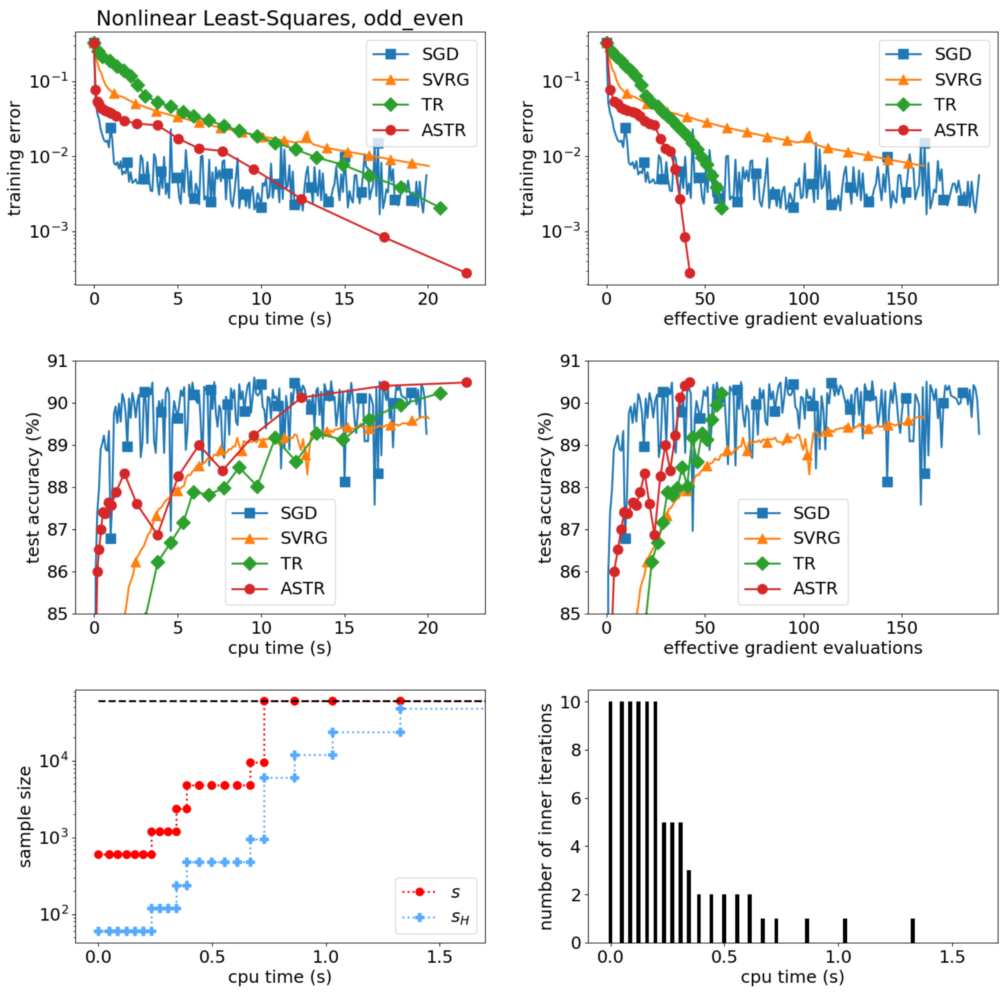


Figure A.11: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *odd_even*.

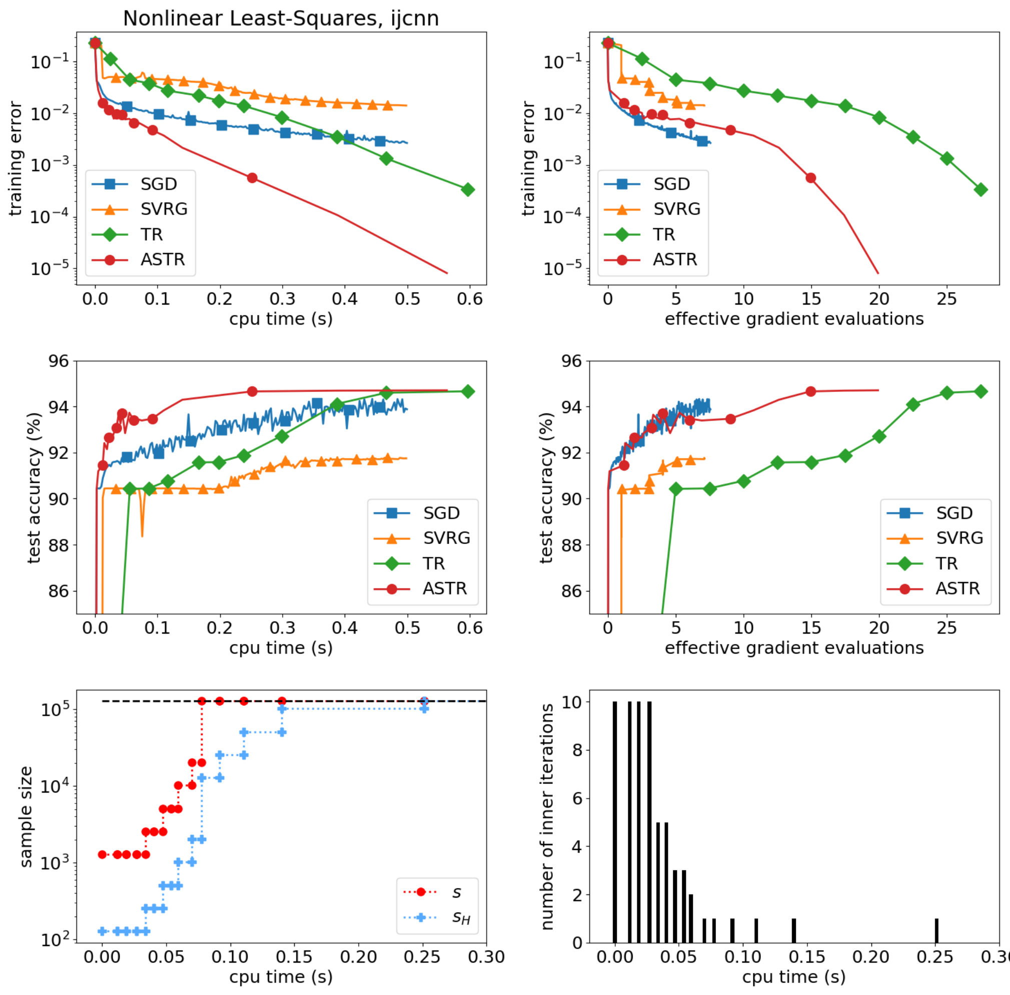


Figure A.12: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *ijcnn*.

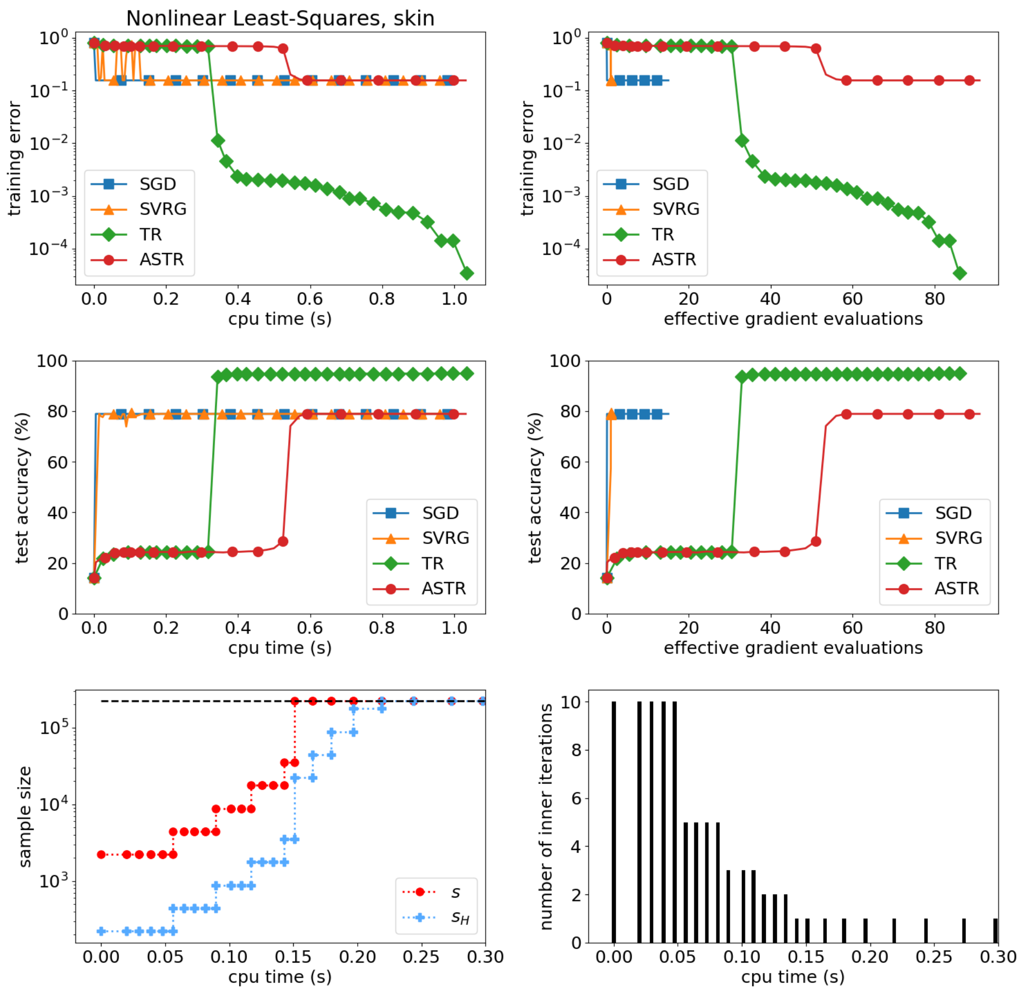


Figure A.13: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *skin*.

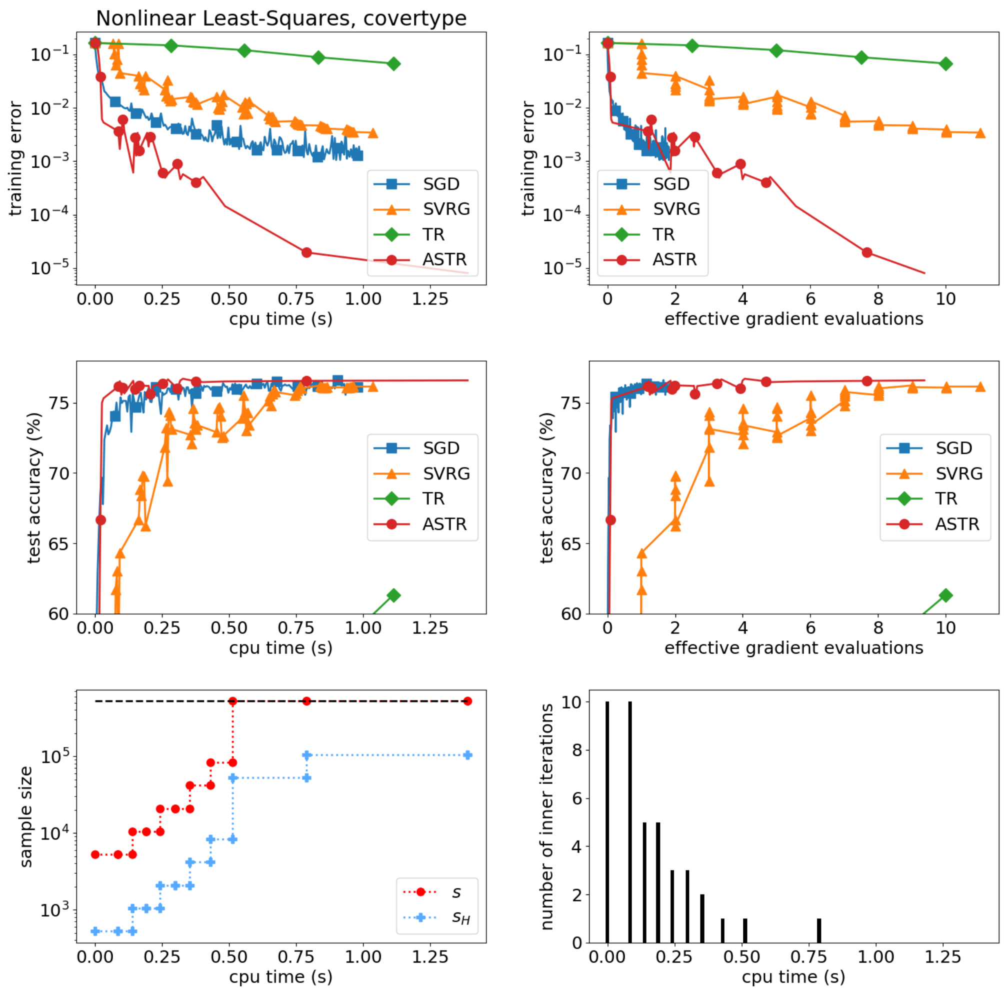


Figure A.14: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *covertype*.

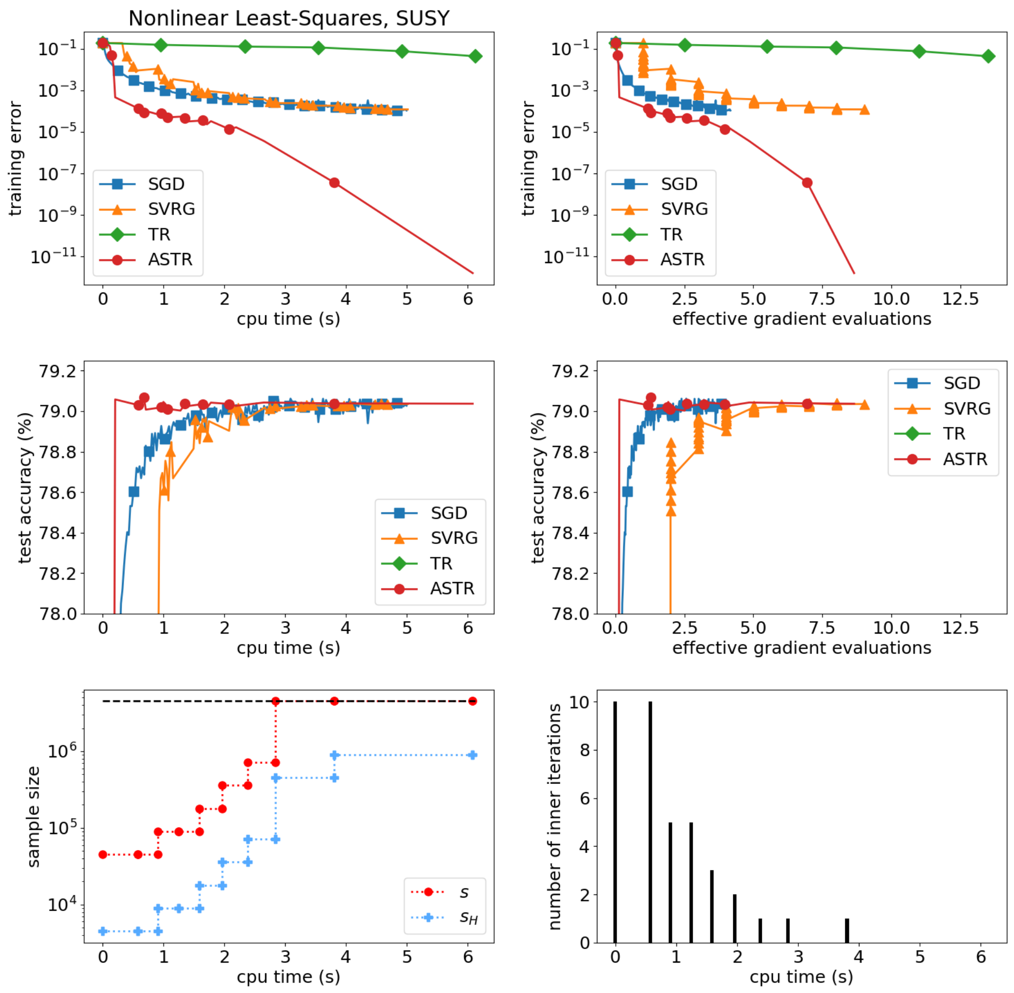


Figure A.15: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *SUSY*.

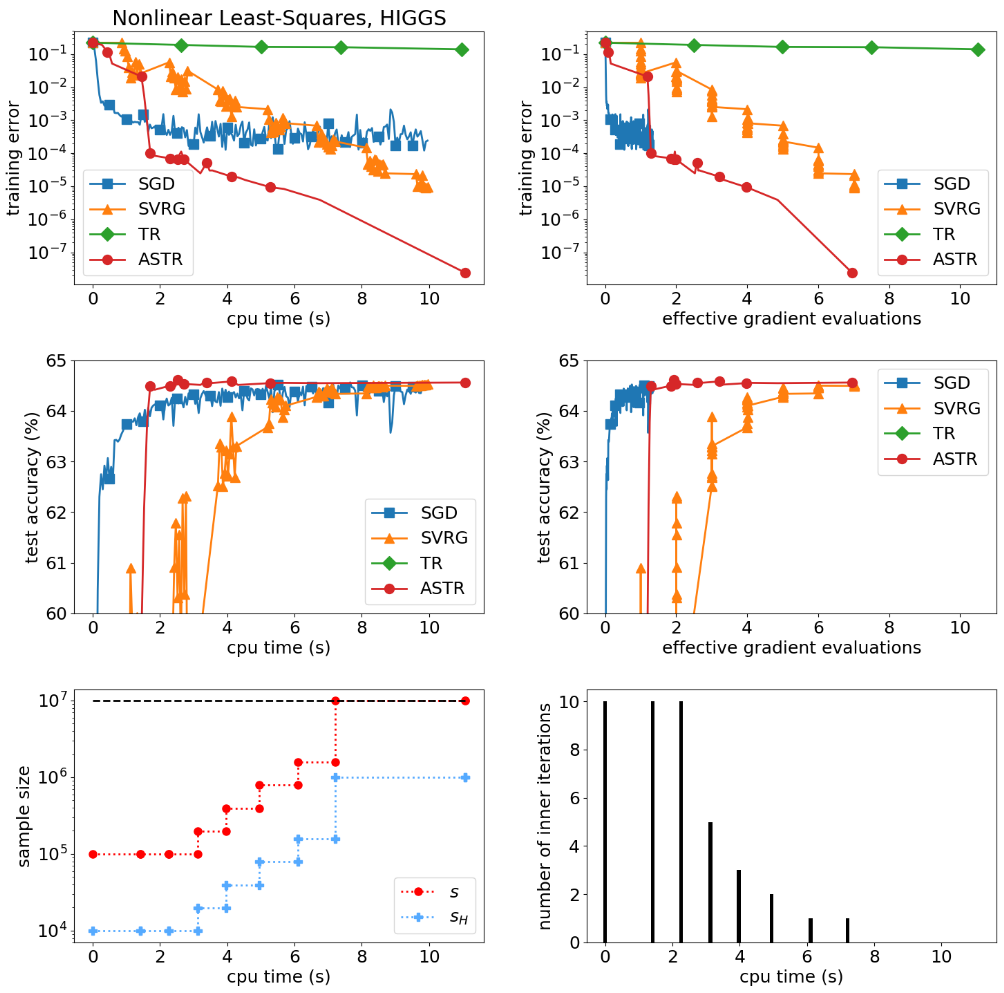


Figure A.16: Performance of SGD, SVRG, TR and ASTR on the nonlinear least squares problem with data set *HIGGS*.

A.2 Least-Squares Spline Approximation with Free Knots

The test functions that were used to generate the data sets are listed in Table A.2. Five data sets of different sizes were generated per test function, totaling 60 data sets. In combination with varying numbers of free knots (see Table A.2), we obtain 240 problem instances

The results of the application of the B&B and MIQCP approaches to the 240 problem instances are shown in Tables A.3 to A.14. Each table shows the CPU time and the minimal least-squares error that was obtained with the B&B and MIQCP approaches for each of the 20 problem instances corresponding to one of the test functions.

name	function y(x)	interval	noise	source
coslin	$\cos(6x) + x$	$[0, 2.5]$	$\mathcal{N}(0, 0.05^2)$	-
cube	$\sin(2\pi x^3)^3$	$[0, 1]$	$\mathcal{N}(0, 0.02^2)$	Lindstrom (1999)
arc_tan	$\arctan(10x)$	$[-10, 10]$	$\mathcal{U}(-0.075, 0.075)$	Beliakov (2004)
rational	$\frac{10x}{1+100x^2}$	$[-2, 2]$	$\mathcal{N}(0, 0.01^2)$	Hu (1993)
Inhom1	$\sin(\frac{4}{x}) + 1.5$	$[0, 1]$	$\mathcal{N}(0, 0.02^2)$	Mammen and van de Geer (1997)
Inhom2	$\sin(\frac{2}{0.2+x}) + 1.5$	$[0, 1]$	$\mathcal{N}(0, 0.01^2)$	Mammen and van de Geer (1997)
Inhom3	$\begin{cases} 2.55 - 4.5x,\ 0 \le x < 0.3 \\ -0.75 + 4.5x,\ 0.3 \le x < 0.7 \\ 5.55 - 4.5x,\ x \ge 0.7 \end{cases}$	$[0, 1]$	$\mathcal{N}(0, 0.02^2)$	Mammen and van de Geer (1997)
logit	$\frac{1}{1+\exp(-20(x-0.5))}$	$[0, 1]$	$\mathcal{N}(0, 0.05^2)$	Ruppert (2002)
bump	$x + 2\exp(-(16(x-0.5))^2)$	$[0, 1]$	$\mathcal{N}(0, 0.05^2)$	Ruppert (2002)
sine3	$\sin(6\pi x)$	$[0, 1]$	$\mathcal{N}(0, 0.05^2)$	Ruppert (2002)
sine6	$\sin(12\pi x)$	$[0, 1]$	$\mathcal{N}(0, 0.05^2)$	Ruppert (2002)
SpaHet3	$\sqrt{(x-x^2)} \cdot \frac{\sin(2\pi(1+2^{-\frac{3}{5}}))}{(x+2^{-\frac{3}{5}})}$	$[0, 1]$	$\mathcal{N}(0, 0.05^2)$	Ruppert (2002)

Table A.2: An overview of the functions used to generate the synthetic data sets used in the numerical experiments on least-squares spline and piecewise polynomial approximation problems. In addition, the nature and magnitude of the added noise and the sources that introduced these test functions in the context of free knot spline approximation are shown in the last two columns, respectively.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.07	3.4591	0.92	3.4591
	3	0.25	0.9112	8.01	0.9013
	4	0.53	0.1044	7.95	0.1044
	5	8.72	0.0940	71.65	0.0940
	6	90.92	0.0895	1030.00	0.0895
	7	1028.44	0.0854	3600.00	0.0855
100	2	0.29	6.1540	10.10	6.1540
	3	1.07	1.4899	80.14	1.4899
	4	4.02	0.2041	121.05	0.2041
	5	106.39	0.1921	1969.82	0.1921
	6	2471.47	0.1799	3600.00	0.1876
200	2	1.01	11.8372	126.75	11.8372
	3	5.44	2.9371	1797.52	2.9371
	4	32.58	0.4192	1165.82	0.4192
	5	1795.14	0.4095	3600.00	0.4200
400	2	4.38	23.9315	3098.95	23.9315
	3	30.83	6.0669	3600.00	6.8201
	4	300.28	0.9724	3600.00	1.7208
800	2	19.94	46.7016	3600.00	47.7514
	3	209.47	11.8452	3600.00	13.5497

Table A.3: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *coslin* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.06	0.5753	0.51	0.5753
	3	0.89	0.5042	5.82	0.5042
	4	7.19	0.3160	84.99	0.3160
	5	13.03	0.0294	29.38	0.0294
	6	82.66	0.0183	292.54	0.0183
	7	392.80	0.0130	1378.41	0.0130
100	2	0.18	1.1390	7.39	1.1390
	3	4.90	0.9678	71.32	0.9678
	4	75.40	0.5554	1763.19	0.5576
	5	196.79	0.0531	2425.19	0.0534
	6	2122.54	0.0314	3600.00	0.0315
200	2	0.70	2.3114	55.39	2.3114
	3	34.47	1.9061	2542.63	1.9061
	4	988.15	1.1100	3600.00	1.1129
	5	3600.00	0.1185	3600.00	0.2545
400	2	2.77	4.5829	2658.01	4.5829
	3	279.66	3.6729	3600.00	3.8171
	4	3600.00	2.4240	3600.00	2.6586
800	2	12.69	9.2442	3600.00	9.4368
	3	2518.26	7.4096	3600.00	9.3481

Table A.4: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *cube* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.06	4.6964	0.48	4.6964
	3	0.58	3.4588	2.53	3.4588
	4	1.53	0.4325	15.74	0.4325
	5	10.10	0.3455	39.87	0.3455
	6	51.51	0.2558	408.95	0.2558
	7	210.92	0.2329	1997.10	0.2481
100	2	0.21	9.1059	2.53	9.1059
	3	3.86	6.6438	148.64	6.6438
	4	0.21	0.2232	212.95	0.2232
	5	4.71	0.2165	1806.63	0.2165
	6	78.87	0.2064	3600.00	0.2105
200	2	0.90	18.1362	86.03	18.1362
	3	27.39	13.1515	3570.80	13.1515
	4	0.71	0.4567	636.43	0.4567
	5	10.53	0.4144	3600.00	0.4306
400	2	3.38	36.1392	3043.60	36.1392
	3	250.25	26.4483	3600.00	27.0581
	4	5.95	0.9081	3600.00	1.0660
800	2	15.83	72.0712	3600.00	72.5003
	3	2164.30	52.7153	3600.00	84.4990

Table A.5: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *arctan* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.06	0.7367	0.32	0.7367
	3	1.48	0.5666	4.24	0.5666
	4	2.26	0.0318	6.99	0.0318
	5	1.18	0.0192	15.99	0.0192
	6	6.34	0.0095	38.59	0.0099
	7	27.02	0.0082	562.21	0.0082
100	2	0.24	1.4512	13.51	1.4512
	3	4.41	1.1025	68.04	1.1025
	4	0.24	0.0182	33.89	0.0239
	5	1.73	0.0137	67.53	0.0138
	6	2.04	0.0087	404.93	0.0088
200	2	0.89	2.9621	69.46	2.9623
	3	31.41	2.2638	1253.74	2.2649
	4	2.01	0.0423	556.68	0.0423
	5	9.87	0.0288	2285.96	0.0289
400	2	3.87	5.8450	1265.97	5.8452
	3	262.54	4.4643	3600.00	4.5481
	4	26.58	0.0982	3600.00	0.1008
800	2	16.78	11.9212	3600.00	11.9777
	3	2520.92	9.1307	3600.00	9.3086

Table A.6: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *rational* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.09	10.9860	0.65	10.9860
	3	0.72	8.6328	7.07	8.6328
	4	7.46	7.1044	32.63	7.1044
	5	57.07	5.7851	218.39	5.7851
	6	354.36	4.6809	877.02	4.6970
	7	1949.43	3.7589	2452.50	3.7589
100	2	0.33	23.0728	14.84	23.0728
	3	4.84	17.7147	220.74	17.7147
	4	97.05	15.0395	2556.93	15.0649
	5	1328.56	12.5097	3600.00	12.5606
	6	3600.00	20.4869	3600.00	11.5174
200	2	1.37	45.9278	238.58	45.9278
	3	36.65	37.6970	3600.00	37.9377
	4	1219.98	31.5434	3600.00	32.5706
	5	3600.00	43.8824	3600.00	27.6896
400	2	5.14	94.1947	3600.00	94.2006
	3	304.57	77.1099	3600.00	84.7810
	4	3600.00	72.2202	3600.00	78.1342
800	2	22.04	183.7737	3600.00	190.3930
	3	2891.00	151.8663	3600.00	190.7947

Table A.7: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *Inhom1* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.02	0.0182	0.43	0.0182
	3	0.21	0.0080	0.76	0.0080
	4	3.55	0.0069	7.23	0.0069
	5	29.26	0.0040	18.71	0.0040
	6	295.27	0.0037	113.66	0.0037
	7	2522.02	0.0035	840.06	0.0035
100	2	0.03	0.0333	1.14	0.0333
	3	0.76	0.0140	8.99	0.0142
	4	21.43	0.0100	80.74	0.0100
	5	472.39	0.0074	804.64	0.0074
	6	3600.00	0.0447	3600.00	0.0073
200	2	0.06	0.0579	48.01	0.0579
	3	3.17	0.0295	128.31	0.0295
	4	163.64	0.0194	3600.00	0.0195
	5	3600.00	0.6379	3600.00	0.0160
400	2	0.15	0.1016	201.55	0.1016
	3	14.48	0.0606	3500.27	0.0606
	4	1595.61	0.0460	3600.00	0.0630
800	2	0.46	0.1995	3214.82	0.1995
	3	69.01	0.1260	3600.00	0.1353

Table A.8: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *Inhom2* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.07	0.5018	0.39	0.5033
	3	0.55	0.3750	6.52	0.3750
	4	3.40	0.2077	52.92	0.2078
	5	26.18	0.1585	248.20	0.1585
	6	156.17	0.0964	882.31	0.0964
	7	832.57	0.0723	3600.00	0.0723
100	2	0.20	0.9394	10.03	0.9394
	3	3.46	0.6942	116.00	0.6942
	4	41.64	0.3957	791.54	0.3957
	5	456.95	0.2472	3600.00	0.2472
	6	3600.00	0.1336	3600.00	0.1382
200	2	0.62	1.7637	110.13	1.7637
	3	22.36	1.3302	2867.66	1.3302
	4	539.18	0.7983	3600.00	0.8076
	5	3600.00	0.4517	3600.00	0.4716
400	2	2.35	3.4699	3600.00	3.4702
	3	172.10	2.6261	3600.00	2.6495
	4	3600.00	1.5362	3600.00	1.9998
800	2	10.11	6.9963	3600.00	6.9993
	3	1645.45	5.3194	3600.00	8.1098

Table A.9: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the Inhom3 data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.07	0.1139	0.39	0.1139
	3	0.59	0.0970	6.93	0.0969
	4	9.91	0.0933	79.92	0.0933
	5	141.17	0.0911	1390.74	0.0911
	6	1049.85	0.0844	3600.00	0.0848
	7	3600.00	0.0822	3600.00	0.0820
100	2	0.26	0.2193	8.13	0.2193
	3	4.93	0.1962	155.72	0.1961
	4	76.86	0.1791	2794.67	0.1791
	5	2124.09	0.1732	3600.00	0.1732
	6	3600.00	0.1822	3600.00	0.1718
200	2	0.79	0.4427	73.99	0.4427
	3	22.35	0.4115	3600.00	0.4113
	4	873.29	0.3961	3600.00	0.4008
	5	3600.00	0.5014	3600.00	0.4021
400	2	2.88	0.9864	1851.05	0.9864
	3	240.02	0.9560	3600.00	0.9634
	4	3600.00	0.9737	3600.00	0.9574
800	2	13.78	2.1311	3600.00	2.1323
	3	2171.49	2.0462	3600.00	2.2193

Table A.10: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *logit* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.08	5.7810	0.70	5.7810
	3	0.49	0.8783	3.97	1.0780
	4	1.26	0.4670	10.91	0.4670
	5	0.79	0.1027	25.54	0.1035
	6	6.75	0.0912	178.70	0.0912
	7	77.91	0.0895	1472.02	0.0895
100	2	0.31	11.2940	14.80	11.2940
	3	3.12	1.5471	98.27	1.5471
	4	12.49	0.7787	1820.18	0.7791
	5	10.25	0.1781	2890.07	0.1781
	6	202.99	0.1698	3600.00	0.1732
200	2	1.18	22.5535	268.56	22.5535
	3	22.65	3.3619	1455.88	3.3619
	4	138.53	1.8640	3600.00	1.9237
	5	193.40	0.3961	3600.00	0.5218
400	2	5.24	45.7506	3600.00	45.7670
	3	178.02	6.7443	3600.00	6.9803
	4	1840.29	3.8048	3600.00	4.1288
800	2	24.44	91.0807	3600.00	101.2249
	3	1600.59	13.3475	3600.00	13.4037

Table A.11: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *bump* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.13	17.0711	1.46	17.0711
	3	0.97	10.4822	18.98	10.4822
	4	0.60	0.3702	13.73	0.3843
	5	6.01	0.2340	73.87	0.2413
	6	33.19	0.1027	44.48	0.1027
	7	289.52	0.0926	484.61	0.0938
100	2	0.52	33.1035	22.50	33.1035
	3	6.41	19.6492	416.24	19.6492
	4	3.60	0.4631	179.56	0.4631
	5	66.21	0.3024	803.71	0.3063
	6	935.16	0.1927	1983.31	0.1927
200	2	2.06	67.0331	398.53	67.0331
	3	44.15	38.5993	3600.00	39.3872
	4	30.74	0.9182	3600.00	0.9600
	5	1226.02	0.6415	3600.00	0.7174
400	2	9.10	135.4760	3600.00	136.1795
	3	359.62	77.8900	3600.00	80.1551
	4	301.97	1.9672	3600.00	3.1658
800	2	42.35	267.9905	3600.00	273.7274
	3	3528.50	153.3054	3600.00	154.2675

Table A.12: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *sine3* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.18	21.1645	3.65	21.1645
	3	1.48	18.4579	26.52	18.4579
	4	12.77	16.9435	170.77	16.9435
	5	56.04	14.2477	1146.07	14.2477
	6	342.77	12.7009	3600.00	12.7177
	7	923.43	10.0586	3600.00	10.0586
100	2	0.71	43.9084	28.31	43.9084
	3	9.75	38.2630	413.70	38.2630
	4	149.12	35.0655	3600.00	35.0655
	5	1257.85	29.4076	3600.00	29.4646
	6	3600.00	26.5850	3600.00	26.2792
200	2	2.94	87.8447	392.33	87.8447
	3	68.91	76.4570	3600.00	76.4825
	4	2025.08	69.9809	3600.00	76.3092
	5	3600.00	67.1722	3600.00	69.8451
400	2	12.61	175.9133	3600.00	176.5174
	3	534.93	153.9283	3600.00	157.7253
	4	3600.00	151.3318	3600.00	153.3733
800	2	58.69	357.5205	3600.00	364.8069
	3	3600.00	312.9597	3600.00	366.5322

Table A.13: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *sine6* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
50	2	0.04	0.1393	0.33	0.1393
	3	0.33	0.0974	3.24	0.0974
	4	6.99	0.0950	21.24	0.0950
	5	103.58	0.0928	409.67	0.0929
	6	938.14	0.0869	3600.00	0.0872
	7	3600.00	0.0878	3600.00	0.0845
100	2	0.10	0.2273	1.84	0.2273
	3	1.19	0.1807	27.76	0.1807
	4	49.77	0.1756	957.79	0.1756
	5	1625.99	0.1717	3600.00	0.1717
	6	3600.00	0.2070	3600.00	0.1697
200	2	0.24	0.4619	41.64	0.4620
	3	7.18	0.4044	1095.09	0.4044
	4	640.30	0.3977	3600.00	0.3993
	5	3600.00	0.5200	3600.00	0.3949
400	2	1.09	1.0607	1332.05	1.0607
	3	46.68	0.9583	3600.00	0.9581
	4	3600.00	1.4674	3600.00	0.9573
800	2	3.21	2.1295	3600.00	2.1295
	3	213.29	1.9869	3600.00	2.1089

Table A.14: CPU times (in seconds) and least-squares errors of the cubic spline functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *SpaHet3* data sets.

A.3 Least-Squares Polynomial Approximation with Free Knots

The test functions that were used to generate the data sets are listed in Table A.2. Five data sets of different sizes were generated per test function, totaling 60 data sets. In combination with varying numbers of free knots (see Table A.2), we obtain 240 problem instances

The results of the application of the B&B and MIQCP approaches to the 240 problem instances are shown in Tables A.15 to A.25. Each table shows the CPU time and the minimal least-squares error that was obtained with the B&B and MIQCP approaches for each of the 20 problem instances corresponding to one of the test functions.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.01	0.5107	0.23	0.7888
	3	0.10	0.2049	8.24	0.2049
	4	2.71	0.1405	28.72	0.1405
	5	55.67	0.1132	117.27	0.1132
	6	739.60	0.0943	475.81	0.0943
	7	3600.00	0.1425	3600.00	0.0785
200	2	0.01	1.0492	11.49	1.0492
	3	0.51	0.4759	140.00	0.4759
	4	31.80	0.3660	1658.04	0.3660
	5	1896.04	0.3368	3600.00	0.3425
	6	3600.00	0.3722	3600.00	0.3156
400	2	0.03	2.0178	114.60	2.0572
	3	2.40	1.0737	3082.96	1.0737
	4	351.82	0.8843	3600.00	0.8905
	5	3600.00	0.9197	3600.00	0.9049
800	2	0.09	4.3832	2462.61	4.3832
	3	14.45	2.4931	3600.00	2.5524
	4	3600.00	1.9357	3600.00	2.0062
1600	2	0.25	8.7039	3600.00	12.4831
	3	85.61	4.9216	3600.00	5.1713

Table A.15: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *coslin* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.01	0.2027	0.66	0.2027
	3	0.19	0.0398	1.22	0.0516
	4	4.06	0.0295	25.97	0.0309
	5	61.28	0.0221	374.50	0.0221
	6	691.22	0.0174	274.84	0.0174
	7	3600.00	0.0287	2654.69	0.0136
200	2	0.02	0.3993	2.72	0.3993
	3	0.87	0.0839	38.96	0.0839
	4	42.72	0.0679	176.24	0.0699
	5	1616.99	0.0566	1421.53	0.0566
	6	3600.00	0.1034	3600.00	0.0517
400	2	0.06	0.8939	129.56	0.8939
	3	3.98	0.1834	868.65	0.2434
	4	427.97	0.1549	3600.00	0.1553
	5	3600.00	0.7362	3600.00	0.1769
800	2	0.14	1.7095	2447.29	1.7095
	3	20.45	0.3953	3600.00	0.5423
	4	3600.00	3.3116	3600.00	1.1113
1600	2	0.45	3.5120	3600.00	17.2600
	3	115.35	0.7763	3600.00	8.3445

Table A.16: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *cube* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.00	0.1777	0.35	0.1819
	3	0.03	0.1533	10.20	0.1533
	4	0.62	0.1393	25.08	0.1393
	5	7.85	0.1216	186.67	0.1216
	6	99.07	0.1090	2028.98	0.1090
	7	863.58	0.0930	3600.00	0.0980
200	2	0.00	0.4382	1.60	0.4382
	3	0.12	0.3827	115.03	0.3846
	4	4.79	0.3631	686.84	0.3631
	5	177.41	0.3431	3600.00	0.3437
	6	3600.00	3.0895	3600.00	0.3250
400	2	0.01	0.8905	148.64	0.8905
	3	0.50	0.7637	904.91	0.7637
	4	38.04	0.7196	3600.00	0.7339
	5	3346.17	0.6906	3600.00	0.8234
800	2	0.03	1.7500	1011.89	1.7500
	3	2.46	1.5143	3600.00	2.3085
	4	473.32	1.4751	3600.00	1.9899
1600	2	0.13	3.6059	3600.00	7.4444
	3	14.31	3.0868	3600.00	10.7422

Table A.17: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *arctan* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.00	0.0176	0.39	0.0176
	3	0.01	0.0084	0.85	0.0084
	4	0.12	0.0055	13.25	0.0060
	5	2.28	0.0046	16.41	0.0046
	6	30.22	0.0037	76.87	0.0037
	7	312.59	0.0030	316.72	0.0030
200	2	0.01	0.0421	1.21	0.0421
	3	0.05	0.0209	36.38	0.0209
	4	1.06	0.0156	236.85	0.0156
	5	50.75	0.0141	1842.65	0.0141
	6	2181.12	0.0130	3600.00	0.0133
400	2	0.02	0.0979	234.93	0.0979
	3	0.26	0.0497	577.15	0.0498
	4	11.25	0.0383	1940.71	0.0383
	5	1215.64	0.0352	3600.00	0.0366
800	2	0.04	0.1930	1680.66	0.1930
	3	1.50	0.1090	3600.00	0.1152
	4	103.99	0.0804	3600.00	0.0910
1600	2	0.13	0.3863	3600.00	2.1625
	3	9.36	0.2192	3600.00	0.4026

Table A.18: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *rational* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.02	11.8090	0.62	11.8090
	3	0.36	7.7033	12.24	7.7033
	4	6.71	4.9851	56.67	4.9851
	5	97.32	2.5491	75.19	2.5491
	6	942.03	0.5317	80.76	0.5317
	7	3600.00	9.6872	66.09	0.1963
200	2	0.05	25.8713	3.74	25.8713
	3	1.95	18.8375	56.07	18.8375
	4	74.22	13.8222	338.44	13.8222
	5	2540.94	9.8131	1432.61	9.8131
	6	3600.00	23.7368	2451.69	6.8354
400	2	0.13	53.8883	220.64	53.8883
	3	9.48	39.3469	1237.96	39.3469
	4	823.45	30.2025	3600.00	33.4801
	5	3600.00	57.6433	3600.00	28.0787
800	2	0.34	104.5557	2882.40	104.5557
	3	45.47	74.8116	3600.00	75.0148
	4	3600.00	120.8058	3600.00	74.7258
1600	2	1.01	220.8426	3600.00	274.6243
	3	278.96	161.2179	3600.00	234.4409

Table A.19: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *Inhom1* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.01	0.0152	0.77	0.0152
	3	0.34	0.0062	1.20	0.0062
	4	7.44	0.0050	10.36	0.0050
	5	103.25	0.0041	115.76	0.0041
	6	853.59	0.0034	281.28	0.0034
	7	3600.00	0.0214	1252.50	0.0028
200	2	0.05	0.0330	14.62	0.0330
	3	1.43	0.0147	76.58	0.0147
	4	78.72	0.0135	322.23	0.0135
	5	3600.00	0.0128	3196.22	0.0128
	6	3600.00	0.0416	3600.00	0.0122
400	2	0.09	0.0680	92.18	0.0680
	3	7.41	0.0389	2096.09	0.0389
	4	910.11	0.0361	3082.60	0.0361
	5	3600.00	0.0904	3600.00	0.0355
800	2	0.20	0.1295	410.90	0.1295
	3	38.39	0.0844	3600.00	0.0859
	4	3600.00	0.5910	3600.00	0.0842
1600	2	0.64	0.2661	3600.00	0.2677
	3	220.47	0.1696	3600.00	0.4654

Table A.20: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *Inhom2* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.03	0.1668	3.38	0.1668
	3	0.57	0.1387	10.99	0.1387
	4	8.14	0.1132	54.52	0.1132
	5	102.96	0.0972	412.67	0.0972
	6	832.81	0.0812	1926.12	0.0812
	7	3600.00	0.1405	1049.07	0.0685
200	2	0.10	0.3798	17.30	0.3798
	3	4.52	0.3530	813.28	0.3556
	4	188.54	0.3314	3289.98	0.3314
	5	3600.00	0.3748	3600.00	0.3110
	6	3600.00	0.3755	3600.00	0.2977
400	2	0.39	0.9184	318.12	0.9184
	3	39.40	0.8815	3600.00	0.8831
	4	3279.72	0.8436	3600.00	0.8633
	5	3600.00	0.9149	3600.00	0.8564
800	2	1.16	1.9657	3600.00	1.9883
	3	243.69	1.9128	3600.00	1.9599
	4	3600.00	1.9714	3600.00	1.9355
1600	2	3.56	3.9553	3600.00	4.0251
	3	2652.56	3.9146	3600.00	3.9444

Table A.21: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *logit* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.00	0.3902	0.52	0.3902
	3	0.05	0.1545	6.79	0.1545
	4	1.20	0.1319	26.14	0.1319
	5	21.69	0.1099	116.37	0.1100
	6	283.56	0.0903	402.53	0.0903
	7	2622.32	0.0744	2717.25	0.0744
200	2	0.01	0.8147	2.84	0.8147
	3	0.26	0.3839	96.22	0.3847
	4	16.29	0.3563	363.81	0.3567
	5	790.82	0.3314	3600.00	0.3349
	6	3600.00	0.5516	3600.00	0.3140
400	2	0.03	1.7785	224.05	1.7785
	3	1.33	0.9032	847.92	0.9032
	4	192.22	0.8672	3600.00	0.8904
	5	3600.00	1.4422	3600.00	0.8478
800	2	0.06	3.5225	2068.52	3.5225
	3	6.59	1.9587	3600.00	3.2012
	4	2268.58	1.9045	3600.00	4.0858
1600	2	0.25	7.4082	3600.00	15.8993
	3	44.77	4.0155	3600.00	8.8137

Table A.22: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *bump* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.01	0.6713	1.63	0.6713
	3	0.09	0.3275	5.92	0.3275
	4	2.10	0.2116	23.19	0.2119
	5	31.14	0.1358	122.22	0.1358
	6	418.49	0.1047	321.66	0.1047
	7	3600.00	0.0872	1448.17	0.0843
200	2	0.01	1.3100	10.76	1.3100
	3	0.48	0.7596	23.73	0.8926
	4	18.37	0.4690	305.05	0.4690
	5	755.41	0.3860	1865.44	0.3860
	6	3600.00	0.3681	3600.00	0.3205
400	2	0.03	2.7867	82.23	2.7867
	3	1.97	1.5899	636.09	1.5899
	4	164.27	1.0238	3600.00	1.0481
	5	3600.00	1.0137	3600.00	0.9302
800	2	0.08	5.0479	1641.67	5.0479
	3	11.74	3.3369	3600.00	3.4325
	4	2149.06	2.3186	3600.00	2.4599
1600	2	0.23	10.7035	3600.00	10.7365
	3	69.69	6.8835	3600.00	8.8953

Table A.23: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *sine3* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.03	28.8515	7.51	28.8515
	3	0.14	16.0270	16.61	16.0270
	4	1.30	5.0837	22.61	5.0837
	5	9.33	0.7038	17.81	0.7038
	6	110.91	0.4032	78.31	0.4032
	7	1157.62	0.2463	208.57	0.2463
200	2	0.08	57.7344	25.33	57.7344
	3	0.59	30.9351	99.70	30.9351
	4	12.20	10.9239	200.98	11.0774
	5	150.92	1.2587	90.17	1.2587
	6	3600.00	2.0918	1428.42	0.9021
400	2	0.21	116.8796	602.18	116.8796
	3	2.79	61.8021	1474.82	61.8021
	4	118.40	22.8100	3600.00	22.9361
	5	2798.25	2.6731	1007.49	2.7798
800	2	0.55	237.2338	3600.00	239.2360
	3	13.94	124.2191	3600.00	164.3217
	4	1215.57	46.4083	3600.00	52.6634
1600	2	1.66	470.9995	3600.00	515.8757
	3	80.66	245.9257	3600.00	465.6945

Table A.24: CPU times (in seconds) and least-squares errors of the piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *sine6* data sets.

number of		B&B		MIQCP	
data points	knots	time	error	time	error
100	2	0.01	0.1626	0.84	0.1626
	3	0.40	0.1304	11.72	0.1357
	4	7.64	0.1107	47.41	0.1107
	5	92.76	0.0945	319.41	0.0945
	6	693.78	0.0786	2584.06	0.0786
	7	3600.00	0.1432	3600.00	0.0667
200	2	0.05	0.3973	17.64	0.3973
	3	3.12	0.3565	152.50	0.3585
	4	155.20	0.3318	2207.49	0.3318
	5	3600.00	0.3844	3600.00	0.3089
	6	3600.00	0.3763	3600.00	0.3000
400	2	0.14	0.9328	154.48	0.9328
	3	25.45	0.9008	3600.00	0.9008
	4	2788.58	0.8651	3600.00	0.8833
	5	3600.00	0.9236	3600.00	0.8550
800	2	0.56	2.0049	3015.71	2.0049
	3	185.92	1.9438	3600.00	1.9448
	4	3600.00	2.0275	3600.00	1.9492
1600	2	1.68	3.9740	3600.00	4.2123
	3	1516.48	3.9197	3600.00	3.9926

Table A.25: CPU times (in seconds) and least-squares errors of piecewise polynomial functions corresponding to the knot placements computed with the B&B and MIQCP approaches for the *SpaHet3* data sets.

Erklärung über verwendete Hilfsmittel

Ich versichere wahrheitsgemäß, die Dissertation bis auf die in der Abhandlung angegebene Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und genau kenntlich gemacht zu haben, was aus Arbeiten anderer und aus eigenen Veröffentlichungen unverändert oder mit Abänderungen entnommen wurde.

Ort und Datum　　　　　　　　Unterschrift