Gradient methods with higher-order information for unconstrained optimization

Hinskens, R.

Award date:
2018

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics
and Computer Science

Master Thesis

Gradient methods with higher-order information for unconstrained optimization

Author:
R. Hinskens

Supervisor:
dr. M.E. Hochstenbach
Committee members:
dr. ir. C.A.J. Hurkens
dr. E.J. Bekkers

August 16, 2018

Abstract

The purpose of this thesis is to investigate iterative gradient methods and their performance for general unconstrained optimization problems. Two famous methods are the classical Steepest Descent and the Barzilai–Borwein methods. The latter often seems to perform better in test cases than the Steepest Descent. The reason for this improvement is not completely understood. However, one can still use the strategy to develop new methods with useful properties. Some of these methods were developed by looking at the application to a quadratic example. Several gradient methods force a certain decrease in the objective function, which is often desired. Some methods do not contain such conditions. In this thesis we will analyze the decreasing behavior of these methods.

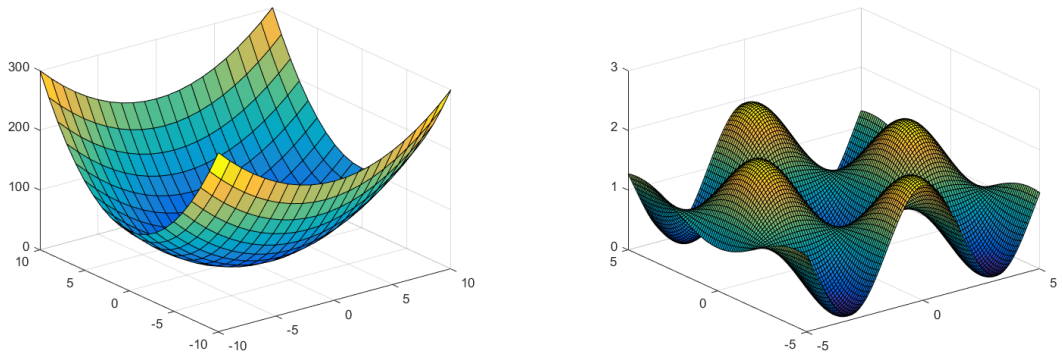
A specific class of these methods that use information of more that one previous step are called the limited memory methods. These methods use higher order information by looking at the spectral properties of the Hessian of the objective function. However, the Hessian might not be available. Furthermore we will discuss some implementing issues concerning these methods. The known limited memory methods are analyzed and a new limited memory method is introduced. In this thesis we will discuss some numerical observations of the described methods applied to both the quadratic case and some general nonlinear test cases.

Contents

CHAPTER 1

Introduction

Large-scale unconstrained optimization is very important due to the great amount of real life applications. The very large size of these problems make first-order methods (or gradient methods in particular) a suitable choice. Gradient methods have recently proved their effectiveness in solving these problems from fields such as signal and image processing, machine learning, optics, chemistry, social sciences and other areas (see, e.g., [4]).



(a) Quadratic function

(b) Griewank function [19]

Figure 1.1: Two examples for test functions

Optimization algorithms are often iterative. An initial guess is required so that a sequence of improved estimates can be computed until they hopefully terminate at a solution. Some algorithms use information gathered at previous iterations, while others use only local information obtained at the current point. Good algorithms should possess properties like *robustness*, *efficiency* and *accuracy*. It is important that an algorithm should be well-applicable to many problems in a certain class of problems, it should not require too much computer time or storage and it should not be overly sensitive for errors, such as rounding errors.

Often these methods do not attain all of these goals at the same time. For example a method might be very fast in finding the global minimum for the function in Figure 1.1a, but it might get stuck in a local minimum for the function in Figure 1.1b. Many of these algorithms are based on the most basic gradient-based methods: the Steepest Descent methods.

State of the art

Research for new iterative methods for unconstrained optimization has been very popular in the past decades due to the research of Barzilai and Borwein in 1988 [2]. Their discoveries have led to new ideas for improving these gradient methods by making suitable choices for the step lengths. New insights have been developed by looking at the spectral properties of the Hessian of the objective function and using information of more than one previous iteration. The extension of step length selection strategies from convex quadratic to general nonlinear optimization has involved interesting theoretical issues, leading to new line search strategies (see, e.g., [16]).

Research objectives

Many applications of unconstrained optimization contain thousands or more variables. Problems of this size require that the computer storage and computational costs are kept at a tolerable level. A diverse collection of large-scale optimization methods has been developed to achieve this goal, each being particularly effective for certain problem types. Our goal is to compare the robustness, efficiency and accuracy of some popular methods and construct a new method by combining the best of the other known methods. In this thesis we will restrict ourselves to optimization problems without boundary conditions. In mathematical terms, we want to solve the following optimization problem for the continuously differentiable objective function $f : \mathbb{R}^n \to \mathbb{R}$:

$$\min_{x \in \mathbb{R}^n} f(x). \tag{1.1}$$

We will look at different iterative methods, for example of the form $x_{k+1} = x_k + \alpha_k p_k$, where α_k is the step length and p_k is the search direction. These methods should converge to the vector at which the objective function attains its minimal value. Some methods require an improvement after every iteration, but some methods allow some backsliding to prevent getting stuck in local minima. Researchers have made great improvements in the performance of these methods by simply choosing the step length for each iteration in a smart way.

Outline of this thesis

In **chapter 2** we address some preliminary knowledge on the topics we discuss later on. Furthermore some notations and definitions are introduced. In **chapter 3** some famous gradient methods are analyzed that form a basis for the rest of the thesis. In **chapter 4** a quadratic example is discussed in detail of which the Hessian and other useful information is known. For this example we prove some convergence results and properties for the different methods. The quadratic example forms a basis for the methods in the upcoming chapters. In **chapter 5** more advanced so-called limited memory methods are discussed and analyzed based on very recent research. Furthermore we compare numerical results of the discussed methods for the quadratic case. In **chapter 6** we generalize the methods to be applicable to general nonlinear problems of which the Hessian and other information might be unknown. We compare numerical results of some generalized methods on a set of test cases gathered from the literature. In **chapter 7** we draw the conclusions and summarize the work. Finally, in **chapter 8** some possible future work is addressed.

CHAPTER 2

Preliminaries

In this chapter basic definitions and concepts are introduced regarding gradient methods. These concepts will be used in the following chapters. The reader should be familiar with the basics in numerical linear algebra and continuous optimization (in particular gradient methods). For chapters 5 and 6 it is recommended to have some preliminary knowledge on spectral theory. All implementations are done in MATLAB R2016b.

Notation and definitions

Vectors and matrices

$x \in \mathbb{R}^n$ is the real valued column *vector* $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ with $\|\cdot\|$ the ℓ^2 (Euclidean) vector *norm*.

$A \in \mathbb{R}^{m \times n}$ is a *matrix* of size $m \times n$ with entries $A_{i,j}$, where $i = 1, \ldots, m$ and $j = 1, \ldots, n$. We need the following notations and definitions concerning matrices:

I is the *identity* matrix, where $I_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$.

A^T is the *transposed* matrix of A, where $A^T_{i,j} = A_{j,i}$.

A^{-1} is the *inverse* matrix of A, where $A^{-1}A = I$.

$\lambda \in \mathbb{R}$ is an *eigenvalue* of matrix A with corresponding *eigenvector* $x \in \mathbb{R}^n$ if $Ax = \lambda x$.

A matrix $A \in \mathbb{R}^{n \times n}$ is called *orthogonal* if $A^{-1} = A^T$.

A matrix $A \in \mathbb{R}^{n\times n}$ is called *symmetric* if $A^T = A$.

A matrix $A \in \mathbb{R}^{n\times n}$ is called *positive definite* if $x^TAx > 0$ for all nonzero $x \in \mathbb{R}^n$, or equivalently, if all eigenvalues of A are positive.

A matrix $A \in \mathbb{R}^{n\times n}$ is called *positive semidefinite* if $x^TAx \geq 0$ for all nonzero $x \in \mathbb{R}^n$, or equivalently, if all eigenvalues of A are nonnegative.

For a given symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a nonzero vector $x \in \mathbb{R}^n$, the *Rayleigh quotient* is defined as $\mathcal{R}(A, x) = \dfrac{x^TAx}{x^Tx}$.

The matrix $\frac{1}{2}(A + A^T)$ is called the *symmetric part* of A.

Functions and sequences

We need the following notations and definitions concerning functions:
Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. Its *gradient* is the vector of partial derivatives, i.e.,

$$\nabla f(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{bmatrix},$$

and its *Hessian* is the symmetric $n \times n$ matrix

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} f(x) & \frac{\partial^2}{\partial x_1 \partial x_2} f(x) & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n} f(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1} f(x) & \frac{\partial^2}{\partial x_2^2} f(x) & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n} f(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} f(x) & \frac{\partial^2}{\partial x_n \partial x_2} f(x) & \cdots & \frac{\partial^2}{\partial x_n^2} f(x) \end{bmatrix}.$$

From now on we denote $\{x_k\}_{k\in\mathbb{N}}$ as a sequence of vectors $x_k \in \mathbb{R}^n$ and denote $x_k^{(i)}$ as the ith component of vector x_k.
We denote f_k as the function $f : \mathbb{R}^n \to \mathbb{R}$ evaluated at vector $x_k \in \mathbb{R}^n$: $f(x_k)$.

We denote ∇f_k as the gradient of $f : \mathbb{R}^n \to \mathbb{R}$ evaluated at vector $x_k \in \mathbb{R}^n$: $\nabla f(x_k)$.

We denote $\nabla^2 f_k$ as the Hessian of $f : \mathbb{R}^n \to \mathbb{R}$ evaluated at vector $x_k \in \mathbb{R}^n$: $\nabla^2 f(x_k)$.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *convex* if $\forall x, y \in \mathbb{R}^n$, $\forall t \in [0, 1]$:
$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y)$.

For a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and vectors b, $c \in \mathbb{R}^n$, a function $f : \mathbb{R}^n \to \mathbb{R}$ of the form $f(x) = x^TAx + b^Tx + c$ is called a *quadratic form*.

A sequence $\{a_n\}_{n\in\mathbb{N}}$ is (Q)-*linear convergent* to a^* if there exists a rate of convergence $r \in (0,1)$ and $n_0 \in \mathbb{N}$ such that $\dfrac{\|a_{n+1} - a^*\|}{\|a_n - a^*\|} \leq r$ for all $n \geq n_0$.

A sequence $\{a_n\}_{n\in\mathbb{N}}$ is *R-linear convergent* to a^* if there exists an $n_0 \in \mathbb{N}$ such that $\|a_n - a^*\| \leq \sigma_n$ for all $n \geq n_0$, where $\{\sigma_n\}_{n\in\mathbb{N}}$ converges Q-linearly to 0.

Optimization

We need the following definitions and observations concerning optimization:

A vector x^* is a *global minimizer* of function $f : \mathbb{R}^n \to \mathbb{R}$ if $f(x) \geq f(x^*)$ for all $x \in \mathbb{R}^n$.

A vector x^* is a *local minimizer* of function $f : \mathbb{R}^n \to \mathbb{R}$ if there exists a neighborhood U (open set) of x^* such that $f(x) \geq f(x^*)$ for all $x \in U$.

A vector x^* is a *strict local minimizer* of function $f : \mathbb{R}^n \to \mathbb{R}$ if there exists a neighborhood U (open set) of x^* such that $f(x) > f(x^*)$ for all $x \in U \backslash \{x^*\}$.

Second-order necessary conditions for a minimum as proved in [27] is the following: If x^* is a local minimizer of the continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ and the Hessian $\nabla^2 f$ exists, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

Second-order sufficient conditions for a minimum as proved in [27] is the following: Suppose $\nabla^2 f$ is continuous in an open neighborhood of x^*, $\nabla^2 f(x^*)$ is positive definite and $\nabla f(x^*) = 0$, then x^* is a strict local minimizer of f.

An iterative gradient method is called *monotonic* if $\|f_{k+1} - f^*\| \leq \|f_k - f^*\|$ for all $k \in \mathbb{N}$, where f^* is a solution to the optimization problem $\min\limits_{x\in\mathbb{R}^n} f(x)$.

Supplementary

We also need the following notations and definitions:

`tol` is the tolerance of a method, i.e., a bound for the gradient or function value at which the method terminates.

$[m]$ is the equivalence class modulo m starting with 1; identical to the set $\{1, \dots m\}$.

The *order-r Krylov subspace* generated by $A \in \mathbb{R}^{n\times n}$ and a vector $b \in \mathbb{R}^n$ is the linear subspace spanned by the images of b under the first r powers of A (starting from $A^0 = I$), i.e., $\mathcal{K}_r(A, b) = \langle b, Ab, A^2b, \dots, A^{r-1}b\rangle$.

QR-decomposition: Any matrix $A \in \mathbb{R}^{n\times m}$ with $m \leq n$ can be decomposed as the product of an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ and an upper triangular matrix $R \in \mathbb{R}^{n\times m}$.
Note that the bottom $(n - m)$ rows of R consists of zeros, i.e., $A = QR = Q\begin{bmatrix}\tilde{R}\\ 0\end{bmatrix} = \tilde{Q}\tilde{R}$, where $\tilde{Q}$ is the $n \times m$ matrix such that $Q = \begin{bmatrix}\tilde{Q} & \hat{Q}\end{bmatrix}$. Here $\tilde{Q}\tilde{R}$ is called the *reduced QR-decomposition* of A.

CHAPTER 3

First order gradient methods

Gradient methods of the first order have a search direction based on only first order partial derivatives. These line search methods compute a search direction p_k and then decide how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{3.1}$$

where the positive scalar α_k is called the step length. The efficiency, accuracy and robustness of a method depend on smart choices of both the search direction and the step length. In this thesis several choices are investigated. Many line search algorithms require p_k to be a descent direction, i.e., $\nabla f_k^T p_k < 0$. That forces a reduction of the function f along the search direction if the step length is small enough. In that case it ensures monotonic behavior. There also exist nonmonotone algorithms that do not force a decrease in f at every step. Those methods require f to be decreased after some predetermined j iterations. In other words $f_k < f_{k-j}$. That makes it sometimes possible to escape from local minimizers. One famous method that does ensure monotonic behavior is called the Steepest Descent method.

3.1 Classical Steepest Descent method

This method was introduced by Cauchy [6] in 1847. In the classical Steepest Descent method we have an iteration of the form (3.1), where the step length is given by

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha p_k) \tag{3.2}$$

and the search direction is given by $p_k = -\nabla f(x_k)$.

Algorithm 1: Steepest Descent with exact line search

Input: $x_1 \in \mathbb{R}^n$, $\texttt{tol} > 0$, $f : \mathbb{R}^n \to \mathbb{R}$
Output: Approximate solution to $\arg\min_{x \in \mathbb{R}^n} f(x)$

$p_1 = -\nabla f_1$
if $\|p_1\| < \texttt{tol}$
  return x_1
for $k = 1, 2, \ldots$
  $\alpha_k = \arg\min_{\alpha > 0} f(x_k + \alpha p_k)$
  $x_{k+1} = x_k + \alpha_k p_k$
  $p_{k+1} = -\nabla f_{k+1}$
  if $\|p_{k+1}\| < \texttt{tol}$
    return x_{k+1}
  end if
end

For small to medium-scale problems each iteration of this method is inexpensive. It requires only vector operations, which makes it also efficient for large-scale problems. Note that, by construction, the Steepest Descent method is monotonic. Despite the simplicity of this method and the use of the direct line search the iterates may converge slowly to the optimum as we will prove later on in Lemma 4.1.3.

3.1.1 Armijo and Wolfe conditions

The method of Steepest Descent contains a step length rule that might be costly or infeasible in practice. Therefore more efficient methods are used. The more recent Armijo gradient method uses backtracking line search and works better in practice [27]. Armijo line search is an inexact line search condition that decreases the step length until a sufficient reduction of the objective function f is satisfied. This is measured by the following inequality:

$$f(x_k + \alpha p_k) \leq f_k + c\,\alpha \nabla f_k^T p_k, \tag{3.3}$$

for some $c \in (0, 1)$.

In other words, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla f_k^T p_k$. This inequality is called the *Armijo condition*. Beck [4] uses $c = 0.25$ in his examples, but often in practice, the parameter c is chosen to be much smaller as discussed in [27], for example $c = 10^{-4}$.

Algorithm 2: Steepest Descent with Armijo line search

Input: $x_1 \in \mathbb{R}^n$, $\texttt{tol} > 0$, $f : \mathbb{R}^n \to \mathbb{R}$, $\tau \in (0,1)$, $c \in (0,1)$.
Output: Approximate solution to $\arg\min_{x \in \mathbb{R}^n} f(x)$

$p_1 = -\nabla f_1$
if $\|p_1\| < \texttt{tol}$
  return x_1
for $k = 1, 2, \ldots$
  Compute step length $\alpha_k = \tau^{\gamma_k}$, where $\gamma_k \in \mathbb{N}$ such that (3.3) is satisfied for $\alpha = \tau^{\gamma_k}$, but not for $\alpha = \tau^{\gamma_k - 1}$
  $x_{k+1} = x_k + \alpha_k p_k$
  $p_{k+1} = -\nabla f_{k+1}$
  if $\|p_{k+1}\| < \texttt{tol}$
    return x_{k+1}
  end if
end

Another strategy to compute the step length is by adding the following curvature condition:

$$\nabla f(x_k + \alpha p_k)^T p_k \geq \tilde{c}\, \nabla f_k^T p_k,$$

where $\tilde{c} \in (c, 1)$. Nocedal and Wright [27] explain that typical values of $\tilde{c}$ are 0.9 when the search direction p_k is chosen by a Newton or quasi-Newton method, and 0.1 when p_k is obtained from a nonlinear conjugate gradient method. This condition together with the Armijo condition (3.3) are called the *Wolfe conditions* [4].

3.2 Barzilai–Borwein method

In [2], Barzilai and Borwein show how a suitable choice of the step length can significantly adjust the speed of convergence of the classical Steepest Descent method. Since then, many new step length rules have been discovered to improve the speed of convergence of these gradient methods. Barzilai and Borwein introduced two new methods based on injecting second-order information into the step lengths [2]:

$$\alpha_k = \arg\min_{\alpha > 0} \|\alpha^{-1}(x_k - x_{k-1}) - (p_k - p_{k-1})\|, \tag{3.4}$$

$$\alpha_k = \arg\min_{\alpha > 0} \|(x_k - x_{k-1}) - \alpha(p_k - p_{k-1})\|. \tag{3.5}$$

The extra information causes the method to display nonmonotonic behavior. We will discuss this step length choice in more detail in section 4.3. In practice, the Barzilai–Borwein methods often perform better than the Steepest Descent method.

Now we have introduced the classical Steepest Descent and Barzilai–Borwein methods we will investigate some useful properties of these methods in the next chapter, which might lead to the development of new methods and a better understanding of the performance of the Barzilai–Borwein methods.

CHAPTER 4

Application to the quadratic case

To investigate the performance of the Steepest Descent and Barzilai–Borwein methods we will restrict ourselves to an example where the Hessian of the objective function is known. This might lead to interesting connections between the spectral properties of the Hessian and the convergence rate of these methods. Therefore we will focus on the minimization of a convex quadratic function. Consider the function $f : \mathbb{R}^n \to \mathbb{R}$, given by $f(x) = \frac{1}{2}x^T Ax$, where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite (SPD). See Figure 4.1 for a 2D example.

Since A is SPD, we have that all its eigenvalues are positive and thus the equation $Ax = 0$ only has the trivial solution $x = 0$ and hence we conclude that f has a unique global minimum in $x = 0$.

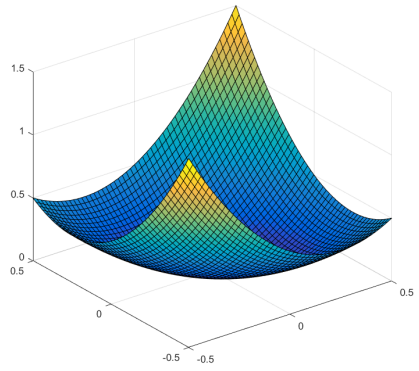


Figure 4.1: Quadratic example with $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

Now we can derive the gradient and the Hessian of this objective function. Matrix A is symmetric, hence $x^TAy = y^TAx$. We use this to calculate the gradient of f:

$$f(x+h) - f(x) = \tfrac{1}{2}(x+h)^TA(x+h) - \tfrac{1}{2}x^TAx = h^TAx + \tfrac{1}{2}h^TAh,$$

which implies that $\nabla f(x) = Ax$. The Hessian follows directly from its definition: $\nabla^2 f(x) = A$.

Next we will apply both the classical Steepest Descent method as well as the Barzilai–Borwein method to this example and get some interesting results concerning convergence and eigenvalues of the Hessian. We will investigate multiple choices for step lengths as well. One choice is the exact (Cauchy) step length.

4.1 Cauchy step length

For the quadratic example we can calculate the exact step length from (3.2) as follows:

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha p_k) = \arg\min_{\alpha>0} \tfrac{1}{2}(x_k + \alpha p_k)^TA(x_k + \alpha p_k).$$

To derive a formula for α_k we define the function ϕ_k as follows:

$$\phi_k(\alpha) := \tfrac{1}{2}(x_k + \alpha p_k)^TA(x_k + \alpha p_k).$$

Using the fact that A is symmetric we find the following expression for the derivative of ϕ_k:

$$\phi_k'(\alpha) = (x_k + \alpha p_k)^TAp_k.$$

Setting this derivative to zero results in the following expression for α_k:

$$\alpha_k^{\mathrm{SD}} = \frac{p_k^Tp_k}{p_k^TAp_k}. \tag{4.1}$$

Note that this is the reciprocal of the Rayleigh quotient of A and p_k.

4.1.1 Orthogonality property

A. Beck [4] shows that for the Steepest Descent method with exact step length (4.1) the search direction p_{k+1} is orthogonal to the search direction of the previous step, i.e., $p_k^Tp_{k+1} = 0$. We prove this in the following lemma:

Lemma 4.1.1. *Let $\{x_k\}_{k\in\mathbb{N}}$ be the sequence generated by the gradient method with exact line search for solving a problem of minimizing a continuously differentiable function f. Then for any $k = 1, 2, \ldots$ the following orthogonality property holds:*

$$(x_{k+2}x_{k+1})^T(x_{k+1}x_k) = 0.$$

Proof. We have given the gradient method (3.1), hence we have that $x_{k+1} - x_k = -\alpha_k\nabla f_k$ and $x_{k+2} - x_{k+1} = -\alpha_{k+1}\nabla f_{k+1}$. From this, we only have to prove that $\nabla f_k^T\nabla f_{k+1} = 0$. Define the function $\phi_k(\alpha) := f(x_k - \alpha\nabla f_k)$. Since $\alpha_k = \arg\min_{\alpha\geq 0}\phi_k(\alpha)$ and the optimal solution is not $\alpha_k = 0$, it follows that $\phi_k'(\alpha_k) = 0$. Hence

$$(-\nabla f_k)^T\nabla f(x_k - \alpha_k\nabla f_k) = 0,$$

which means that the result $\nabla f_k^T\nabla f_{k+1} = 0$ holds. □

4.1.2 Convergence rate

To derive an expression for the convergence rate of the method with this step length we need Lemma 4.1.2 and Lemma 4.1.3.

Lemma 4.1.2. *Let A be a symmetric positive definite matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$. Then $\min_{\|x\|=1} x^T A x = \min_{\|y\|=1} y^T \Lambda y$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$.*

Proof. A is symmetric positive definite, so we can write $A = Q\Lambda Q^T$ (eigenvalue decomposition). Hence Q is orthogonal and thus $Q^TQ = I$. Apply this property together with $x = Qy$ to the minimization problem:

$$\begin{aligned}\min_{\|x\|=1} x^T A x &= \min_{\|x\|=1} x^T Q\Lambda Q^T x \\ &= \min_{\|y\|=1} y^T Q^T Q\Lambda Q^T Q y \\ &= \min_{\|y\|=1} y^T \Lambda y.\end{aligned}$$

□

Hence without loss of generality we can assume A diagonal in the analysis of the stated optimization problem, which we will use in the numerical experiments in section 5.6. Note that the lemma also holds for the maximization problem.

Lemma 4.1.3. ***(Kantorovich inequality** [24])*
Let A be a symmetric positive definite matrix. For any vector $x \neq 0$, the following inequality holds:

$$\frac{(x^Tx)^2}{(x^TAx)(x^TA^{-1}x)} \geq \frac{4\lambda_{\min}\lambda_{\max}}{(\lambda_{\min}+\lambda_{\max})^2},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are, respectively, the smallest and largest eigenvalues of A.

Proof. **Step 1**: We calculate $\max_{\|x\|=1} x^TAx$.
Denote $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A. Since A is SPD, we can use the previous lemma to obtain the following:

$$\max_{\|x\|=1} x^TAx = \max_{\|y\|=1} y^T\Lambda y = \max_{\|y\|=1} \sum_{i=1}^{n} \lambda_i y_i^2 \leq \max_{\|y\|=1} \lambda_{\max}\|y\|^2 = \lambda_{\max}.$$

Now all we have to check is if x^TAx might actually reach this upper bound to get the result that the maximum is actually $\lambda_{\max}$. Since $\lambda_{\max}$ is an eigenvalue of A, we know that $\exists i \in \{1, \ldots, n\}$ such that $\lambda_i = \lambda_{\max}$. Now take the following vector with the property that $\|y\| = 1$: $y = [0, \ldots, 0, 1, 0, \ldots, 0]^T$, where $y_j = 1$ if $j = i$ and $y_j = 0$ otherwise. Then we get that $\max_{\|y\|=1} \sum_{i=1}^{n} \lambda_i y_i^2 = \lambda_{\max}$ and we get the result.

Step 2: We show that $\max_{t\in[\lambda_{\min},\lambda_{\max}]} \left(t + \frac{\lambda_{\min}\lambda_{\max}}{t}\right) = \lambda_{\min} + \lambda_{\max}$.

Define $g(t) = t + \dfrac{\lambda_{\min}\lambda_{\max}}{t}$, with $t \in [\lambda_{\min}, \lambda_{\max}]$.
Note that since A is SPD, we have that $\lambda_{\min}$ and $\lambda_{\max}$ are positive and thus t is positive. We get that $g''(t) = 1 + \dfrac{2\lambda_{\min}\lambda_{\max}}{t^3} > 0$. This implies that g is convex and thus will g take its extreme values at the boundary. Since $g(\lambda_{\min}) = g(\lambda_{\max}) = \lambda_{\min} + \lambda_{\max}$ we get the result.

Step 3: We show that the maximum eigenvalue of $A + \lambda_{\min}\lambda_{\max}A^{-1}$ is $\lambda_{\min} + \lambda_{\max}$.
Let x be an eigenvector of A with eigenvalue λ. Then:

$$\begin{aligned}(A + \lambda_{\min}\lambda_{\max}A^{-1})x &= Ax + \lambda_{\min}\lambda_{\max}A^{-1}x \\ &= \lambda x + \lambda_{\min}\lambda_{\max}(Q\Lambda Q^T)^{-1})x \\ &= \lambda x + \lambda_{\min}\lambda_{\max}(Q^{-T}\Lambda^{-1}Q^{-1})x \\ &= \lambda x + \lambda_{\min}\lambda_{\max}(Q\Lambda^{-1}Q^T)x \\ &= \lambda x + \frac{\lambda_{\min}\lambda_{\max}}{\lambda}x.\end{aligned}$$

The last step is true because we can observe that A^{-1} has eigenvalue-decomposition $Q\Lambda^{-1}Q^T$, which implies that if λ_j $(j = 1, \ldots, n)$ are the eigenvalues of A, then $\frac{1}{\lambda_j}$ are the eigenvalues of A^{-1}. Therefore we know now that $\lambda_j + \frac{\lambda_{\min}\lambda_{\max}}{\lambda_j}$ are the eigenvalues of $A + \lambda_{\min}\lambda_{\max}A^{-1}$. We use step 2 to get the result that the maximum eigenvalue of $A+\lambda_{\min}\lambda_{\max}A^{-1}$ is $\lambda_{\min}+\lambda_{\max}$.

Step 4: We show that for any two real numbers a and b, we have that $ab \leq \frac{1}{4}(a+b)^2$.
We know that $0 \leq (a-b)^2 = a^2 - 2ab + b^2$ and thus $4ab \leq a^2 + 2ab + b^2$. This gives the result.

Step 5: We show that $(x^TAx)\lambda_{\min}\lambda_{\max}(x^TA^{-1}x) \leq \frac{1}{4}(\lambda_{\min} + \lambda_{\max})^2(x^Tx)^2$.
First we apply step 4:

$$\begin{aligned}(x^TAx)\lambda_{\min}\lambda_{\max}(x^TA^{-1}x) &\leq \frac{1}{4}[(x^TAx) + \lambda_{\min}\lambda_{\max}(x^TA^{-1}x)]^2 \\ &= \frac{1}{4}[x^TAx + x^T(\lambda_{\min}\lambda_{\max}A^{-1})x]^2 \\ &= \frac{1}{4}[x^T(A + \lambda_{\min}\lambda_{\max}A^{-1})x]^2 \\ &\leq \frac{1}{4}[x^T(\lambda_{\min} + \lambda_{\max})x]^2 \\ &= \frac{1}{4}(\lambda_{\min} + \lambda_{\max})^2(x^Tx)^2,\end{aligned}$$

where we used step 3 at the last inequality.

Step 6: Concluding proof.
Rewriting the inequality from step 5 results in the Kantorovich inequality:

$$\frac{(x^Tx)^2}{(x^TAx)(x^TA^{-1}x)} \geq \frac{4\lambda_{\min}\lambda_{\max}}{(\lambda_{\min} + \lambda_{\max})^2}.$$

□

Now we can derive the convergence rate by looking at the following difference:

$$\begin{aligned} f_k - f_{k+1} &= \frac{1}{2}x_k^T A x_k - \frac{1}{2}x_{k+1}^T A x_{k+1} \\ &= \frac{1}{2}x_k^T A x_k - \frac{1}{2}(x_k - \alpha_k \nabla f_k)^T A (x_k - \alpha_k \nabla f_k) \\ &= \frac{1}{2}\alpha_k x_k^T A \nabla f_k + \frac{1}{2}\alpha_k f_k^T A x_k + \frac{1}{2}\alpha_k^2 \nabla f_k^T A \nabla f_k \\ &= \alpha_k x_k^T A \nabla f_k - \frac{1}{2}\alpha_k^2 \nabla f_k^T A \nabla f_k. \end{aligned}$$

The last step is true because A is a symmetric matrix, which implies we can use $x^T A y = y^T A x$. Now we can substitute α_k and use $x_k^T A = \nabla f_k^T$ to obtain the following.

$$\begin{aligned} f_k - f_{k+1} &= \frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T A \nabla f_k} x_k^T A \nabla f_k - \frac{1}{2}\left(\frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T A \nabla f_k}\right)^2 \nabla f_k^T A \nabla f_k \\ &= \frac{(\nabla f_k^T \nabla f_k)^2}{\nabla f_k^T A \nabla f_k} - \frac{1}{2}\frac{(\nabla f_k^T \nabla f_k)^2}{\nabla f_k^T A \nabla f_k} \\ &= \frac{1}{2}\frac{(\nabla f_k^T \nabla f_k)^2}{\nabla f_k^T A \nabla f_k} \\ &= \frac{(\nabla f_k^T \nabla f_k)^2}{(\nabla f_k^T A \nabla f_k)(x_k^T A x_k)}\frac{1}{2}x_k^T A x_k. \end{aligned}$$

Now we can use $x_k^T A x_k = \nabla f_k^T A^{-1} \nabla f_k$ to obtain the following and apply the *Kantorovich inequality* (Lemma 4.1.3):

$$\begin{aligned} f_k - f_{k+1} &= \frac{(\nabla f_k^T \nabla f_k)^2}{(\nabla f_k^T A \nabla f_k)(\nabla f_k^T A^{-1} \nabla f_k)} f_k \\ &\geq \frac{4\lambda_{\max}\lambda_{\min}}{(\lambda_{\max} + \lambda_{\min})^2} f_k. \end{aligned}$$

Here $\lambda_{\min}$ and $\lambda_{\max}$ are respectively the smallest and largest eigenvalues of A. Hence the convergence rate of a step of the Steepest Descent method is:

$$f_{k+1} \leq \frac{(\lambda_{\max} - \lambda_{\min})^2}{(\lambda_{\max} + \lambda_{\min})^2} f_k. \tag{4.2}$$

4.1.3 Asymptotic behavior

In [1] is shown that the SD method asymptotically reduces its search in the two-dimensional subspace spanned by q_1 and q_n, i.e.,

$$p_k \approx \mu_1^k q_1 + \mu_2^k q_n,$$

for some weights μ_1 and μ_2. Here q_1 and q_n are the eigenvectors of A corresponding to respectively the smallest and largest eigenvalue of A.

To reduce the amount of iterations we could choose the starting vector x_1 close to this two-dimensional subspace:

$$x_1 = \frac{sq_1 + (1-s)q_n + \varepsilon r}{\|sq_1 + (1-s)q_n + \varepsilon r\|},$$

where r is a normalized random vector from the Normal distribution with mean 0 and $s \in [0, 1]$. If ε is small, x_1 is chosen close to the two dimensional subspace. In Figure 4.2 convergence results are shown for the quadratic example with $A = \text{diag}(1, \ldots, 500)$.

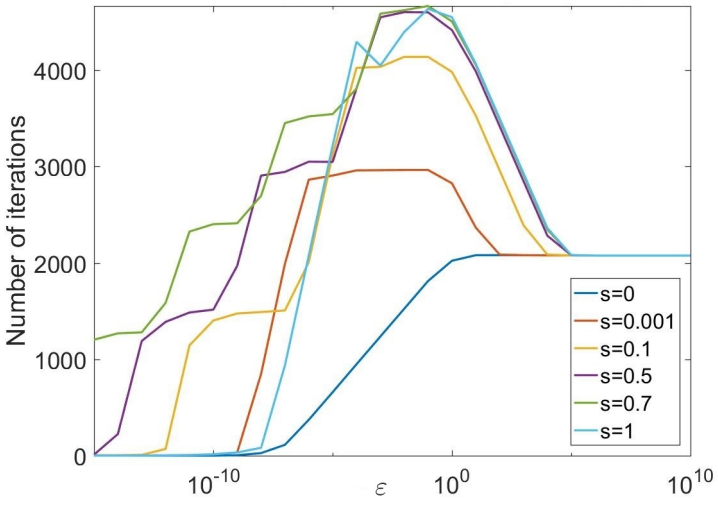


Figure 4.2: The number of iterations for different values of s and ε using the SD method.

We can see that the starting vector has a great influence in the methods performance. Therefore we will perform runs with a lot of starting vectors in the upcoming numerical experiments.

4.2 Harmonic step length

Another idea is to choose the step length such that the gradient of the objective function is minimized instead of the objective function itself.

$$\alpha_k = \arg\min_{\alpha>0} \|\nabla f(x_k + \alpha p_k)\|.$$

Note that α_k also minimizes the following function:

$$\phi_k(\alpha) := \tfrac{1}{2}\|\nabla f(x_k + \alpha p_k)\|^2 = \tfrac{1}{2}\|A(x_k + \alpha p_k)\|^2.$$

Using the fact that A is symmetric we find the following expression for the derivative of ϕ_k:

$$\phi_k'(\alpha) = (x_k + \alpha p_k)^T A^2 p_k.$$

Setting this derivative to zero and using the fact that A is symmetric results in the following expression for the so-called *minimal gradient* or *harmonic step length* [2]:

$$\alpha_k^{\mathrm{H}} = \frac{p_k^T A p_k}{p_k^T A^2 p_k}. \tag{4.3}$$

Another (similar) idea is to take step lengths equal to the Rayleigh quotient of A^{-1} instead of the reciprocal of the Rayleigh quotient of A: $\alpha_k = \dfrac{u_k^T A^{-1} u_k}{u_k^T u_k}$. However, the computation of A^{-1} is often very expensive. If one substitutes $u_k = Ap_k$ to get rid of A^{-1} the result is again the harmonic step length. This step length is called harmonic, because the Rayleigh quotient of A^{-1} is a weighted average of eigenvalues of A^{-1}. This weighted average is also known as the *harmonic mean* [28] of the eigenvalues of A^{-1}. Next we will prove some useful properties for the harmonic method as well.

4.2.1 Orthogonality property

For the harmonic step length the search direction is not orthogonal to the previous step, but we have A-orthogonality.

Lemma 4.2.1. *For the harmonic step length (4.3) the search direction of the Steepest Descent method is A-orthogonal to the previous step, i.e., $p_k^T A p_{k+1} = 0$.*

Proof.

$$\begin{aligned}
p_k^T A p_{k+1} &= p_k^T A(-Ax_{k+1}) \\
&= p_k^T AA(-x_k - \alpha_k p_k) \\
&= p_k^T A(-Ax_k) - \alpha_k p_k^T A^2 p_k \\
&= p_k^T A p_k - \frac{p_k^T A p_k}{p_k^T A^2 p_k} p_k^T A^2 p_k \\
&= p_k^T A p_k - p_k^T A p_k \\
&= 0.
\end{aligned}$$

□

4.2.2 Convergence rate

To derive the convergence rate for the harmonic method, we need the following lemma as shown in [30].

Lemma 4.2.2. *If $A \in \mathbb{R}^{n\times n}$ is SPD, then for any $x \in \mathbb{R}^n$ the sequence $a_k = \dfrac{x^T A^{k-1} x}{x^T A^k x}$, with $k = 0, 1, \ldots$ is monotonically decreasing.*

Proof. To prove the lemma, we show that $a_{k+1} \leq a_k$.
Using that A is symmetric, that $k = \frac{k+1}{2} + \frac{k-1}{2}$ and by applying the Cauchy-Schwarz inequality the following holds:

$$(x^T A^k x)^2 = ((A^{(k+1)/2}x)^T (A^{(k-1)/2}x))^2 \leq \|A^{(k+1)/2}x\|^2 \|A^{(k-1)/2}x\|^2 = (x^T A^{k+1} x)(x^T A^{k-1} x).$$

This can be rewritten to

$$\frac{(x^T A^k x)}{(x^T A^{k+1} x)} \leq \frac{(x^T A^{k-1} x)}{(x^T A^k x)},$$

hence a_k is monotonically decreasing. □

Note that from Lemma 4.2.2 we can conclude the following:

$$\alpha_k^H \leq \alpha_k^{SD}.$$

We can show that the convergence rate in (4.2) also holds for the SD method with harmonic step length. The result follows from the fact that both A and A^2 are SPD, by applying Lemma 4.2.2 and finally by applying property (4.2) for the classical Steepest Descent method:

$$\begin{aligned} f(x_k + \alpha_k^H p_k) &= \tfrac{1}{2}(x_k + \alpha_k^H p_k)^T A (x_k + \alpha_k^H p_k) \\ &\leq \tfrac{1}{2}(x_k + \alpha_k^{SD} p_k)^T A (x_k + \alpha_k^{SD} p_k) \\ &= f(x_k + \alpha_k^{SD} p_k) \\ &\leq \frac{(\lambda_{\max} - \lambda_{\min})^2}{(\lambda_{\max} + \lambda_{\min})^2} f_k. \end{aligned}$$

4.3 Barzilai–Borwein step lengths

Considering the Cauchy step length (4.1) and harmonic step length (4.3) we can derive the step lengths for the BB method in a similar way following the analysis of Barzilai–Borwein [2]. Recall from (3.4) that the BB1 step length minimizes

$$\alpha_k^{\mathrm{BB1}} = \arg\min_{\alpha>0} \|\alpha^{-1} s_k - y_k\|,$$

where $s_k = x_k - x_{k-1}$ and $y_k = p_k - p_{k-1}$. This implies that α_k^{BB1} is the reciprocal of the minimizer of the function

$$\phi_k(\beta) := \tfrac{1}{2}\|\beta s_k - y_k\|^2.$$

We get the following expression for the derivative of ϕ_k w.r.t. β:

$$\phi_k'(\beta) := \beta s_k^T s_k - s_k^T y_k.$$

By setting this derivative to zero we get the following expression for the BB1 step length:

$$\alpha_k^{\mathrm{BB1}} = \frac{s_k^T s_k}{s_k^T y_k} = \frac{(x_k - x_{k-1})^T (x_k - x_{k-1})}{(x_k - x_{k-1})^T (p_k - p_{k-1})}. \tag{4.4}$$

Recall from (3.5) that the BB2 step length minimizes

$$\alpha_k^{\mathrm{BB2}} = \arg\min_{\alpha>0} \|s_k - \alpha y_k\|,$$

where $s_k = x_k - x_{k-1}$ and $y_k = p_k - p_{k-1}$. This implies that α_k^{BB2} also minimizes the function

$$\psi_k(\alpha) := \tfrac{1}{2} \|s_k - \alpha y_k\|^2.$$

We get the following expression for the derivative of ψ_k w.r.t. α:

$$\psi_k'(\alpha) := \alpha y_k^T y_k - s_k^T y_k.$$

By setting this derivative to zero we get the following expression for the BB2 step length:

$$\alpha_k^{\mathrm{BB2}} = \frac{s_k^T y_k}{y_k^T y_k} = \frac{(x_k - x_{k-1})^T (p_k - p_{k-1})}{(p_k - p_{k-1})^T (p_k - p_{k-1})} \tag{4.5}$$

By applying (3.1) on (4.4) and (4.5) and using $p_k = -Ax_k$ we get the following expressions for the Barzilai–Borwein step lengths in the quadratic case:

$$\alpha_k^{\mathrm{BB1}} = \frac{p_{k-1}^T p_{k-1}}{p_{k-1}^T A p_{k-1}}, \quad \alpha_k^{\mathrm{BB2}} = \frac{p_{k-1}^T A p_{k-1}}{p_{k-1}^T A^2 p_{k-1}}. \tag{4.6}$$

Here α_k^{BB1} is equal to the Cauchy step length (4.1) at iteration $k-1$ and α_k^{BB2} is equal to the harmonic step length (4.3) at iteration $k-1$. Note that by Lemma 4.2.2 we have that

$$\alpha_k^{\mathrm{BB1}} \geq \alpha_k^{\mathrm{BB2}}.$$

Dai and Liao [11] show that the BB methods applied to strictly convex quadratic problems have R-linear convergence. However, it is still not completely clear why these methods often perform better than the SD method.

4.3.1 Orthogonality property

For the Barzilai–Borwein method BB1 we would like to know if a similar orthogonality property holds. We will try to find a vector q depending only on p_k and p_{k-1} such that $q \perp p_{k+1}$.

Lemma 4.3.1. *Assume $A = diag(1, \ldots, n)$. Denote $p_k^{(i)}$ for the ith component of p_k. If $q_i = \dfrac{(p_{k-1}^{(i)})^2}{p_k^{(i)}}$, i=1,...,n, then $q \perp p_{k+1}$.*

Proof.

$$\begin{aligned} p_{k+1} &= -Ax_{k+1} \\ &= -Ax_k - \alpha_{k-1}Ap_k \\ &= p_k - \frac{p_{k-1}^T p_{k-1}}{p_{k-1}^T Ap_{k-1}} Ap_k \end{aligned}$$

Now we can multiply both sides of the equation with q^T:

$$\begin{aligned} q^T p_{k+1} &= q^T p_k - \frac{p_{k-1}^T p_{k-1}}{p_{k-1}^T Ap_{k-1}} q^T Ap_k \\ &= \sum_{i=1}^{n} \frac{(p_{k-1}^{(i)})^2}{p_k^{(i)}} p_k^{(i)} - \frac{\sum_{i=1}^{n}(p_{k-1}^{(i)})^2}{\sum_{i=1}^{n} i(p_{k-1}^{(i)})^2} \sum_{i=1}^{n} \frac{(p_{k-1}^{(i)})^2}{p_k^{(i)}} ip_k^{(i)} \\ &= \sum_{i=1}^{n}(p_{k-1}^{(i)})^2 - \frac{\sum_{i=1}^{n}(p_{k-1}^{(i)})^2}{\sum_{i=1}^{n} i(p_{k-1}^{(i)})^2} \sum_{i=1}^{n} i(p_{k-1}^{(i)})^2 \\ &= \sum_{i=1}^{n}(p_{k-1}^{(i)})^2 - \sum_{i=1}^{n}(p_{k-1}^{(i)})^2 \\ &= 0 \end{aligned}$$

□

We have not yet found an application for the orthogonality property from Lemma 4.3.1, hence we leave this open for possible future work on the behavior of the BB1 method.

Now that the SD, harmonic, BB1 and BB2 methods are introduced we can look at recent literature for some step length choices based on these methods.

4.4 Some recent step length choices based on SD and BB

Due to the great results many new step lengths have been developed that generalize the Barzilai–Borwein methods.

4.4.1 Steepest Descent with alignment

The Steepest Descent method performs its search asymptotically in the two-dimensional subspace generated by the eigenvectors corresponding to the smallest and largest eigenvalue as discussed in section 4.1.3. De Asmundis et al. [14] write about the spectral properties of Steepest Descent methods. In that article some classical convergence results are reviewed for the SD method, which is some kind of theoretical basis for the rest of the article.

One of the ideas was that if the search direction is aligned with an eigenvector direction of A, the convergence of the algorithm might be faster. The nice behavior of the BB method is often explained by saying that the nonmonotonicity of such methods produces an erratic path of $1/\alpha_k$ of the interior of the spectrum of A which causes the sequences $\{\mu_{k,i}\}_{k\in\mathbb{N}}$ to go to zero together. Here $\mu_{k,i}$ is the component of ∇f_k along q_i, which means $\nabla f_k = \sum_{i=1}^{n} \mu_{k,i} q_i$, where q_k is the eigenvector corresponding to eigenvalue λ_k.

In [14, Sec. 3] is shown that the sequence of step lengths from the exact line search provides an approximation to the sum of the extreme eigenvalues of the Hessian. The SDA method is introduced, which is the Steepest Descent with alignment. This method is aimed at aligning the search direction with the eigenvector direction corresponding to the smallest eigenvalue, which forces the algorithm into the one-dimensional subspace spanned by that eigenvector direction. In the SDA method step lengths of the form

$$\tilde{\alpha}_k = \left(\frac{1}{\alpha_k^{SD}} + \frac{1}{\alpha_{k-1}^{SD}} \right)^{-1}$$

are chosen at some selected iterations. The switch condition that decides which step length is chosen can be found in the SDA algorithm in [14].

In [14, Sec. 4] is shown that a gradient method where the step length is twice the classical Steepest Descent step length, will also end up in a one-dimensional subspace, but now spanned by the eigenvector associated with the largest eigenvalue.

4.4.2 Dai and Yuan step lengths

Yuan [34] introduces the so-called Yuan-formula for the new step length depending on the step length of both the current and previous iteration of the SD method:

$$\alpha_k^{\mathrm{Y}} = 2 \left(\sqrt{ \left(\frac{1}{\alpha_{k-1}^{\mathrm{SD}}} + \frac{1}{\alpha_k^{\mathrm{SD}}} \right)^2 - 4 \frac{1}{\alpha_{k-1}^{\mathrm{SD}} \alpha_k^{\mathrm{SD}}} + 4 \frac{\|p_k\|^2}{(\alpha_{k-1}^{\mathrm{SD}} \|p_{k-1}\|)^2} } + \frac{1}{\alpha_{k-1}^{\mathrm{SD}}} + \frac{1}{\alpha_k^{\mathrm{SD}}} \right).$$

In [12] several articles are summarized and a new method is introduced called the SDC method which alternates some SD iterates with some gradient iterates that use a constant step length computed through the Yuan formula:

$$\alpha_k^{\mathrm{SDC}} = \begin{cases} \alpha_k^{\mathrm{SD}} & \text{if } \mathrm{mod}(k, h+l) < h, \\ \alpha_s^{Y} & \text{otherwise, with } s = \max\{i \le k : \ \mathrm{mod}(i, h+l) = h\}, \end{cases}$$

where $h \geq 2$ and $l \geq 1$. In other words: h consecutive exact line searches are performed and then, using the last two SD step lengths, the Yuan step length is computed and applied in l consecutive gradient iterations. Dai and Yuan introduced a method based on this SDC method:

$$\alpha_k^{\mathrm{DY}} = \begin{cases} \alpha_k^{\mathrm{SDC}} & \text{if } \mathrm{mod}(k, h+l) < h \\ \alpha_k^{\mathrm{Y}} & \text{otherwise.} \end{cases}$$

This constant step length at some iterations is justified by its spectral properties, which speed up the convergence of the SD method, by forcing a selective elimination of the eigencomponents of the gradient, starting from the one relative to the eigenvector with largest eigenvalue and proceeding toward the eigencomponents associated with smaller eigenvalues. De Asmundis et al. [12] confirms by numerical results that the monotonicity is naturally satisfied if the number of SD iterates is not too small.

In [13] the regularizing behavior of the SDA and SDC methods is analyzed. Here is shown that the tendency of these methods to eliminate first the eigencomponents of the gradient corresponding to large singular values allows to reconstruct the most significant part of the solution. By numerical experiments performed on image restoration problems this behavior is confirmed.

4.4.3 Adaptive BarzilaiBorwein

Two methods we focus on will be the adaptive Barzilai–Borwein method (ABB) as originally formulated in [35] and a modification ($\mathrm{ABB}_{\min}$) as formulated in [18]. The step lengths are defined as follows:

$$\alpha_k^{\mathrm{ABB}} = \begin{cases} \alpha_k^{\mathrm{BB2}} & \text{if} \quad \alpha_k^{\mathrm{BB2}} < \tau\, \alpha_k^{\mathrm{BB1}} \\ \alpha_k^{\mathrm{BB1}} & \text{otherwise,} \end{cases}$$

$$\alpha_k^{\mathrm{ABB}_{\min}} = \begin{cases} \min\{\alpha_j^{\mathrm{BB2}} : j = \max\{1, k-m\}, \ldots, k\} & \text{if} \quad \alpha_k^{\mathrm{BB2}} < \tau\, \alpha_k^{\mathrm{BB1}} \\ \alpha_k^{\mathrm{BB1}} & \text{otherwise,} \end{cases}$$

where m is a nonnegative integer and $\tau \in (0, 1)$. The general idea behind this method is that it adaptively chooses a small step length or a large step length at each iteration. A small step length is used to induce a favorable descent direction for the next iteration, while the large step length is used to produce a sufficient reduction. This method is derived by Zhou, Gao and Dai [35] by looking at the worst-case behavior of both the SD and harmonic methods.

They show that if the SD method has the slowest convergence rate then $\alpha_k^{\mathrm{H}} > 0.8 \cdot \alpha_k^{\mathrm{SD}}$. Due to the sign difference this is identical to the condition $\alpha_k^{\mathrm{BB2}} < 0.8 \cdot \alpha_k^{\mathrm{BB1}}$. Therefore it seems preferable to use the harmonic step length if this is the case.

The harmonic step length is chosen to avoid the worst-case behavior of the SD method. Furthermore, the worst-case behavior of the harmonic method is usually caused by very small step lengths and not by the descent direction. Therefore it is possible to avoid the worst-case behavior of the harmonic method for very ill-conditioned problems.

The parameter m in ABB_{min} is called a *memory parameter* since this method uses information of the past m iterations. In [18] some numerical results indicate that ABB_{min} often performs better than the BB1 and BB2 methods because of this information.

In the past sections we have discussed some useful properties of the Cauchy, harmonic, Barzilai–Borwein and some other recent step length choices. One important discovery is the relation to the spectral properties of the Hessian. By observing the effect of a memory parameter and the research by Dai and Yuan discussed in section 4.4.2 on spectral properties of the Hessian we can understand the motivation to research limited memory methods. This research has been very popular the past few years, which we will elaborate on some more in the next chapters.

CHAPTER 5

Limited memory methods

Di Serafino et al. [32] write about recent analysis by Fletcher [16] on properties of these limited memory gradient methods. The analysis focuses mainly on the quadratic example from the previous chapter. We follow that analysis to describe some limited memory methods.

5.1 Ritz values

Let $\{\lambda_1, \ldots, \lambda_n\}$ be the eigenvalues of $A = Q\Lambda Q^T$ with corresponding eigenvectors $\{q_1, \ldots, q_n\}$ (columns of Q). The following lemmas are useful results about the search direction of gradient methods.

Lemma 5.1.1. *The gradient p_k can be expressed as a linear combination of $\{q_i\}_{i\in[n]}$ [16]:*

$$p_k = \sum_{i=1}^{n} \mu_{k,i} q_i, \text{ with } \mu_{k,i} \in \mathbb{R}, \tag{5.1}$$

where the weights satisfy $\mu_{k+1,i} = (1 - \alpha_k \lambda_i)\mu_{k,i}$.

Proof. The columns of Q form an orthogonal basis of $\mathbb{R}^n$. Therefore we can write p_k as a linear combination of the columns of Q:

$$p_k = \sum_{i=1}^{n} \mu_{k,i} q_i, \text{ with } \mu_{k,i} \in \mathbb{R}.$$

From the recursive equation of p_k we get the following:

$$p_{k+1} = p_k + \alpha_k A p_k = (I + \alpha_k A) p_k.$$

If we write all search directions in this equation as linear combinations of the columns of Q we get the following relation:

$$\sum_{i=1}^{n} \mu_{k+1,i} q_i = \sum_{i=1}^{n} \mu_{k,i}(I + \alpha_k A) q_i.$$

Now we can use this to prove the recursive property of the coefficients:

$$\sum_{i=1}^{n} \mu_{k+1,i} q_i = \sum_{i=1}^{n} \mu_{k,i}(I + \alpha_k A) q_i = \sum_{i=1}^{n} \mu_{k,i}(q_i + \alpha_k \lambda_i q_i) = \sum_{i=1}^{n} \mu_{k,i}(1 + \alpha_k \lambda_i) q_i.$$

Here we used that λ_i is an eigenvalue of A with corresponding eigenvector q_i. The result follows from the fact that the columns of Q form an orthogonal basis of $\mathbb{R}^n$. □

Lemma 5.1.2. *For any gradient method we have the following properties [16]:*

(i) $p_k = \prod_{j=1}^{k-1}(I - \alpha_j A) p_1,$

(ii) $\mu_{k,i} = \mu_{k-1,i}(1 - \alpha_{k-1}\lambda_i).$

Proof. For any gradient method we have that

$$p_k = p_{k-1} - \alpha_{k-1} A p_{k-1} = (I - \alpha_{k-1} A) p_{k-1}.$$

Now we can use this to retrieve the following:

$$\begin{aligned} p_k &= (I - \alpha_{k-1}A)p_{k-1} \\ &= (I - \alpha_{k-1}A)(p_{k-2} - \alpha_{k-2}Ap_{k-2}) \\ &= (I - \alpha_{k-1}A)(I - \alpha_{k-2}A)p_{k-2} \\ &= \dots \\ &= \prod_{j=1}^{k-1}(I - \alpha_j A) p_1. \end{aligned}$$

Hence we have (i).
Next we use (5.1) to get the following:

$$\begin{aligned} \sum_{i=1}^{n} \mu_{k,i} q_i &= p_k \\ &= p_{k-1} - \alpha_{k-1} A p_{k-1} \\ &= \sum_{i=1}^{n} \mu_{k-1,i} q_i - \alpha_{k-1} A \sum_{i=1}^{n} \mu_{k-1,i} q_i \\ &= \sum_{i=1}^{n} \mu_{k-1,i} q_i - \alpha_{k-1} \sum_{i=1}^{n} \mu_{k-1,i} \lambda_i q_i \\ &= \sum_{i=1}^{n} (\mu_{k-1,i} - \alpha_{k-1} \mu_{k-1,i} \lambda_i) q_i \end{aligned}$$

Result (ii) follows directly from the fact that the coefficients of both sums must be equal since the columns of Q form an orthogonal basis of $\mathbb{R}^n$. □

Note that because of the commutating property in Lemma 5.1.2(i), the search direction does not depend on the order of the step lengths.

Proposition 5.1.3. *A direct consequence of Lemma 5.1.2 are the following properties [16]:*

1. *if $\mu_{k,i} = 0$ for some i, then $\mu_{h,i} = 0$ for $h \geq k$;*
2. *if $\alpha_k = 1/\lambda_i$, then $\mu_{k+1,i} = 0$;*
3. *$|\mu_{k+1,i}| < |\mu_{k,i}|$ if and only if $\alpha_k < 2/\lambda_i$;*
4. *if α_k is sufficiently close to $1/\lambda_j$, then $|\mu_{k+1,i}| > |\mu_{k,i}|$ for $i < j$ and $\lambda_i > 2\lambda_j$.*

Fletcher [16] concludes from these properties that small step lengths (close to $1/\lambda_{\max}$) tend to decrease a large number of eigencomponents, with negligible reduction of those corresponding to small eigenvalues. However, the reduction of those corresponding to small eigenvalues can also be achieved by choosing larger step lengths (close to $1/\lambda_{\min}$), but this might cause a nonmonotonic behavior of the sequence $\{f_k\}$.

A property described by Fletcher [16] that is possessed by all steepest descent methods is:

$$x_k - x_{k-m} \in \langle p_{k-m}, Ap_{k-m}, A^2p_{k-m}, \ldots, A^{m-1}p_{k-m}\rangle. \tag{5.2}$$

Hence the displacement of the current state x_k from any back value x_{k-m} lies in the span of the Krylov sequence initiated by p_{k-m}. See for example Fletcher [16] to conclude the following property:

$$p_k - p_{k-m} \in \langle Ap_{k-m}, A^2p_{k-m}, A^3p_{k-m}, \ldots, A^mp_{k-m}\rangle. \tag{5.3}$$

Paige et al. [28] use this Krylov sequence to provide m distinct estimates of the eigenvalues of A by first constructing an orthogonal basis followed by a so-called *Ritz basis* by running the Lanczos iterative process [25] for m iterations, starting from p_{k-m}. These estimates are called *Ritz values*. We will use this concept to describe the Limited Memory Steepest Descent method.

5.2 Limited Memory Steepest Descent (LMSD)

By the above analysis Fletcher [16] introduces a new method called the *Limited Memory Steepest Descent (LMSD)*. To fashion a limited memory method, on any iteration the recent m back values are also available in storage, where m is limited to an upper bound $\bar{m}$ with the assumption $\bar{m} \ll n$. The LMSD-method operates in cycles of m iterations, where $x_{k,1}$ represents the initial point of the kth cycle. Curtis [8] uses a notation with double indices (k, j) to indicate the jth iteration at the kth cycle. We will use this notation as well. A sequence of at most m step lengths $\{\alpha_{k,j}\}_{j\in[m]}$ is computed at the beginning of each cycle. We get the following relation for the search directions:

$$p_{k,j+1} = p_{k,j} - \alpha_{k,j} A p_{k,j}, \quad (k, j) \in \mathbb{N} \times [m].$$

The result at the end of a cycle is set as the initial point for cycle $k + 1$, i.e.,

$$p_{k+1,1} = p_{k,m+1}.$$

The iterate update in the LMSD algorithm is the standard update in a Barzilai–Borwein method where the step lengths are chosen as the reciprocals of the Ritz values of A.

For obtaining these Ritz values define the matrix $P_k \in \mathbb{R}^{n\times m}$ as

$$P_k = [p_{k,j-m}, \ldots, p_{k,j}], \tag{5.4}$$

where we define indices $(k, -1) := (k - 1, m), (k, -2) := (k - 1, m - 1)$ and so forth if $j < m$. Fletcher [16] describes several ways to obtain these Ritz values. The most intuitive way is by estimating the Ritz values $\theta_{k,j}$ as the eigenvalues of $T_k = Q_k^T A Q_k$. The tridiagonal matrix $T_k \in \mathbb{R}^{m\times m}$ is the result of running the Lanczos algorithm for m iterations on the SPD matrix A. Hence Q_k is orthogonal.

Lemma 5.2.1. *Let U be any nonsingular matrix. If λ is an eigenvalue of $U^{-1}AU$ with corresponding eigenvector x, then λ is an eigenvalue of A with corresponding eigenvector $y = Ux$.*

Proof. Since λ is an eigenvalue of $V = U^{-1}AU$ with corresponding eigenvector x, we have that $Vx = \lambda x$. Also, since U is invertible we have $U^{-1}U = UU^{-1} = I$ and thus $A = UVU^{-1}$. Now the result follows directly:

$$Ay = A(Ux) = UVU^{-1}Ux = U(Vx) = U(\lambda x) = \lambda Ux = \lambda y.$$

Hence λ is an eigenvalue of A with corresponding eigenvector $y = Ux$. □

A consequence of Lemma 5.2.1 is that if λ is an eigenvalue of $T_k = Q_k^T A Q_k$ with corresponding eigenvector x, it is also an eigenvalue of A with corresponding eigenvector Qx.
Thus the Lanczos algorithm transforms the eigendecomposition problem for A into the eigendecomposition problem for T_k. Note that also matrix Q_k can be avoided by using the fact that $Q_k R_k$ is the reduced QR-decomposition of P_k. Hence $T_k = (P_k R_k^{-1})^T A P_k R_k^{-1}$.
The step lengths for the next m gradient iterations are defined as the reciprocals of these eigenvalues $\theta_{k,j}$ of T_k:

$$\alpha_{k,j}^{\text{LMSD}} = \frac{1}{\theta_{k-1,j}}, \quad j \in [m], \tag{5.5}$$

such that the step lengths are ordered by size starting with the smallest step length, as proposed in [16]. Note that the smallest step length corresponds to the largest Ritz value. The reason for this sorting is that the large step lengths are applied after some iterations in which smaller step lengths have reduced the eigencomponents of the gradient corresponding to large eigenvalues. Curtis' analysis [8] confirms that the practical performance of the method can be improved if one has the knowledge to choose one or more step lengths exactly equal to reciprocals of eigenvalues of A. This is a very important remark which we will elaborate on some more at the numerical experiments.

As described in the previous section, m iterations should have been performed before a new sweep of Ritz values can be computed. How to choose the initial step lengths has no effect on the theoretical results. This is shown in [8]. Many strategies for the startup of this algorithm are possible. Di Serafino et al. [32] initialize the LMSD method with a single step length equal to 1, which indicates a sweep of length one. This can be used to define two step lengths allowing two new iterations. This can be continued until there are enough back gradients to start a sweep of length m. Fletcher [16] uses the strategy to specify the first m Ritz values as equally spaced values in the interior of the spectrum. Note that the LMSD method for $m = 1$ is the same as the BB1 method, so another strategy is to construct the first cycle of Ritz values of the first m iterations by applying the BB1 method:

$$\alpha_{1,j}^{\text{LMSD}} = \begin{cases} \dfrac{p_{1,j}^T p_{1,j}}{p_{1,j}^T A p_{1,j}}, & j = 1, \\ \dfrac{p_{1,j-1}^T p_{1,j-1}}{p_{1,j-1}^T A p_{1,j-1}}, & j = 2, \dots m. \end{cases}$$

In Algorithm 3 this last strategy is implemented.

Algorithm 3: Limited Memory Steepest Descent for $f(x) = \frac{1}{2}x^TAx$

Input: $x_{1,1} \in \mathbb{R}^n$, $m \in \mathbb{N} : m \ll n$, $\texttt{tol} > 0$, $f : \mathbb{R}^n \to \mathbb{R}$ given by $f(x) = \frac{1}{2}x^TAx$, where A is SPD.
Output: Approximate solution to $\arg\min_{x \in \mathbb{R}^n} f(x)$

$p_{1,1} = -\nabla f_{1,1}$
if $\|p_{1,1}\| < \texttt{tol}$
  return $x_{1,1}$
end if
for $k = 1, 2, \ldots$
  for $j \in [m]$
    Update P_k according to (5.4)
    if $k = 1$
      if $j = 1$
        $\alpha_{k,j} = \alpha_{k,j}^{\mathrm{SD}}$
      else
        $\alpha_{k,j} = \alpha_{k,j}^{\mathrm{BB1}}$
      end if/else
    end if
    $x_{k,j+1} = x_{k,j} + \alpha_{k,j}p_{k,j}$
    $p_{k,j+1} = -\nabla f_{k,j+1}$
    if $\|p_{k,j+1}\| < \texttt{tol}$
      return $x_{k,j+1}$
    end if
  end for
  $x_{k+1,1} = x_{k,m+1}$
  $p_{k+1,1} = p_{k,m+1}$
  Calculate R_k by reduced QR-decomposition of P_k.
  If R_k is too ill-conditioned, update P_k and R_k by removing oldest back gradient
  Calculate $\{\theta_{k,j}\}_{j\in[m]}$ as the eigenvalues of $T_k = (P_kR_k^{-1})^TAP_kR_k^{-1}$ in decreasing order
  $\{\alpha_{k+1,j}\}_{j\in[m]} = \{\theta_{k,j}^{-1}\}_{j\in[m]}$
end for

Fletcher [16] discusses the value of the memory parameter m. If the value m is increased, the back gradients can approach linear dependence. Because of this, matrix R_k can become increasingly ill-conditioned and hence numerical issues restrict the effectiveness of the method. Fletcher suggests an upper bound of $\bar{m} = 5, 6$ or 7.
Despite this limit on m, one problem that still might occur is that matrix R_k becomes ill-conditioned. In this case the oldest column(s) of P_k are removed and fewer than m step lengths are provided for the next sweep. This also implies that back gradients from the previous sweep are kept for defining the m columns for the next matrix P_k. Note furthermore that the eigenvalues of T_k are ordered in decreasing order. This implies that the step lengths are used in increasing order.

5.3 Harmonic Limited Memory Steepest Descent (HLMSD)

Fletcher [16] also introduces an alternative limit memory method by replacing the Ritz values of A by so-called *harmonic Ritz values* of A. Recall that in the LMSD method, the step lengths are chosen as the reciprocals of the eigenvalues of $Q_k^T A Q_k$.
For the harmonic case the step lengths are chosen as the reciprocals of the eigenvalues of $(Q^T A^2 Q, Q^T A Q)$, i.e.,

$$\alpha_{k,j}^{\text{HLMSD}} = \frac{1}{\widehat{\theta}_{k-1,j}}, \quad j = 1, \ldots, m. \tag{5.6}$$

Here $\widehat{\theta}_{k,j}$ is determined by the condition $Q_k^T A^2 Q_k x = \widehat{\theta}_{k,j} Q_k^T A Q_k x$, for some $x \in \mathbb{R}^n$.

The HLMSD method is similar to Algorithm 3 with the adjustments described in this section. Note that for $m = 1$ the HLMSD method is the same as the BB2 method. Therefore the BB2 method is used as a startup method to generate the first m gradients required to calculate the first set of harmonic Ritz values.

5.4 Adaptive Limited Memory Steepest Descent (ALMSD)

In the ABB method from section 4.4.3 a combination of both the BB1 and the BB2 method is used. ABB makes sure that the worst-case scenarios of BB1 and BB2 are avoided. Based on that strategy the same might work for the LMSD and HLMSD method.

We introduce a new method by choosing a sweep of Ritz values from either the LMSD or the HLMSD method, following the same rules as in the ABB method. To determine which sweep to use, the rules in ABB are used for the first step length in both methods (the smallest step length of the upcoming sweep). Hence we choose a sweep of harmonic Ritz values if $\min\limits_{j\in[m]} \alpha_{k,j}^{\text{HLMSD}} < \tau \min\limits_{j\in[m]} \alpha_{k,j}^{\text{LMSD}}$ and a sweep of normal Ritz values otherwise. We call this method the Adaptive LMSD method (ALMSD). This results in the following formula for the upcoming sweep of step lengths with $j \leq m$:

$$\alpha_{k,j}^{\text{ALMSD}} = \begin{cases} \alpha_{k,j}^{\text{HLMSD}} & \text{if } \min\limits_{j\in[m]} \alpha_{k,j}^{\text{HLMSD}} < \tau \min\limits_{j\in[m]} \alpha_{k,j}^{\text{LMSD}} \\ \alpha_{k,j}^{\text{LMSD}} & \text{otherwise.} \end{cases}$$

Note that for $m = 1$ this method is identical to the ABB method. Therefore the ABB method is used as a startup method to generate the first m gradients required to calculate the first set of (harmonic) Ritz values.

In the next section we will prove some useful properties of the LMSD, HLMSD and ALMSD methods.

5.5 Properties of the limited memory methods

In this section several properties of the limited memory methods are discussed. For example convergence properties, monotonic behavior and properties of the Ritz values are important to gain insight in their performance.

5.5.1 R-linear convergence

Note that Q-linear convergence is stronger than R-linear convergence. Here we will sketch the proof of R-linear convergence of the sequence $\{f_k\}_{k\in\mathbb{N}}$ in the LMSD method. We use the proof of Curtis [8] as a guideline, where it is assumed that $\lambda_n \geq \lambda_{n-1} \geq \cdots \geq \lambda_1$. First we prove a lemma concerning the interlacing property of the Ritz values.

Lemma 5.5.1. *For all $k \in \mathbb{N}$, the eigenvalues $\theta_{k,j}$ of T_k satisfy:*

$$\theta_{k,j} \in [\lambda_{m+1-j}, \lambda_{n+1-j}], \quad j \in [m],$$

where $\theta_{k,1} \geq \theta_{k,2} \geq \cdots \geq \theta_{k,m}$.

Proof. The result follows directly from the *Cauchy interlacing theorem* [21] applied to $T_k = Q_k^T A Q_k$. □

In the following lemma (similar to [8, Lem. 3.7]) we introduce some useful notation to write the Ritz values depending on the eigenvalues of A.

Lemma 5.5.2. *For all $(k, j) \in \mathbb{N} \times [m]$, let $q_{k,j} \in \mathbb{R}^m$ denote the unit eigenvector corresponding to the eigenvalue $\theta_{k,j}$ of T_k. Using the weights of (5.1), define:*

$$M_k := \begin{bmatrix} \mu_{k,1,1} & \mu_{k,2,1} & \cdots & \mu_{k,m,1} \\ \mu_{k,1,2} & \mu_{k,2,2} & \cdots & \mu_{k,m,2} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{k,1,n} & \mu_{k,2,n} & \cdots & \mu_{k,m,n} \end{bmatrix}, \quad c_{k,j} := M_k R_k^1 q_{k,j}.$$

It follows that $\theta_{k,j} = c_{k,j}^T \Lambda c_{k,j}$ and $c_{k,j}^T c_{k,j} = 1$.

Proof. First write (5.1) as an equation of matrices: $P_k = Q_k M_k$. Use this and $P_k = Q_k R_k$ to get:

$$T_k = Q_k^T A Q_k = R_k^{-T} M_k^T Q_k^T A Q_k M_k R_k^{-1} = R_k^{-T} M_k^T \Lambda M_k R_k^{-1}.$$

Now we get the following for the Ritz values for all $j \in [m]$:

$$\theta_{k,j} = q_{k,j}^T T_k q_{k,j} = q_{k,j}^T R_k^{-T} M_k^T \Lambda M_k R_k^{-1} q_{k,j} = c_{k,j}^T \Lambda c_{k,j}.$$

Next we use this and $Q_k^T Q_k = I$ to get $c_{k,j}^T c_{k,j} = 1$. □

In the next lemma (similar to [8, Lem. 3.8]) we prove a recursive bound for the weights in (5.1) followed by a recursive bound for the gradient.

Lemma 5.5.3. $\forall(k,j,i) \in \mathbb{N} \times [m] \times [n]$*:*

(i) $\left|\mu_{k,j+1,i}\right| \leq \delta_{j,i}\left|\mu_{k,j,i}\right|$*, where* $\delta_{j,i} = \max\left\{\left|1 - \frac{\lambda_i}{\lambda_{m+1-j}}\right|, \left|1 - \frac{\lambda_i}{\lambda_{n+1-j}}\right|\right\}$*;*

(ii) $\left|\mu_{k+1,j,i}\right| \leq \Delta_i\left|\mu_{k,j,i}\right|$*, where* $\Delta_i = \prod_{j=1}^{m} \delta_{j,i}$*;*

(iii) $\sum_{i=1}^{p} \mu_{k,j+1,i}^2 \leq \max_{i\in[p]}\{\delta_{j,i}^2\} \sum_{i=1}^{p} \mu_{k,j,i}^2$*;*

(iv) $\forall(k,j) \in \mathbb{N} \times [m]$*:* $\|p_{k+1,j}\| \leq \Delta \|p_{k,j}\|$*, where* $\Delta = \max_{i\in[n]} \Delta_i$*.*

Proof. Lemma 5.5.1 implies that

$$\alpha_{k,j} \in \left[\frac{1}{\lambda_{n+1-j}}, \frac{1}{\lambda_{m+1-j}}\right] \subseteq \left[\frac{1}{\lambda_n}, \frac{1}{\lambda_1}\right], \quad \forall(k,j) \in \mathbb{N} \times [m].$$

By using this and $\mu_{k,j+1,i} = (1 - \alpha_{k,j}\lambda_i)\mu_{k,j,i}$ from (5.1) we get (i). The second result follows from (i) and step $p_{k+1,1} = p_{k,m+1}$ from the LMSD algorithm. From (i) we get

$$\sum_{i=1}^{p} \mu_{k,j+1,i}^2 \leq \sum_{i=1}^{p} \delta_{j,i}^2 \mu_{k,j,i}^2 \leq \max_{i\in[p]}\{\delta_{j,i}^2\} \sum_{i=1}^{p} \mu_{k,j,i}^2,$$

which results in (iii). Lemma 5.5.1 and (5.1) imply

$$\|p_{k,j}\|^2 = \sum_{i=1}^{n} \mu_{k,j,i}, \quad \forall(k,j) \in \mathbb{N} \times [m].$$

This and (ii) imply

$$\|p_{k+1,j}\|^2 = \sum_{i=1}^{n} \mu_{k+1,j,i} \leq \sum_{i=1}^{n} \Delta_i^2 \mu_{k,j,i}^2 \leq \Delta^2 \sum_{i=1}^{n} \mu_{k,j,i}^2 = \Delta^2 \|p_{k,j}\|^2.$$

Hence (iv) is satisfied. □

Observe from Lemma 5.5.3(iv) that under the assumption $\Delta_i \in [0,1)$ $\forall i \in [n]$ we even have Q-linear convergence. The remainder of the proof for R-linear convergence uses the results of the previous lemmas to construct an upper bound for the weights $\mu_{k,j,i}$ depending on the spectrum of A and the norm of $p_{k,1}$. For this we refer to [8, Lem. 3.9, 3.10, 3.11]. These bounds from Curtis can then be used to proof the following lemma as shown in the proof of [8, Lem. 3.12].

Lemma 5.5.4. *There exists* $K \in \mathbb{N}$ *dependent only on the spectrum of* A *such that* $\|p_{k+K,1}\| \leq \frac{1}{2}\|p_{k,1}\|$ *for all* $k \in \mathbb{N}$*.*

This lemma and previous properties can now be used to proof the R-linear convergence property in the following theorem (with [8, Thm. 3.13] as a guideline).

Theorem 5.5.5. *The sequence $\{\|p_{k,1}\|\}_{k\in\mathbb{N}}$ in the LMSD method vanishes R-linearly.*

Proof. Since we already know that if $\Delta \in [0,1)$ we have Q-linear convergence, we only have to prove R-linear convergence for $\Delta \geq 1$. By using the result of Lemma 5.5.4, we have that there exists a $K \in \mathbb{N}$ such that

$$\|p_{1+Kl,1}\| \leq \tfrac{1}{2} \|p_{1+K(l-1),1}\| \text{ for all } l \in \mathbb{N}.$$

Recursively it follows that:

$$\|p_{1+Kl,1}\| \leq (\tfrac{1}{2})^l \|p_{1,1}\| \text{ for all } l \in \mathbb{N}. \tag{5.7}$$

For any $k \geq 1$, write $k = Kl + s$ for some $l \in \mathbb{N} \cup \{0\}$ and $s \in [K-1] \cup \{0\}$. We can apply lemma 5.5.3(iv) $s-1$ times and use $\Delta \geq 1$ and $s \leq K$ to get the following estimate:

$$\|p_{k,1}\| \leq \Delta \|p_{k-1,1}\| \leq \cdots \leq \Delta^{s-1} \|p_{k-s+1,1}\| \leq \Delta^{K-1} \|p_{1+Kl,1}\|.$$

Now we can use this result, (5.7) and $l = \dfrac{k}{K} - \dfrac{s}{K} \geq \dfrac{k}{K} - 1$ to get the following estimate:

$$\begin{aligned}
\|p_{k,1}\| &\leq \Delta^{K-1} \|p_{1+Kl,1}\| \\
&\leq \Delta^{K-1} (\tfrac{1}{2})^l \|p_{1,1}\| \\
&\leq \Delta^{K-1} (\tfrac{1}{2})^{k/K-1} \|p_{1,1}\| \\
&= c_1 c_2^k \left\|p_{1,1}\right\|,
\end{aligned}$$

where $c_1 = 2\Delta^{K-1}$ and $c_2 = 2^{-1/K}$. Observe that $c_2 \in (0,1)$ to conclude that $\{c_1 c_2^k \|p_{1,1}\|\}_{k\in\mathbb{N}}$ vanishes Q-linearly and hence $\{\|p_{k,1}\|\}_{k\in\mathbb{N}}$ vanishes R-linearly. □

Now that we have R-linear convergence for the LMSD method, we would like to achieve the same result for the LMSD method with harmonic Ritz values. First we must show that the Ritz values of this HLMSD method satisfy the same properties as shown for the Ritz values of the LMSD method. We follow the proof of [8, Lem. A.1] as a guideline.

Lemma 5.5.6. *For all $k \in \mathbb{N}$, the eigenvalues $\widehat{\theta}_{k,j}$ of $(Q_k^T A^2 Q_k, Q_k^T A Q_k)$ satisfy:*

$$\widehat{\theta}_{k,j} \in [\lambda_{m+1-j}, \lambda_{n+1-j}], \quad j \in [m],$$

where $\widehat{\theta}_{k,1} \geq \widehat{\theta}_{k,2} \geq \cdots \geq \widehat{\theta}_{k,m}$.

Proof. Beattie [3, Thm. 2.1] uses the *min-max theorem for eigenvalues* with $K = A$, $M = I$, and $P = Q_k$ to prove this lemma. Note that the Ritz values are labeled differently in this proof. □

Using Lemma 5.5.6 and Lemma 5.5.3 we can see that [8, Lem. 3.9] also holds for the HLMSD method. To prove [8, Lem. 3.10, 3.11] for the HLMSD method a slight addition is required to the proofs. These modifications are described on [8, p. 23]. Using the result of [8, Lem. 3.11] for the HLMSD method, the proofs of Lemma 5.5.4 and Theorem 5.5.5 follow without modifications. Hence, the HLMSD method also converges R-linearly.

The R-linear convergence of the ALMSD method follows directly from the R-linear convergence of the LMSD and HLMSD methods, since the sequence $\{\|p_{k,1}\|\}_{k\in\mathbb{N}}$ can be split in two subsequences of which one vanishes R-linearly by the proof for LMSD and one vanishes by the proof for HLMSD.

5.5.2 Monotonic behavior

As described in chapter 3, monotonic behavior is often desired to improve convergence. Therefore, it is interesting to investigate the monotonic behavior of the methods described in the previous sections. The Steepest Descent method is monotonic by construction, but the Barzilai–Borwein methods are nonmonotonic. Both the LMSD method and HLMSD method are also nonmonotonic and therefore also ALMSD is nonmonotonic. A question that comes to mind is 'how often does a method display nonmonotonic behavior?'.
For the quadratic example with $n = 10^3$ and a random start vector we have that the $\text{ABB}_{\min}$ method is nonmonotonic in 1.66% of the iterations and the LMSD method is nonmonotonic in 6.8% of the sweeps. The HLMSD method is nonmonotonic in 12.7% of the sweeps and the ALMSD method is nonmonotonic in 16.6% of the sweeps. These percentages are an average over 1000 different starting vectors on the unit sphere.

It is possible to force monotonicity by applying a monotonic method for one iteration if monotonicity is not satisfied in that iteration. For the limited memory methods we can calculate new (harmonic) Ritz values after the single iteration and continue the current method. However, this is not preferable if it happens often. In section 5.6 we will perform numerical experiments on other problems to investigate the monotonic behavior as well.

5.6 Numerical results for the quadratic example

In this section we discuss numerical results of the described methods and step length choices from the previous sections for the quadratic case.

We consider minimization problem (1.1) with objective function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) = \tfrac{1}{2}x^T Ax,$$

with A an SPD matrix. The parameter values of these methods are based on the literature. We set $n = 10^3$ and calculate the average amount of iterates for each method over the same 1000 random starting vectors on the unit sphere. We set $\texttt{tol} = 10^{-6}$ and the program is interrupted if the stopping criterion is not satisfied within 10^4 iterations. All numerical results are computed by running MATLAB implementations of the following methods:

- Steepest Descent with step length (4.1)
- Harmonic/Minimal Gradient with step length (4.3)
- Barzilai–Borwein with BB1 step length (4.6)
- Barzilai–Borwein with BB2 step length (4.6)
- Adaptive Barzilai–Borwein with $\tau = 0.8$ as described in section 4.4.3.
- Adaptive Barzilai–Borwein with minimum condition with $\tau = 0.8$ and $m = 5$ as described in section 4.4.3
- Yuan method as described in section 4.4.2
- Steepest Descent with alignment as described in section 4.4.1
- Steepest Descent with some constant Yuan iterates, with $h = 2$ and $l = 2$ as described in section 4.4.2
- Limited Memory Steepest Descent as described in section 5.2
- Harmonic Limited Memory Steepest Descent as described in section 5.3
- Adaptive Limited Memory Steepest Descent as described in section 5.4

For the limited memory methods we set $m = 3$ and $m = 5$.

In Table 5.1 numerical results are shown for the described methods applied to the quadratic example with matrix A given by $A = \text{spdiag}(1, \ldots, n)$. In column $\#f$ the amount of function evaluations is shown and in column nm-$\#f$ the amount of function evaluations is shown at which nonmonotonic behavior is registered.

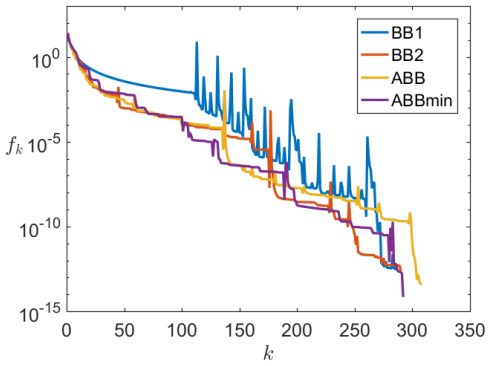
Table 5.1: Numerical results for the quadratic case

Method	$\#f$	nm-$\#f$
SD (exact)	4994	0
harmonic	4849	0
BB1	310	12
BB2	314	2
ABB	284	2
ABB_{min}	268	1
Yuan	274	0
SDA	291	0
SDC	283	0
LMSD ($m = 3$)	311	16
HLMSD ($m = 3$)	313	3
ALMSD ($m = 3$)	306	15
LMSD ($m = 5$)	288	10
HLMSD ($m = 5$)	290	6
ALMSD ($m = 5$)	288	10

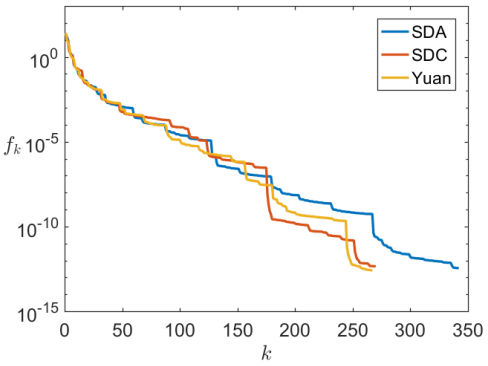
For $m = 5$, ALMSD performed on average 57 sweeps of LMSD step lengths and 0 sweeps of HLMSD step lengths, which explains the same results as LMSD. However, for $m = 3$, ALMSD performed on average 101 sweeps of LMSD step lengths and 1 sweep of HLMSD step lengths. Note that the one HLMSD sweep did actually result in a slight improvement. We will see in section 6.5.3 that ALMSD performs more HLMSD sweeps on other problems. In Figures 5.1 and 5.2 function and gradient evaluations are plotted for all methods starting with the normalized starting vector $x_1 = \dfrac{(n, n-1, \ldots, 1)^T}{\|(n, n-1, \ldots, 1)\|}$. This choice for the starting vector is inspired by the starting vector choices in [4]. Its angle with the gradient is not close to zero. In these figures one can see the (non)monotonic behavior of each method and the convergence speed. We can see that the limited memory methods oscillate more than the other methods.

In Figure 5.3 the reciprocals of the step lengths are plotted for every iteration for the quadratic function with $A = \text{spdiag}(1, 2, \ldots, 10, 31, 32, \ldots, 40)$. The gray lines indicate the eigenvalues of A. The LMSD methods provide Ritz values at each sweep, which attempt to approximate the eigenvalues of A to reduce the corresponding eigencomponents. In Figure 5.3 we observe that methods such as SDC, ABB_{min}, LMSD, HLMSD and ALMSD are better in describing the actual distribution of the eigenvalues than methods such as BB1, BB2, ABB, SDA and Yuan.

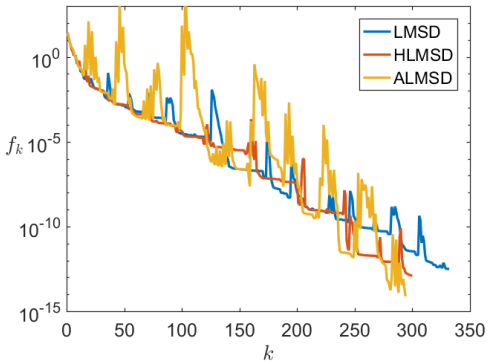
In this chapter we have introduced the limited memory methods for the quadratic case and shown some useful properties. Also, some observations were made concerning the eigenvalues of the Hessian and the monotonic behavior. In the next chapter we will investigate whether these observations are similar for other nonlinear problems.



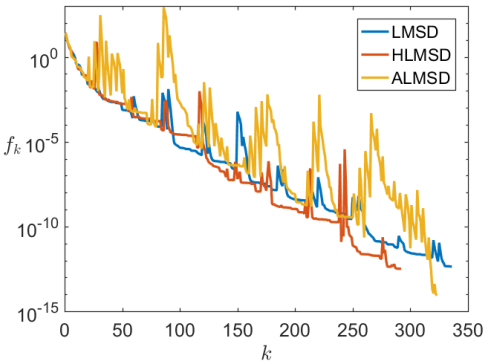
(a) BB1, BB2, ABB, ABB_{min}



(b) SDA, SDC, Yuan

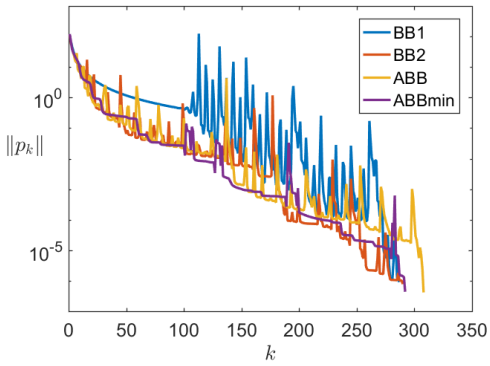


(c) LMSD, HLMSD, ALMSD for $m = 3$

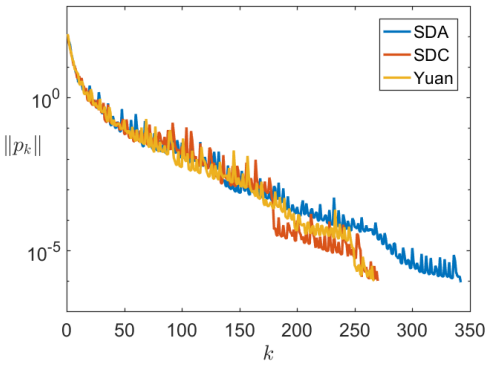


(d) LMSD, HLMSD, ALMSD for $m = 5$

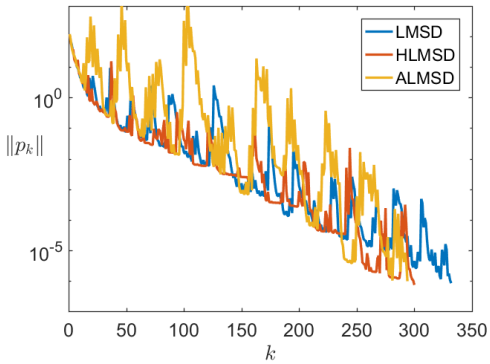
Figure 5.1: Function evaluations for quadratic case with $x_1 = \dfrac{(n, n-1, \ldots, 1)^T}{\|(n, n-1, \ldots, 1)\|}$



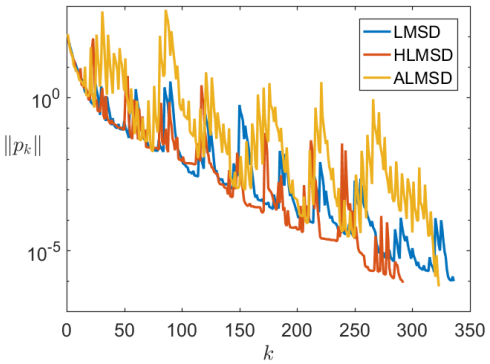
(a) BB1, BB2, ABB, ABB_{min}



(b) SDA, SDC, Yuan

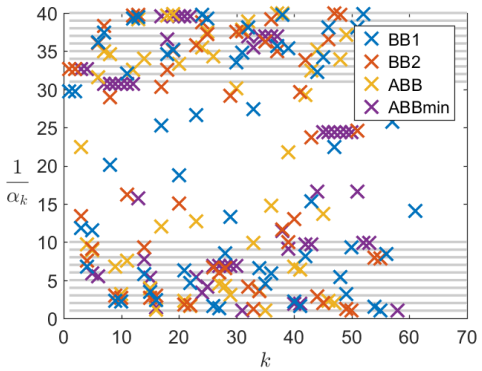


(c) LMSD, HLMSD, ALMSD for $m = 3$

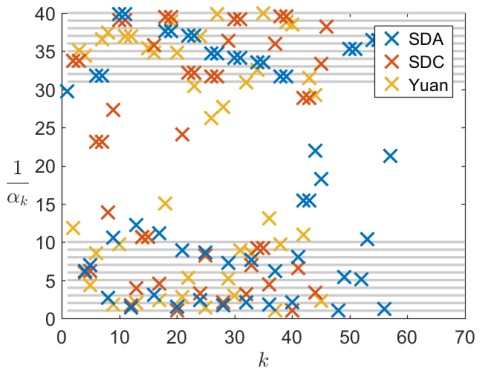


(d) LMSD, HLMSD, ALMSD for $m = 5$

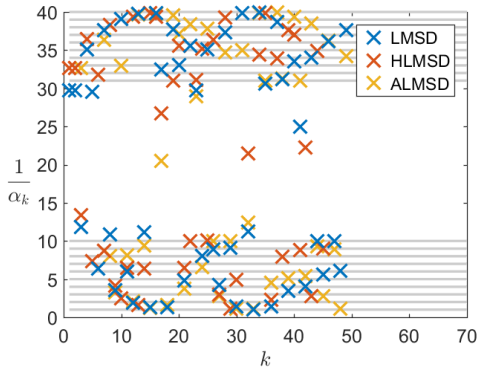
Figure 5.2: Gradient evaluations for quadratic case with $x_1 = \dfrac{(n, n-1, \ldots, 1)^T}{\|(n, n-1, \ldots, 1)\|}$



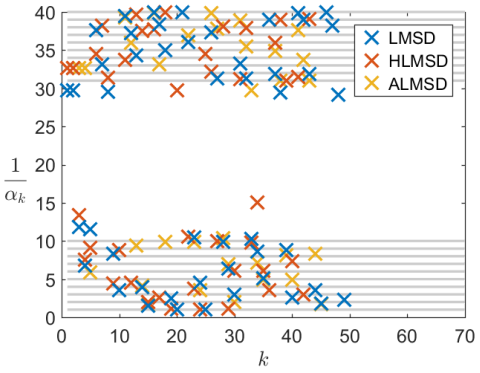
(a) BB1, BB2, ABB, ABB_{min}



(b) SDA, SDC, Yuan



(c) LMSD, HLMSD, ALMSD for $m = 3$



(d) LMSD, HLMSD, ALMSD for $m = 5$

Figure 5.3: Distribution of $1/\alpha_k$ for $A = \text{spdiag}(1, 2, \ldots, 10, 31, 32 \ldots, 40)$

CHAPTER 6

General nonlinear unconstrained optimization

The gradient methods discussed in previous sections can be extended for general nonlinear optimization problems. In the following sections is discussed if and how properties of these methods are preserved in the general case.

6.1 BB1, BB2 and ABB_{min} for general nonlinear optimization

Di Serafino et al. [32] extend the Barzilai–Borwein methods for general unconstrained minimization problems. The algorithms are very similar to the algorithms for the quadratic case. In Algorithm 4 the ABB_{min} algorithm for general unconstrained problems is stated with a few modifications to improve convergence. Now we will discuss these modifications.

Following the analysis of Fletcher [16] we force a sufficient reduction of the objective function with respect to f_{ref}, which is the maximum of the past M function evaluations. If the reduction condition is not satisfied, we use Armijo line search to reduce the step length until the condition is satisfied. As discussed in section 3.1.1 the parameter c is set rather small, i.e., $c = 10^{-4}$. In the following section on LMSD we discuss this reduction condition in more detail. Note that the generalized version of BB1 is Algorithm 4 with $\tau = 0$. The generalized version of BB2 is the BB1 algorithm where the BB1 step length is replaced by the BB2 step length. All methods keep the sequence of step lengths bounded below and above by the positive constants α_{min} and α_{max}. Furthermore, in the case $p_k^T(p_{k+1} - p_k) \geq 0$, we set $\alpha_{k+1} = \alpha_{\text{max}}$. The parameter setting of τ and M is the following: $\tau = 0.8$ and $M = 9$ as it gives satisfactory results in many situations according to [18]. We will set $\alpha_{\text{min}} = 10^{-10}$, $\alpha_{\text{max}} = 10^5$ and $\alpha_1 = 1$ according to [32].

Algorithm 4: $\mathbf{ABB}_{\min}$ for general unconstrained minimization problems.

Input: $x_1 \in \mathbb{R}^n$, $\texttt{tol} > 0$, $M \in \mathbb{N}$, $m \in \mathbb{N}$: $m \ll n$, $f : \mathbb{R}^n \to \mathbb{R}$, $0 \leq \alpha_{\min} \leq \alpha_{\max}$, $\alpha_1 \in [\alpha_{\min}, \alpha_{\max}]$, $c \in (0,1)$, $\tau \in (0,1)$
Output: Approximate solution to $\arg\min_{x \in \mathbb{R}^n} f(x)$

$p_1 = -\nabla f_1$
if $\|p_1\| < \texttt{tol}$
  return x_1
end if
for $k = 1, 2, \ldots$
  $f_{\text{ref}} = \max\{f_{k-j} \colon 0 \leq j \leq \min(k, M)\}$
  if $f(x_k - \alpha_k p_k) - f_{\text{ref}} > -c\,\alpha_k \|p_k\|^2$
    $\alpha_k = \alpha_k^{\text{Armijo}}$
  end if
  $x_{k+1} = x_k - \alpha_k p_k$
  if $\|p_{k+1}\| < \texttt{tol}\|p_1\|$
    return x_{k+1}
  end if
  $d = p_{k+1} - p_k$
  $\zeta = -p_k^T d$
  if $\zeta > 0$
    $\alpha_{k+1}^{\text{BB1}} = \max\left\{\alpha_{\min}, \min\left\{\dfrac{\alpha_k p_k^T p_k}{\zeta}, \alpha_{\max}\right\}\right\}$
    $\alpha_{k+1}^{\text{BB2}} = \max\left\{\alpha_{\min}, \min\left\{\dfrac{\alpha_k \zeta}{d^T d}, \alpha_{\max}\right\}\right\}$
    if $\alpha_{k+1}^{\text{BB2}} < \tau\,\alpha_{k+1}^{\text{BB1}}$
      $\alpha_{k+1} = \min\{\alpha_j^{\text{BB2}} \colon j = \max\{1, k+1-m\}, \ldots, k+1\}$
    else
      $\alpha_{k+1} = \alpha_{k+1}^{\text{BB1}}$
    end if
  else
    $\alpha_{k+1} = \alpha_{\max}$
  end if
end for

6.2 LMSD for general nonlinear optimization

Di Serafino et al. [32] study the limited memory methods for general unconstrained minimization problems. Although the nonmonotonic sweep method is effective for the quadratic case, it is often not favorable to have nonmonotonic behavior for general nonlinear problems. Dai and Fletcher [10] give counter examples to show that the nonmonotonic behavior of the Barzilai–Borwein method could result in cycling behavior when used in a projection method.

As Fletcher [16] explains, it is preferable to use a sweep method in which the sequence $\{f_{k,1}\}_{k\in\mathbb{N}}$ decreases monotonically and nonmonotonic behavior $f_{k,j} > f_{k,j-1}$ is allowed within a sweep. The line search is modified in the following way: If the new step length does not produce a sufficient reduction, the current memory sweep of m iterations is interrupted, the step length is adjusted by backtracking and new Ritz-like values are computed. By this choice, the memory parameter may vary with a maximum value bounded by m.

Dai [9] observes a numerical drawback of nonmonotonic line search methods and suggested a standard Armijo line search when the following condition is not satisfied for some $c \in (0, 1)$:

$$f(x_{k,j} + \alpha_{k,j}p_{k,j}) \leq f_{k,j} - c\,\alpha_{k,j}\|p_{k,j}\|^2.$$

Di Serafino et al. [32] modify this condition further using Fletcher's analysis [16] by setting f_{ref} equal to the value of the objective function at the beginning of the sweep:

$$f(x_{k,j} + \alpha_{k,j}p_{k,j}) \leq f_{\text{ref}} - c\,\alpha_{k,j}\|p_{k,j}\|^2. \tag{6.1}$$

Then, when a calculated step length does not result in a sufficient reduction with respect to f_{ref}, the sweep is interrupted and a standard Armijo line search [27] is performed before computing a new set of Ritz-like values.

For general unconstrained problems, it is not possible to compute the matrix T_k in the same way as in the quadratic example in section 5.2. The reason for this is that we do not necessarily have the Hessian available. However, it is possible to rewrite the matrix T_k such that it does not depend on the Hessian anymore.

Recall from recursion (3.1) that we have $p_{k+1,j} = p_{k,j} - \alpha_{k,j}Ap_{k,j}$ in the quadratic case. This implies the following expression for $Ap_{k,j}$:

$$Ap_{k,j} = \frac{p_{k,j} - p_{k+1,j}}{\alpha_{k,j}}.$$

By writing this in matrix form we get the following identity:

$$AP_k = [P_k \quad p_{k+1,j}]J_k, \tag{6.2}$$

where P_k and J_k are defined as the following matrices:

$$P_k = [p_{k,j-m}, \ldots, p_{k,j}], \quad J_k = \begin{bmatrix} \alpha_{k,j-m}^{-1} & & \\ -\alpha_{k,j-m}^{-1} & \ddots & \\ & \ddots & \alpha_{k,j}^{-1} \\ & & -\alpha_{k,j}^{-1} \end{bmatrix}, \tag{6.3}$$

with indices $(k, -1) := (k - 1, m), (k, -2) := (k - 1, m - 1)$ and so forth if $j < m$.
Now we can rewrite T_k independent of A by using identity (6.2) and $P_k = Q_kR_k$:

$$T_k = Q_k^TAQ_k = (P_kR_k^{-1})^TAP_kR_k^{-1} = R_k^{-T}P_k^T[P_k \quad p_{k+1,j}]J_kR_k^{-1} = [R_k \quad R_k^{-T}P_k^Tp_{k+1,j}]J_kR_k^{-1}.$$

We use the notation $r_k := R_k^{-T}P_k^Tp_{k+1,j}$ to get the following expression for T_k:

$$T_k = [R_k \quad r_k]J_kR_k^{-1}. \tag{6.4}$$

This matrix T_k can be used to calculate the Ritz-like values for the nonlinear case, but there might be some issues.

In the quadratic case the Hessian was considered symmetric positive definite. This implies that all its eigenvalues are positive and real-valued and hence that all the Ritz values are positive and real.
Since the matrix T_k is not generally symmetric for general nonlinear problems the eigenvalues are not necessarily real-valued. To avoid this problem we can construct a symmetric matrix close to T_k and use the eigenvalues of that matrix to calculate the Ritz-like values of the next sweep.

One strategy to generate real Ritz-like values is by calculating the eigenvalues of the symmetric part of T_k: $\frac{1}{2}(T_k + T_k^T)$, but there might be better symmetric approximations. Higham [20] has shown that the nearest symmetric positive semidefinite matrix in the Frobenius norm to an arbitrary real matrix X is shown to be $\frac{1}{2}(B+H)$, where H is the symmetric polar factor of the symmetric part $B = \frac{1}{2}(X + X^T)$. The so-called polar factor can be computed using singular value decomposition, as described in [20]. Therefore, $\frac{1}{2}(T_k + H_k)$ with H_k the symmetric polar factor of T_k seems a better choice to calculate real Ritz-like values. However, Fletcher [16] noted that in practice, due to ill-conditioning in R_k, the product with R_k^{-1} in (6.4) seriously magnifies non-quadratic terms in the upper triangle part of T_k. To solve this problem Fletcher introduced a symmetric tridiagonal matrix T_k^{sym} by replacing the strictly upper triangle of T_k by the transpose of its strictly lower triangle. In MATLAB notation:

$$T_k^{\text{sym}} = \text{tril}(T) + \text{tril}(T, 1)'. \tag{6.5}$$

The Ritz-like values $\theta_{k,j}$, that are used as step lengths for the next m iterations, are defined as the eigenvalues of one of these symmetric matrices. In the upcoming numerical results we calculate Ritz-like values as eigenvalues of T_k^{sym}.

Another issue is that in the nonlinear case the Hessian is not necessarily positive definite. This implies that there might be nonpositive Ritz-like values. This case is handled by discarding these nonpositive values, which means there are less than m step lengths provided for the upcoming sweep. If no positive eigenvalues are available, the initial step length $\alpha_{1,1}$, which is used for the first iteration, is used to provide a sweep of length 1. Di Serafino et al. [32] also noted that the presence of nonpositive eigenvalues highlights critical situations, which can originate from a nonpositive curvature or an inadequate approximation of the eigenvalues of the current Hessian. For that reason, not only the nonpositive eigenvalues are discarded but also the oldest back gradients. Furthermore, regardless of the step length rule, all the methods keep the sequence of step lengths bounded below and above by the positive constants $\alpha_{\min}$ and $\alpha_{\max}$. The parameters $x_{1,1}$, m, `tol`, c, $\alpha_{\min}$, $\alpha_{\max}$ and $\alpha_{1,1}$ of the general LMSD algorithm are set identical to the parameters in the quadratic algorithm described in section 5.2.

Dai [9] also discusses the R-linear convergence rate for nonmonotonic line search methods for general unconstrained optimization problems under the assumptions that f is bounded from below, strongly convex and has Lipschitz continuous gradient. We will not go in further detail on those properties.

Algorithm 5: Limited Memory Steepest Descent for general unconstrained minimization problems.

Input: $x_{1,1} \in \mathbb{R}^n$, $m \in \mathbb{N} : m \ll n$, $\texttt{tol} > 0$, $c \in (0,1)$, $f : \mathbb{R}^n \to \mathbb{R}$, $0 \leq \alpha_{\min} \leq \alpha_{\max}$, $\alpha_{1,1} \in [\alpha_{\min}, \alpha_{\max}]$, $j = 1$.
Output: Approximate solution to $\arg\min_{x \in \mathbb{R}^n} f(x)$

$p_{1,1} = -\nabla f_{1,1}$
if $\|p_{1,1}\| < \texttt{tol}$
  return $x_{1,1}$
end if
for $k = 1, 2, \ldots$
  $f_{\text{ref}} = f_{k,1}$
  while $j > 0$
    $j_0 = j$
    if $f(x_{k,j} + \alpha_{k,j} p_{k,j}) - f_{\text{ref}} > -c\, \alpha_{k,j} \|p_{k,j}\|^2$
      $\alpha_{k,j} = \alpha_{k,j}^{\text{Armijo}}$
      $j = 0$
    else
      $j = j - 1$
    end if/else
    $x_{k,j_0+1} = x_{k,j_0} + \alpha_{k,j_0} p_{k,j_0}$
    $p_{k,j_0+1} = -\nabla f_{k,j_0+1}$
    if $\|p_{k,j_0+1}\| > \|p_{k,j_0}\|$
      $j = 0$
    end if
    if $\|p_{k,j_0+1}\| < \texttt{tol} \|p_{1,1}\|$
      return x_{k,j_0+1}
    end if
    Update P_k and J_k according to (6.3)
  end while
  $x_{k+1,1} = x_{k,j_0+1}$
  $p_{k+1,1} = p_{k,j_0+1}$
  Calculate R_k by reduced QR-decomposition of P_k
  If R_k is too ill-conditioned, update P_k, J_k and R_k by removing oldest back gradient
  Calculate $T_k = [R_k \quad r_k]\, J_k R_k^{-1}$ as in (6.4)
  Calculate $\{\theta_{k,j}\}_{j \in [m]}$ as the eigenvalues of T_k^{sym} from (6.5) in ascending order
  Remove all nonpositive Ritz values
  $\{\alpha_{k+1,j}\}_{j \in [m]} = \{\max\{\alpha_{\min}, \min\{\theta_{k,j}^{-1}, \alpha_{\max}\}\}_{j \in [m]}$
  Set j as the number of step lengths for the upcoming sweep
end for

6.3 HLMSD for general nonlinear optimization

Recall from section 5.3 that Fletcher [16] also introduces the alternative limit memory method with harmonic Ritz values. These values would satisfy the equality

$$Q_k^T A^2 Q_k x = \widehat{\theta}_{k,j} Q_k^T A Q_k x,$$

for some $x \in \mathbb{R}^n$. If we write $T_k = Q_k^T A Q_k$ (as in the LMSD method) and $S_k := Q_k^T A^2 Q_k$, then the harmonic Ritz-like values are the eigenvalues of $(S_k^{-1} T_k)^{-1}$. For the nonlinear cases we do not necessarily have the Hessian available. Note that T_k can be computed without explicit access to the Hessian by formula (6.4). As Fletcher [16] explains, the matrix S_k can also be computed without explicit access to the Hessian by using (6.2) and $P_k = Q_k R_k$:

$$S_k = R_k^{-T} J_k^T [P_k \quad p_{k+1,j}]^T [P_k \quad p_{k+1,j}] J_k R_k^{-1}.$$

This can be rewritten (by using the reduced QR-factorization of $[P_k \quad p_{k+1,j}]$) to:

$$S_k = R_k^{-T} J_k^T \begin{bmatrix} R_k & r_k \\ 0 & \xi_k \end{bmatrix}^T \begin{bmatrix} R_k & r_k \\ 0 & \xi_k \end{bmatrix} J_k R_k^{-1},$$

where $\xi_k = \sqrt{\|p_{k+1,j}\|^2 - \|r_k\|^2}$.

This expression for S_k can be rewritten to the following form as done by Curtis and Guo [7], which we will use in the implementation of the HLMSD method:

$$S_k = T_k^T T_k + z_k z_k^T, \text{ with } z_k = R_k^{-T} J_k^T \begin{bmatrix} 0 \\ \xi_k \end{bmatrix}.$$

Note that S_k is not necessarily symmetric, so the eigenvalues of $(S_k^{-1} T_k)^{-1}$ might not be real-valued. To solve this problem we use T_k^{sym} from (6.5) to define S_k^{sym}:

$$S_k^{\text{sym}} := (T_k^{\text{sym}})^2 + z_k z_k^T, \text{ with } z_k = R_k^{-T} J_k^T \begin{bmatrix} 0 \\ \xi_k \end{bmatrix}. \tag{6.6}$$

In the following lemma we prove that the eigenvalues of $((S_k^{\text{sym}})^{-1} T_k^{\text{sym}})^{-1}$ are real valued so we can define the harmonic Ritz-like values as in [16].

Lemma 6.3.1. *Suppose that there is no nonzero $x \in \mathbb{R}^n$ such that $T_k^{sym} x = 0$ and $z_k^T x = 0$. Then S_k^{sym} is positive definite and the eigenvalues of $((S_k^{sym})^{-1} T_k^{sym})^{-1}$ are real valued.*

Proof. Note that by construction T_k^{sym} and S_k^{sym} are symmetric. First we will prove that S_k^{sym} is positive semidefinite. We use equation (6.6) to retrieve the following for some $x \in \mathbb{R}^m \backslash \{0\}$:

$$\begin{aligned} x^T S_k^{\text{sym}} x &= x^T (T_k^{\text{sym}})^2 x + x^T z_k z_k^T x \\ &= (T_k^{\text{sym}} x)^T (T_k^{\text{sym}} x) + (x^T z_k)(z_k^T x) \\ &= \left\| T_k^{\text{sym}} x \right\|^2 + (x^T z_k)^2 \\ &\geq 0. \end{aligned}$$

The assumption that there is no nonzero $x \in \mathbb{R}^n$ such that $T_k^{\text{sym}} x = 0$ and $z_k^T x = 0$ implies that the equality never holds, thus S_k^{sym} is SPD and hence invertible. Note that T_k^{sym} is also invertible by this assumption.

Since S_k^{sym} is SPD, it is diagonalizable, i.e, there is an orthogonal matrix Q and a diagonal matrix D such that

$$S_k^{\text{sym}} = Q^T D Q,$$

where $D = \text{diag}(\sigma_1, \ldots, \sigma_m)$ with $\sigma_1, \ldots, \sigma_m$ the eigenvalues of S_k^{sym}. Since S_k^{sym} is positive definite we know that the eigenvalues are positive and hence we can take the square root of D:

$$D^{1/2} = \text{diag}(\sqrt{\sigma_1}, \ldots, \sqrt{\sigma_m}).$$

Now we get the following result by using $QQ^T = I$:

$$(Q^T D^{1/2} Q)^2 = (Q^T D^{1/2} Q)(Q^T D^{1/2} Q) = Q^T D Q = S_k^{\text{sym}}.$$

Observe that $Q^T D^{1/2} Q$ is a symmetric matrix as well:

$$(Q^T D^{1/2} Q)^T = Q^T (D^{1/2})^T (Q^T)^T = Q^T D^{1/2} Q.$$

So now we have a symmetric invertible matrix $B = Q^T D^{1/2} Q$ such that $B^2 = S_k^{\text{sym}}$. Next we claim that the matrix $B^{-1} T_k^{\text{sym}} B^{-1}$ is symmetric and has the same eigenvalues as $(S_k^{\text{sym}})^{-1} T_k^{\text{sym}}$. The symmetric part follows directly from the fact that B and T_k^{sym} are symmetric. Suppose λ is an eigenvalue of $B^{-1} T_k^{\text{sym}} B^{-1}$ with corresponding eigenvector y, i.e.,

$$B^{-1} T_k^{\text{sym}} B^{-1} y = \lambda y.$$

We multiply on the left with B and define $x := B^{-1} y$ to retrieve the following result:

$$T_k^{\text{sym}} x = \lambda B^2 x = \lambda S_k^{\text{sym}} x,$$

which implies that the spectrum of $(S_k^{\text{sym}})^{-1} T_k^{\text{sym}}$ is identical to the spectrum of the symmetric matrix $B^{-1} T_k^{\text{sym}} B^{-1}$. This implies that its eigenvalues are real valued. Since T_k^{sym} and S_k^{sym} are both invertible we get the desired result by observing that the eigenvalues of $((S_k^{\text{sym}})^{-1} T_k^{\text{sym}})^{-1}$ are the reciprocals of the eigenvalues of $(S_k^{\text{sym}})^{-1} T_k^{\text{sym}}$. □

Hence we can define the harmonic Ritz-like values as the eigenvalues of $((S_k^{\text{sym}})^{-1} T_k^{\text{sym}})^{-1}$ in ascending order (as initially mentioned by Fletcher [16]) as long as T_k^{sym} and S_k^{sym} are invertible. In practice this is usually the case (see e.g. [7], [16] or [32]). The HLMSD method for general nonlinear problems is similar to Algorithm 5 with the adjustments described in this section.

6.4 ALMSD for general nonlinear optimization

In section 5.4 the adaptive LMSD method for quadratic problems is described. For general nonlinear problems this works similarly. We adjust Algorithm 5 by computing both the Ritz-like values and the harmonic Ritz-like values and determine which set is used for the upcoming sweep. The step lengths we use for the upcoming sweep are computed as follows for $j \leq m$ and $\tau \in (0, 1)$:

$$\alpha_{k,j}^{\text{ALMSD}} = \begin{cases} \alpha_{k,j}^{\text{HLMSD}} & \text{if} \quad \min\limits_{j \in [m]} \alpha_{k,j}^{\text{HLMSD}} < \tau \min\limits_{j \in [m]} \alpha_{k,j}^{\text{LMSD}}, \\ \alpha_{k,j}^{\text{LMSD}} & \text{otherwise.} \end{cases}$$

6.5 Numerical results for nonlinear test problems

In this section we will discuss some numerical results of the BB1, BB2, ABB_{min}, LMSD, HLMSD and ALMSD methods applied to a set of nonlinear test problems. All numerical results are computed by running MATLAB implementations of the described methods.

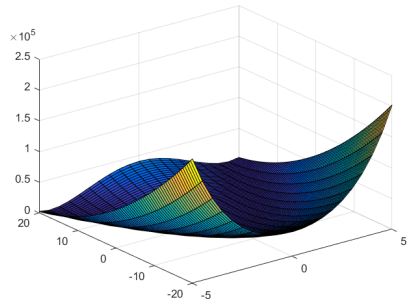
6.5.1 Test problems

In order to analyze the practical behavior of the methods described in the previous sections, we apply them to the following test problems and discuss the most remarkable results.

Problem I: Rosenbrock [31]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2.$$

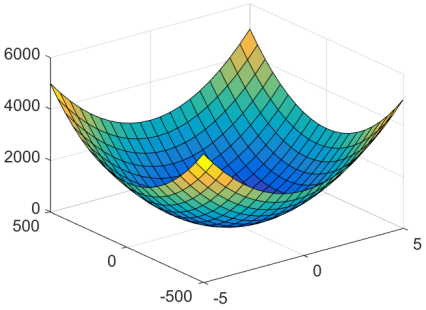
This function has a global minimum at $x^* = (1, 1)^T$. The starting vector is set as $x^* = (-1.2, 1)^T$.



Problem II: 2D Quadratic [4]
For $a = 100$, the objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = ax_1^2 + \frac{1}{a}x_2^2.$$

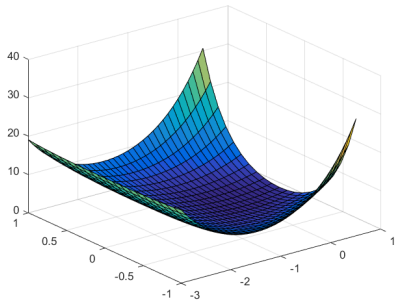
This function has a global minimum at $x^* = (0, 0)^T$. The starting vector is set as $x_1 = (\frac{1}{a}, a)^T$.



Problem III: 2D Exponential [5]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}.$$

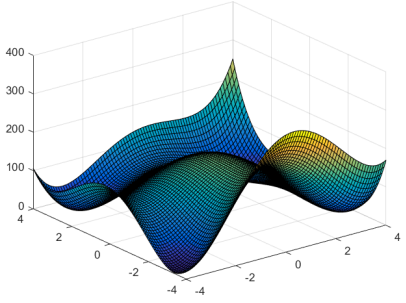
This function has a global minimum at $x^* = (\log\left(\frac{\sqrt{2}}{2}\right), 0)^T$. The starting vector is set as $x_1 = (1, 1)^T$.



Problem IV: 2D Himmelblau [22]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

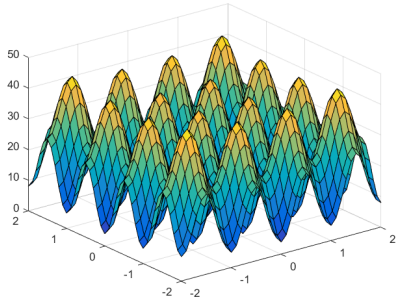
This function has a local maximum at $x^* \approx (-0.27, -0.92)$ and four identical global minima equal to 0 at $x_{*1} = (3, 2)$, $x_{*2} \approx (-3.78, -3.28)$, $x_{*3} \approx (-2.81, 3.13)$, $x_{*4} \approx (3.58, -1.85)$. The starting vector is set as $x_1 = (3, 1)^T$.



Problem V: 2D Rastrigin [26]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = 20 + x_1^2 - 10\cos(2\pi x_1) + x_2^2 - 10\cos(2\pi x_2).$$

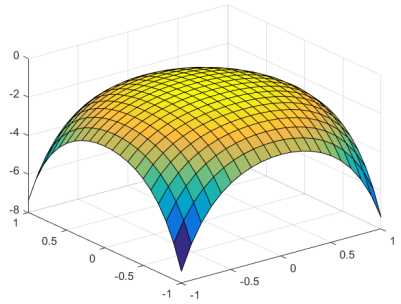
This function has a global minimum at $x^* = (0, 0)^T$. The starting vector is set as $x_1 = (20, 30)^T$.



Problem VI: 2D Gaussian
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = -e^{x_1^2 + x_2^2}.$$

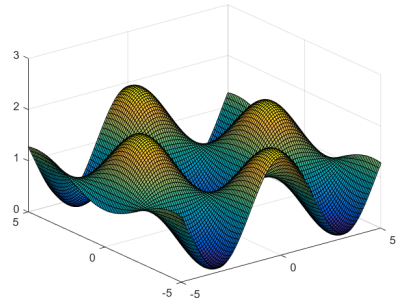
This function has no minima. The starting vector is set as $x_1 = (10^{-5}, 10^{-5})$.



Problem VII: 2D Griewank [19]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = 1 + \frac{x_1^2 + x_2^2}{4000} - \cos(x)\cos\left(\frac{y}{\sqrt{2}}\right).$$

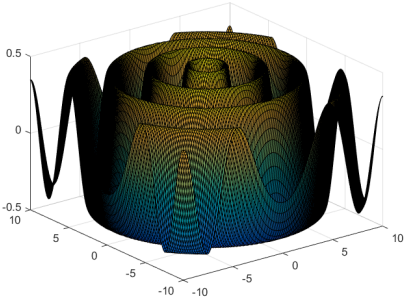
This function has a global minimum at $x^* = (0, 0)^T$. The starting vector is set as $x_1 = (4, 4)^T$.



Problem VIII: 2D Shaffer-6 [15]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = \frac{\sin(\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1 + (x_1^2 + x_2^2)/1000)^2}.$$

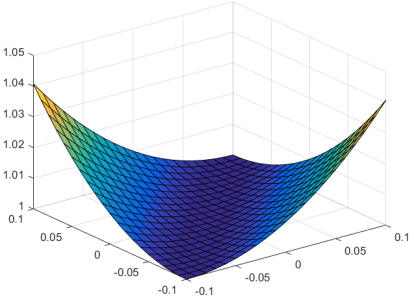
This function has infinitely many local minima, one in $x^* = (0, 0)^T$. The starting vector is set as $x_1 = (1, 1)^T$.



Problem IX: 2D Gaussian-line
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = -e^{(x_1 - x_2)^2}.$$

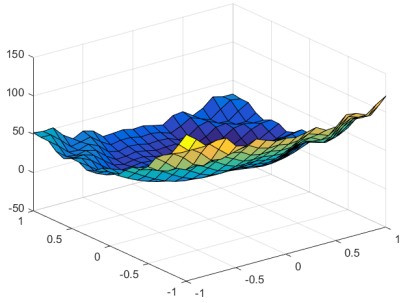
This function has infinitely many global minima on the line $x_1 = x_2$. The starting vector is set as $x_1 = (-2, 1)^T$.



Problem X: Rosty-7 [23]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$\begin{aligned} f(x) =& 2 + 0.01(5x_2 - 5x_1^2)^2 + (1 - 5x_1)^2 + 2(2 - 5x_2 \\ &+ 7\sin(2.5x_1)\sin(17.5x_1x_2). \end{aligned}$$

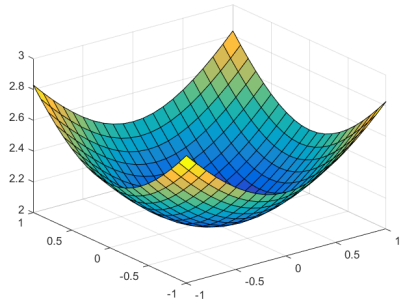
This function has many local minima. The starting vector is set as $x_1 = (1, 1)^T$.



Problem XI: Hessian not bounded below [4]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x) = \sqrt{1 + x_1^2} + \sqrt{1 + x_2^2}.$$

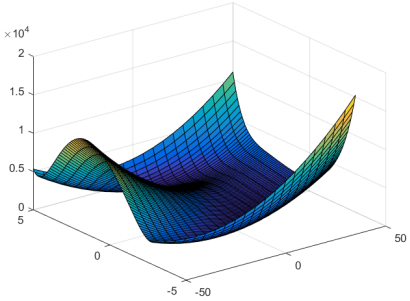
This function has a global minimum at $x^* = (0, 0)^T$. The starting vector is set as $x_1 = (10, 10)^T$.



Problem XII: Freudenstein and Roth [4]
The objective function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$\begin{aligned} f(x) =& (-13 + x_1 + ((5 - x_2)x_2 - 2)x_2)^2 \\ &+ (-29 + x_1 + ((1 + x_2)x_2 - 14)x_2)^2. \end{aligned}$$

This function has a global minimum at $x^* = (5, 4)^T$ and a local minimum at $x_* = \left(\frac{53}{3} - \frac{4\sqrt{22}}{3}, \frac{2}{3} - \frac{\sqrt{22}}{3}\right)$.
The starting vector is set as $x_1 = (-50, 7)^T$, $x_1 = (20, 7)^T$, $x_1 = (20, -18)^T$ or $x_1 = (5, -10)^T$.



Problem XIII: Quadratic diagonal
The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$f(x) = \tfrac{1}{2}x^T Ax.$$

Here A is an SPD matrix given by $A = \text{spdiag}(1, \ldots, n)$. This function has a global minimum at $x^* = (0, \ldots, 0)^T$.

Problem XIV: Trigonometric [17]
The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$f(x) = \|b - Av(x) + Bu(x)\|^2,$$

where $v(x) = (\sin(x_1), \ldots, \sin(x_n))^T$, $u(x) = (\cos(x_1), \ldots, \cos(x_n))^T$ and A and B are square matrices of order n with entries generated as random integers in $(100, 100)$. Given a vector $x^* \in \mathbb{R}^n$ with entries randomly generated from a uniform distribution in (π, π) , the vector b is computed so that $f(x^*) = 0$, i.e. x^* is a minimum point. The starting vector is set as $x_1 = x^* + 0.1r$, where $r \in \mathbb{R}^n$ has random entries from a uniform distribution in $[\pi, \pi]$.

Problem XV: Convex [29]
The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$f(x) = \sum_{i=1}^{n} \frac{i}{10}(e^{x_i} x_i).$$

This is a strictly convex problem, having a diagonal Hessian matrix with diagonal entries equal to $\frac{i}{10}e^{x_i}, i = 1, \ldots, n$. The solution x^* is the zero vector and the minimum value is $f(x) = \frac{n(n+1)}{20}$.

Problem XVI: Chained Rosenbrock [33]
The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$f(x) = \sum_{i=2}^{n} (4\varphi_i(x_{i-1}x_i^2)^2 + (1x_i)^2),$$

where the values φ_i are defined as follows for $i = 1, \ldots, 50$:
$\{1.25, 1.40, 2.40, 1.40, 1.75, 1.20, 2.25, 1.20, 1.00, 1.10, 1.50, 1.60, 1.25, 1.25, 1.20, 1.20, 1.40, 0.50,$
$0.50, 1.25, 1.80, 0.75, 1.25, 1.40, 1.60, 2.00, 1.00, 1.60, 1.25, 2.75, 1.25, 1.25, 1.25, 3.00, 1.50, 2.00,$
$1.25, 1.40, 1.80, 1.50, 2.20, 1.40, 1.50, 1.25, 2.00, 1.50, 1.25, 1.40, 0.60, 1.50\}$.
We will also consider $n = 25$ and $n = 200$, with $\varphi_{i+50j} = \varphi_i$, $i = 1, \ldots, 50$, $j = 1, 2, 3$.
This function has global minimum $x^* = (1, \ldots, 1)^T$.

6.5.2 Numerical results on 2D test problems

In this section we will restrict ourselves to the generalized BB1, BB2 and ABB_{min} methods to gain insight in their performance. The limited memory methods are not applicable to 2D problems since the memory parameter m should be smaller than n. For the first experiment we apply BB1, BB2 and ABB_{min} to the 2D test problems I, II,..., XI from section 6.5.1.

Table 6.1: Function evaluations for the stated nonlinear test problems

Problem	BB1			BB2			ABB_{min}		
	$\#f$	nm-$\#f$	f	$\#f$	nm-$\#f$	f	$\#f$	nm-$\#f$	f
I	180	16	1.14e-08	110	11	3.69e-10	107	2	6.38e-13
II	65	4	1.54e-14	18	1	2.89e-16	15	1	5.90e-36
III	21	2	2.56	19	1	2.56	22	1	2.56
IV	42	4	1.16e-17	15	1	8.42e-13	16	1	3.46e-13
V	3	0	0.00	3	0	0.00	3	0	0.00
VI	$> 10^4$	0	$-\infty$	$> 10^4$	0	$-\infty$	$> 10^4$	0	$-\infty$
VII	12	0	7.40e-3	11	0	7.40e-3	12	0	7.40e-3
VIII	7	0	2.75e-10	7	0	2.75e-10	7	0	2.75e-10
IX	21	0	1.00	21	0	1.00	21	0	1.00
X	16	2	2.87	16	2	2.87	16	2	2.87
XI	16	0	2.00	16	0	2.00	16	0	2.00

In Table 6.1 the results are shown for the three methods with the parameters from Table 6.2 and the program is interrupted if the stopping criterion is not satisfied within 10^4 iterations. The red numbers indicate that there was no convergence to a global minimum (if it exists), but instead there was convergence to a local minimum.

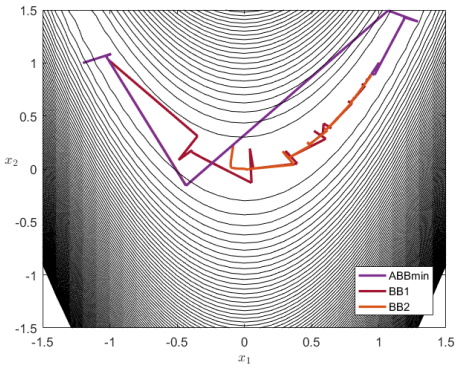
Table 6.2: Parameter values for BB1, BB2 and ABB_{min}

Parameter	Value	Parameter	Value
`tol`	10^{-6}	M	9
α_{min}	10^{-10}	m	6
α_{max}	10^5	c	10^{-4}
α_1	1	τ	0.8

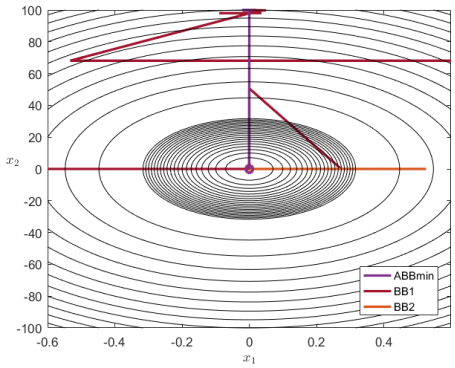
Column $\#f$ displays the number of function evaluations, column nm-$\#f$ the number of function evaluations at which nonmonotonic behavior is registered and column f displays the objective function value once the stopping criterion is satisfied. We will discuss the results for each problem to analyze the behavior of ABB_{min} compared to the behavior of BB1 and BB2.

- Problem I has a nonconvex objective function that has the global minimum inside a long, narrow, parabolic shaped flat alley. Finding the alley is often not a problem, but converging to the global minimum might be more difficult. This can be observed in Figure 6.1a. Note that ABB_{min} has a lower amount of $\#f$ and nm-$\#f$. All three methods do converge to the global minimum if we run the program for 1000 starting vectors taken from the uniform distribution over $[-1.5, 1.5] \times [-1.5, 1.5]$.
- Problem II has a poorly scaled quadratic function. All three methods converge to the global minimum. BB2 and ABB_{min} seem to perform similarly, but BB1 appears to be slower. The result that ABB_{min} performs similarly to BB2 here can be explained by looking at the construction of the ABB_{min} method described in section 4.4.3: If BB1 step lengths are much bigger than BB2 step lengths, ABB_{min} switches to BB2 step lengths.
- Problem III has a global minimum with a rather flat neighborhood. All three methods perform similarly on this problem and converge to the minimum. This implies that the methods are also applicable for this problem type.
- Problem IV has four identical global minima equal to 0. All methods do converge to one of these minima, but not all to the same one as can be seen in Figure 6.1d. In problems with local minima this behavior might be an issue. For this reason we will investigate if these methods end up in local minima as well in problem XII.
- For Problem V, all three methods converged in one step to the minimum for this starting vector. However, different choices for the starting vector give different results. Often the global minimum is not reached. By running these methods over 1000 random starting vectors from the uniform distribution over $[-2, 2]\times[-2, 2]$, we find that all three methods perform similarly and converge to the optimum in only 5% of the starting vector choices.
- Problem VI has an objective function without global minima. This result shows that all three methods also perform the way they should for this problem type.
- Problem VII has an objective function with infinitely many local minima and one global minimum. For this starting vector choice, none of the methods converge to the global minimum. Instead, they end up in a local minimum as can be seen in Figure 6.2a. All three methods show similar behavior. The global minimum is only reached if the starting vector is chosen close to the minimum.
- Problem VIII has an objective function with infinitely many local minima. Because of the occurrence of $(x_{k+1} - x_k)^T(p_{k+1} - p_k) \leq 0$ the step length is adjusted to α_{max} and the next iterative step is far away from the global minimum where the gradient is close to zero. Therefore all three methods do not converge to the global minimum.
- Problem IX has an objective function with infinitely many global minima on the line $x_1 = x_2$. Depending on the starting vector, different points on this line are found as global minimum for all three methods. However, for this problem, all three methods perform identical iterative steps.
- Problem X has an objective function with many local minima. All three methods do not converge to the global minimum, but instead get stuck in the same local minimum. The global minimum is only reached for starting vectors close to the global minimum.

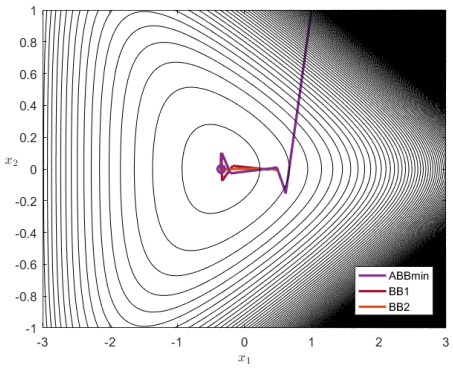
- Problem XI has an objective function that is convex. Despite the fact that the Hessian is positive definite, the Hessian is unbounded from below as shown in [4]. However, the global minimum is found anyway for all three methods.



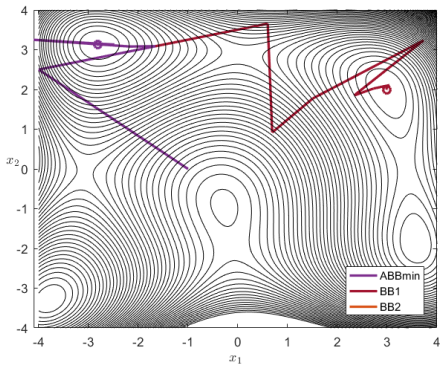
(a) Test problem I



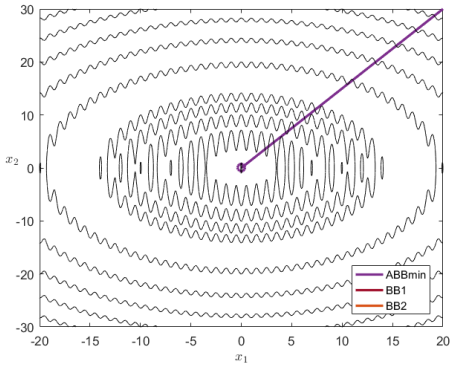
(b) Test problem II



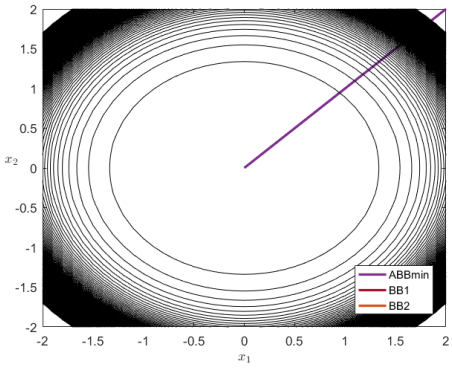
(c) Test problem III



(d) Test problem IV

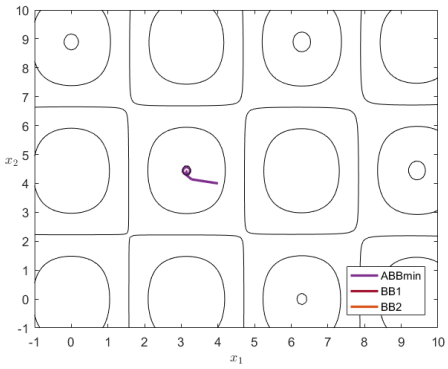


(e) Test problem V

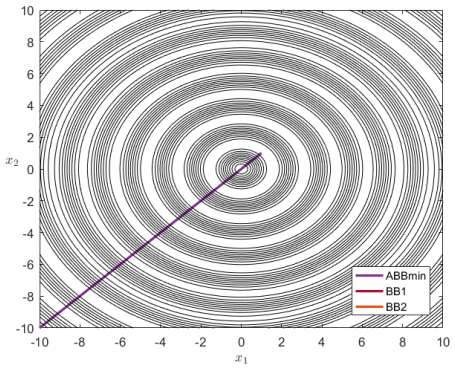


(f) Test problem VI

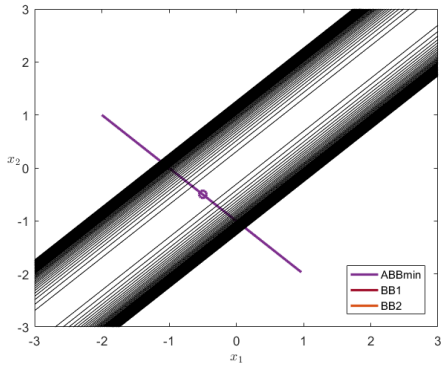
Figure 6.1: Iterative steps of BB1, BB2 and ABB_{min} methods for the stated 2D problems



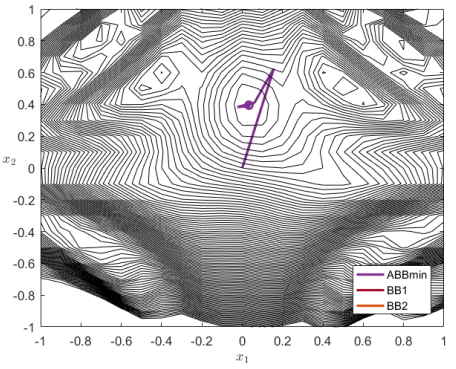
(a) Test problem VII



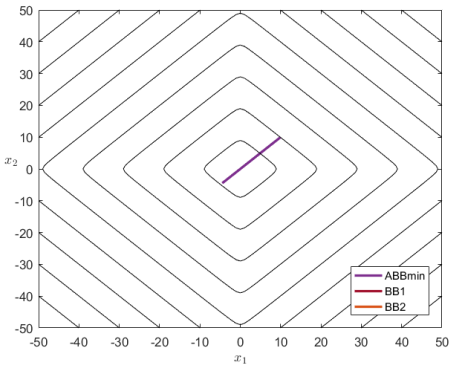
(b) Test problem VIII



(c) Test problem IX



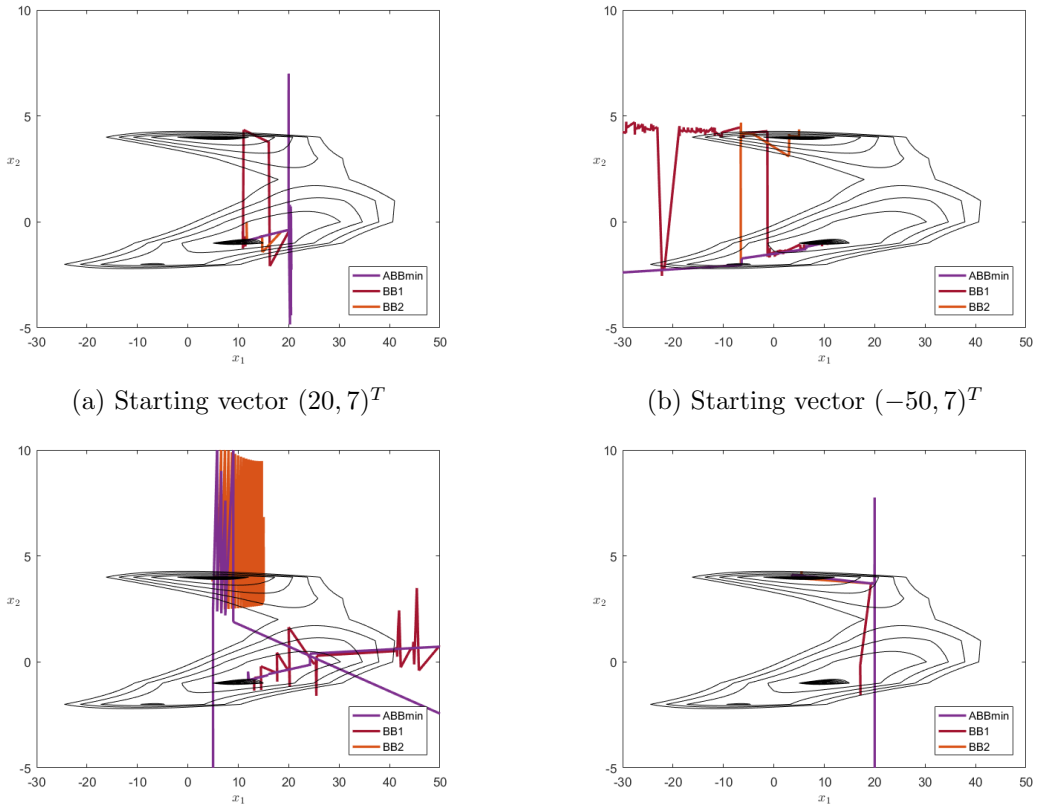
(d) Test problem X



(e) Test problem XI

Figure 6.2: Iterative steps of BB1, BB2 and ABB_{min} methods for the stated 2D problems

For the second experiment we apply BB1, BB2 and ABB_{min} to 2D problem XII and observe the consequences of different choices for the starting vector. We set the parameters the same as in the previous experiments. In Figure 6.3 iterative steps are shown for starting vectors $(20, 7)^T$, $(-50, 7)^T$, $(5, -10)^T$ and $(20, -18)^T$. All three methods either converge to the global minimum or the local minimum, but none of the methods perform better than the other two in general (for all starting vectors). For all four starting vector choices ABB_{min} requires the least amount of function evaluations.



(a) Starting vector $(20, 7)^T$

(b) Starting vector $(-50, 7)^T$

(c) Starting vector $(5, -10)^T$

(d) Starting vector $(20, -18)^T$

Figure 6.3: Iterative steps of BB1, BB2 and ABB_{min} methods for problem XII

For all discussed 2D problems we have seen that the three methods seem to perform similar. In many of the problems the methods do converge to the global minimum, however once local minima are present the methods have trouble localizing the global minimum. ABB_{min} has the least amount of nonmonotonic steps in most of the test cases, while BB1 displays a lot of nonmonotonic behavior.

6.5.3 Numerical results on large scale test problems

Now we will investigate the performance of the limited memory methods. Since $m \ll n$ we can only apply these methods to higher dimensional problems. For the next experiment we will test BB1, BB2, ABB_{min}, LMSD, HLMSD and ALMSD on problems XIII, XIV, XV and XVI from section 6.5.1. The parameter values are set as in Table 6.3 and the program is interrupted if the stopping criterion is not satisfied within 10^4 iterations.

Table 6.3: Parameter values for BB1, BB2, ABB_{min}, LMSD, HLMSD and ALMSD

Parameter	Value	Parameter	Value
`tol`	10^{-6}	M	9
α_{min}	10^{-10}	m	3 or 6
α_{max}	10^{5}	c	10^{-4}
$\alpha_{1,1}$	1	τ	0.8

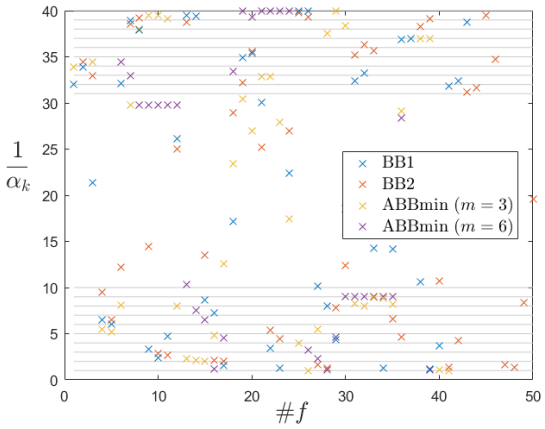
For problem XIII we will consider $n = 1000$ and compute the average number of function evaluations ($\#f$), nonmonotonic function evaluations (nm-$\#f$) and times the reduction condition (6.1) is not satisfied (#red) over 1000 random starting vectors from the unit sphere. The results are shown in Table 6.4.

Table 6.4: Numerical results for test problem XIII with $n = 1000$

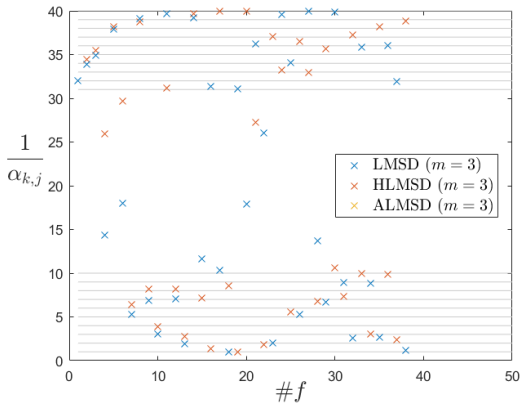
Method	$\#f$	nm-$\#f$	#red
BB1	264	41	39
BB2	192	9	5
ABB_{min} ($m = 3$)	162	7	4
LMSD ($m = 3$)	244	18	21
HLMSD ($m = 3$)	178	2	4
ALMSD ($m = 3$)	237	18	20
ABB_{min} ($m = 6$)	155	5	3
LMSD ($m = 6$)	177	8	9
HLMSD ($m = 6$)	150	2	3
ALMSD ($m = 6$)	177	8	9

All methods found the global minimum of this convex objective function. By construction the methods LMSD, HLMSD and ALMSD are monotone after every sweep. However, inside a sweep nonmonotonic behavior is allowed. For $m = 3$ the limited memory methods display more nonmonotonic behavior compared to the $m = 6$ case. On average the ALMSD method with $m = 3$ used 176 LMSD step lengths and 1 HLMSD step length. The ALMSD method with $m = 6$ used on average 152 LMSD step lengths and 0 HLMSD step lengths. Note that BB1 uses reduction of the step length by backtracking very often compared to the other methods.

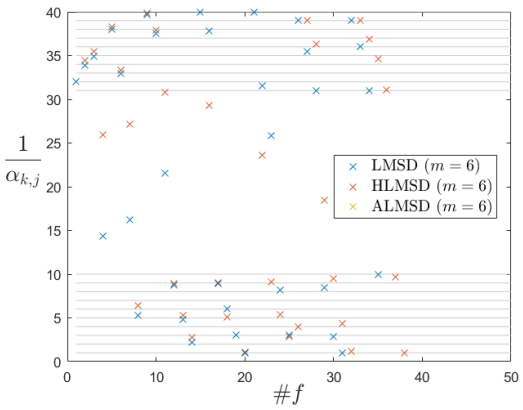
The eigenvalues of the Hessian of problem XIII are known, namely the elements on the diagonal of A. Therefore we could investigate the distribution of $1/\alpha_{k,j}$ and check whether these reciprocals of step sizes are good or bad approximates of the eigenvalues of the Hessian. However, the eigenvalues of $A = \text{spdiag}(1, \ldots, n)$ are spread equally over the interval $[1, n]$. Therefore we look at problem XIII with $n = 20$ and $A = \text{spdiag}(1, 2, \ldots, 10, 31, \ldots, 40)$. In Figure 6.4 can be seen that $\text{ABB}_{\min}$, LMSD, HLMSD and ALMSD produce a better approximation for these eigenvalues than BB1 and BB2. Note that in this problem the reciprocals of the step lengths of ALMSD coincide with the ones from LMSD.



(a) BB1, BB2, $\text{ABB}_{\min}(m = 3)$, $\text{ABB}_{\min}(m = 6)$



(b) LMSD, HLMSD, ALMSD for $m = 3$



(c) LMSD, HLMSD, ALMSD for $m = 6$

Figure 6.4: Distribution of $1/\alpha_{k,j}$ for problem XIII with $A = \text{spdiag}(1, 2, \ldots, 10, 31, 32 \ldots, 40)$

For problem XIV we will consider $n = 50$ and $n = 200$ and compute the average number of function evaluations ($\#f$), nonmonotonic function evaluations (nm-$\#f$) and times the reduction condition is not satisfied (#red) over 200 starting vectors of the form $x_1 = x + 0.1r$, where $r \in \mathbb{R}^n$ has random entries from a uniform distribution in $[\pi, \pi]$. The results are shown in Table 6.5 for $n = 50$ and in Table 6.6 for $n = 200$.

Table 6.5: Numerical results for test problem XIV with $n = 50$

Method	$\#f$	nm-$\#f$	#red
BB1	2680	296	326
BB2	1034	23	11
$\text{ABB}_{\min}$ $(m = 3)$	725	18	9
LMSD $(m = 3)$	1196	74	101
HLMSD $(m = 3)$	832	4	9
ALMSD $(m = 3)$	1031	63	87
$\text{ABB}_{\min}$ $(m = 6)$	685	12	6
LMSD $(m = 6)$	603	24	30
HLMSD $(m = 6)$	662	3	7
ALMSD $(m = 6)$	592	23	29

Table 6.6: Numerical results for test problem XIV with $n = 200$

Method	$\#f$	nm-$\#f$	#red
BB1	4572	496	549
BB2	1612	30	13
$\text{ABB}_{\min}$ $(m = 3)$	1285	25	12
LMSD $(m = 3)$	1910	114	158
HLMSD $(m = 3)$	1505	5	11
ALMSD $(m = 3)$	1854	109	152
$\text{ABB}_{\min}$ $(m = 6)$	1334	21	10
LMSD $(m = 6)$	1041	46	54
HLMSD $(m = 6)$	1140	4	7
ALMSD $(m = 6)$	1050	46	55

Note that there is randomness in the construction of the matrices A and B, the vector x^* and in the starting vector of problem XIV. In this experiment the same matrices and vectors are used for all methods. Taking different matrices and vectors change the problem slightly, which influences the performance of the methods.

For this problem the LMSD and ALMSD methods perform rather well. In the $n = 50$ experiment method ALMSD with $m = 3$ used 700 LMSD step lengths and 12 HLMSD step lengths and ALMSD with $m = 6$ used 477 LMSD step lengths and 10 HLMSD step lengths. In the $n = 200$ experiment method ALMSD with $m = 3$ used 1245 LMSD and 19 HLMSD step lengths and ALMSD with $m = 6$ used 848 LMSD step lengths and 3 HLMSD step lengths.

Note that HLMSD only has a small amount of nonmonotonic function evaluations, while BB1, LMSD and ALMSD display more nonmonotonic behavior. The limited memory methods for $m = 6$ perform better than the ones for $m = 3$ in this problem.

For problem XV we will consider $n = 100$ and $n = 10^5$. For $n = 100$ we compute the average number of function evaluations ($\#f$), nonmonotonic function evaluations (nm-$\#f$) and times the reduction condition is not satisfied (#red) over 1000 random starting vectors on the unit sphere. For $n = 10^5$ we run over 200 random starting vectors on the unit sphere and use $\texttt{tol} = 10^{-5}$. The results are shown in Table 6.7 and Table 6.8.

Table 6.7: Numerical results for test problem XV with $n = 100$

Method	$\#f$	nm-$\#f$	#red
BB1	112	16	15
BB2	82	5	2
ABB_{min} ($m = 3$)	60	3	1
LMSD ($m = 3$)	74	5	7
HLMSD ($m = 3$)	70	3	1
ALMSD ($m = 3$)	74	5	7
ABB_{min} ($m = 6$)	66	2	1
LMSD ($m = 6$)	65	3	2
HLMSD ($m = 6$)	66	2	2
ALMSD ($m = 6$)	65	3	2

Table 6.8: Numerical results for test problem XV with $n = 10^5$ (and $\texttt{tol} = 10^{-5}$)

Method	$\#f$	nm-$\#f$	#red
BB1	202	26	21
BB2	149	8	4
ABB_{min} ($m = 3$)	137	4	3
LMSD ($m = 3$)	176	14	14
HLMSD ($m = 3$)	133	1	2
ALMSD ($m = 3$)	175	14	14
ABB_{min} ($m = 6$)	138	3	2
LMSD ($m = 6$)	138	5	7
HLMSD ($m = 6$)	127	1	3
ALMSD ($m = 6$)	138	5	7

For $n = 100$ the ALMSD method was identical to the LMSD method for $m = 3$ and $m = 6$. For $n = 10^5$ the ALMSD method was identical to the LMSD method for $m = 6$ and used one HLMSD step length for $m = 3$. The limited memory methods seem to perform better for $m = 6$. Method ABB_{min} also performs rather well.

For problem XVI we will consider $n = 50$ and $n = 200$ and we compute the average number of function evaluations ($\#f$), nonmonotonic function evaluations (nm-$\#f$) and times the reduction condition is not satisfied (#red). The results are shown in Table 6.7 and Table 6.8.

Table 6.9: Numerical results for test problem XVI with $n = 50$

Method	$\#f$	nm-$\#f$	#red
BB1	122	19	13
BB2	92	7	4
$\text{ABB}_{\min}$ ($m = 3$)	67	4	3
LMSD ($m = 3$)	92	8	9
HLMSD ($m = 3$)	76	3	5
ALMSD ($m = 3$)	91	8	9
$\text{ABB}_{\min}$ ($m = 6$)	67	4	3
LMSD ($m = 6$)	69	4	4
HLMSD ($m = 6$)	71	2	3
ALMSD ($m = 6$)	70	4	4

Table 6.10: Numerical results for test problem XVI with $n = 200$

Method	$\#f$	nm-$\#f$	#red
BB1	336	44	42
BB2	229	12	8
$\text{ABB}_{\min}$ ($m = 3$)	147	6	5
LMSD ($m = 3$)	230	15	22
HLMSD ($m = 3$)	166	4	8
ALMSD ($m = 3$)	224	15	21
$\text{ABB}_{\min}$ ($m = 6$)	139	4	3
LMSD ($m = 6$)	161	8	10
HLMSD ($m = 6$)	182	6	9
ALMSD ($m = 6$)	180	8	10

For $n = 50$, the ALMSD method used 74 LMSD step lengths and 2 HLMSD step lengths when $m = 3$ and for $m = 6$ ALMSD used 60 LMSD step lengths and 4 HLMSD step lengths. For $n = 200$, the ALMSD method used 169 LMSD step lengths and 7 HLMSD step lengths when $m = 3$ and for $m = 6$ ALMSD used 145 LMSD step lengths and 13 HLMSD step lengths. We observe that $\text{ABB}_{\min}$ performs very well again. Compared to the BB1 and BB2 methods the limited memory methods also perform well. For $m = 3$ LMSD and ALMSD display more nonmonotonic behavior than for $m = 6$.

CHAPTER 7

Conclusions

In this work several gradient methods are presented which are based on the classical Steepest Descent, harmonic and the Barzilai–Borwein methods. The performance of these methods has been analyzed by applying it to a quadratic example. One of these methods is the adaptive Barzilai–Borwein method with minimum condition (ABB_{min}), which looks at the worst-case behavior of the Steepest Descent and harmonic method.

Due to the relation between the step lengths and the spectral properties of the Hessian of the objective function, the limited memory methods are introduced by Fletcher [16] and further analyzed by several researchers. Combining this analysis with the great performance of ABB_{min} we introduced a new method by applying the adaptive strategy of ABB_{min} to the LMSD and the HLMSD methods: the Adaptive Limited Memory Decent method (ALMSD).

By consulting recent research on generalizing the methods for general unconstrained nonlinear problems we ran numerical experiments to test the robustness, accuracy and efficiency of the generalized methods. To this end we gathered several nonlinear test cases. Depending on the parameter choices and starting vector choices each method showed different convergence behavior, but the limited memory methods were better in approximating the spectrum of the Hessian of the objective function and using this information in their step length choices.

In many numerical experiments the Barzilai–Borwein methods are less efficient compared to the adaptive and limited memory methods. We also found some test cases where the newly introduced ALMSD method computes less function evaluations until the stopping criterion is satisfied compared to the other known methods. Finally, compared to the Barzilai–Borwein methods these adaptive and limited memory methods display in many cases less nonmonotonic behavior.

CHAPTER 8

Future work

In this chapter some topics are listed that are not discussed in this thesis. We will discuss some loose ends and suggestions for future work.

Barzilai–Borwein research

For the Steepest Descent and harmonic methods we derived a convergence rate which describes the efficiency of the method. However, for the Barzilai–Borwein (and hence the limited memory methods) such rates are not yet derived. It might be interesting to look for such forms for a better understanding of the performance of the Barzilai–Borwein methods. Also, we have not yet found an application for the orthogonality property from Lemma 4.3.1, hence we leave this open for possible future research on the behavior of the BB1 method.

Asymptotic behavior

We have looked into the asymptotic behavior of the Steepest Descent method and investigated the influence of starting vectors close to that asymptotic behavior. From the numerical results we find that some limited memory methods find points near the global minimum rather fast, but actually getting close enough is the hard part. Therefore it might be interesting to look closer to the asymptotic behavior of the other discussed methods. One possible improvement can be made by using strategies such as preconditioning to get rid of the ill-conditioned matrices, such as the matrix R in the LMSD method. This might result in new possibilities, for example the possibility to vary the memory parameter in the limited memory methods some more to gain new insights.

Other adaptive strategies

The ALMSD method was based on the strategy of the ABB method. However, ABB_{min} was often more effective than ABB, which is caused by the information of more than one previous step lengths. The adaptive part of ALMSD is only applied at the beginning of a sweep. However, it might be interesting to look at applying the adaptive part at every iterative step. The adaptive strategy might be even better if more research is done on the worst-case scenarios of LMSD and HLMSD.

Perturbed gradient methods

If the the model is adjusted such that at each step the search direction has a small error $\varepsilon\, r_k$, where r_k is a normalized random vector from the Normal distribution with mean 0 and ε is a parameter that can be changed to observe the influence. In the quadratic example we receive the following expression for the search direction: $p_k = Ax_k + \varepsilon\, r_k$. This is an example of a perturbed gradient method. One can look at the discussed methods and investigate the influence of such perturbations to develop new insights in the performance of these methods.

However, this is just a small list of possible topics to investigate.

Bibliography

[1] H. Akaike, *On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method*, Ann. Inst. Stat. Math. Tokyo (1959), pp. 1–16.

[2] J. Barzilai, J. M. Borwein, *Two-point step size gradient methods*, IMA J. Num. Anal. 8 (1988), pp. 141–148.

[3] C. Beattie, *Harmonic Ritz and Lehmann bounds*, Num. Lin. Alg. Appl. (1993), pp. 20–22.

[4] A. Beck, *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*, (2014), pp. 55–57, 86–95.

[5] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press (2004), pp. 470–472.

[6] A. Cauchy, *Méthode générale pour la resolution des systèms d'équations simultanées*, Comp. Rend. Sci. Paris 25 (1847), pp. 536–538.

[7] F. Curtis, W. Guo, *Handling nonpositive curvature in a limited memory steepest descent method.*, IMA J. Num. Anal. 36 (2016), pp. 717–742.

[8] F. Curtis, W. Guo, *R-linear convergence of limited memory steepest descent*, IMA J. Num. Anal. 0 (2017), pp. 1–23.

[9] Y. Dai, *On the nonmonotone line search*, J. Opt. Th. Appl. (2002), pp. 315–330.

[10] Y. Dai, R. Fletcher, *Projected BarzilaiBorwein methods for large-scale box-constrained quadratic programming*, Num. Math. 100 (2005), pp. 21–47.

[11] Y. Dai, L. Liao, *R-linear convergence of the Barzilai and Borwein gradient method*, IMA J. Num. Anal. 22 (1) (2002), pp. 1–10.

[12] R. De Asmundis, D. di Serafino, W. W. Hager, G. Toraldo, H. Zhang., *An efficient gradient method using the Yuan steplength*, Comp. Opt. Appl. (2014), pp. 541–563.

[13] R. De Asmundis, D. di Serafino, G. Landi, *On the regularizing behavior of recent gradient methods in the solution of linear ill-posed problems*, (2014).

[14] R. De Asmundis, D. di Serafino, F. R. G. Toraldo, *On spectral properties of steepest descent methods*, IMA J. Num. Anal. (2013), pp. 1416–1435.

[15] J. Dieterich, H. Bernd, *Empirical review of standard benchmark functions using evolutionary global optimization*, CoRR 1207.4318 (2012).

[16] R. Fletcher, *A limited memory steepest descent method*, Math. Prog. Ser. A 135 (2012), pp. 413–436.

[17] R. Fletcher, M. Powell, *A rapidly convergent descent method for minimization*, Comp. J. 6 (1963), pp. 163–168.

[18] G. Frassoldati, L. Zanni, G. Zanghirati, *New adaptive stepsize selections in gradient methods*, J. Ind. Manag. Optim. 4 (2) (2008), pp. 299–312.

[19] A. Griewank, *Generalized descent for global optimization*, J. Opt. Th. Appl. 34 (1981), pp. 11–39.

[20] N. Higham, *Computing a nearest symmetric positive semidefinite matrix*, Lin. Alg. Appl. 103 (1988), pp. 103–118.

[21] R. Hill, B. Parlett *Refined interlacing properties*, SIAM J. Matrix Anal. Appl. 13 (1992), pp. 239–247.

[22] D. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill (1972), pp. 132–141.

[23] R. Jin, W. Chen, A. Sudjianto, *On sequential sampling for global metamodeling in engineering design*, ASME, 28th Design Automation Conference (2002).

[24] L. Kantorovich, *Funkcionallnyi analyz i prikladnaya matematika*, Uspehi Mat. Nauk. 28 (1945), pp. 89–185.

[25] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Natl Bur. Std. 45 (1950), pp. 255–282.

[26] H. Mühlenbein, D. Schomisch, *The Parallel Genetic Algorithm as Function Optimizer*, Par. Comp. 17 (1991), pp. 619–632.

[27] J. Nocedal, S. Wright, *Numerical Optimization*, Springer Series in Op. Res. (2006), pp. 14–39.

[28] C. Paige, B. Parlett, H. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Electron. Trans. Num. Anal. 7 (1998), pp. 18–39.

[29] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Opt. 7 (1997), pp. 26–33.

[30] M. Raydan, B. Svaiter, *Relaxed steepest descent and Cauchy–Barzilai–Borwein Method*, Comp. Opt. Appl. 21 (2002), pp. 155–167.

[31] H. Rosenbrock, *An automatic method for finding the greatest or least value of a function*, Comp. J. 3 (1960), pp. 175–184.

[32] D. di Serafino, V. Ruggiero, G. Toraldo, L. Zanni, *On the steplength selection in gradient methods for unconstrained optimization*, Appl. Math. Comp. 318 (2018), pp. 176–195.

[33] P. Toint, *Some numerical results using a sparse matrix updating formula in unconstrained optimization*, Math. Comp. 32 (1978), pp. 839–852.

[34] Y. Yuan, *A new stepsize for the steepest descent method*, J. Comp. Math. 24 (2006), pp. 149–156.

[35] B. Zhou, L. Gao, Y. Dai, *Gradient methods with adaptive step-sizes*, Comp. Opt. Appl. 35 (2006), pp. 69–86.